

# COMP 4: BENCHMARK QUIZ SYSTEM

*Damask Talary-Brown  
Candidate No. 5556  
Centre No. 64395  
Godalming College*

# Contents

---

<b>Analysis</b> .....	<b>4</b>
Project Problem Definition .....	4
Background to the Problem .....	4
Interview with Anne Lancaster - The Primary Client .....	5
Observation of Existing System .....	7
System Flowchart.....	10
Prospective Users and Acceptable Limitations.....	11
Document Specification Sheets - Volumetrics and Data Dictionaries.....	11
Data Sources and Destinations .....	23
Data Flow Diagrams .....	24
Entity Relationship Diagram .....	26
Tasks to be Computerised and the Benefits of Computerising .....	27
Objectives for the Proposed System.....	28
Potential Solutions and Justification of Chosen Solution .....	29
<b>Design</b> .....	<b>32</b>
IOPS Chart .....	32
System Flowchart.....	32
Data Flow Diagram.....	34
Data Dictionary and Validation .....	35
Justification of Included Question types.....	37
Stepwise Refinement .....	38
Database Normalisation and Design.....	39
Table Relationship Diagram .....	42
Sample of Possible SQL Queries.....	43
Top-Down Design.....	44
OOP Class Design .....	45
Definition of Record Structure.....	46
Security and Integrity of Data .....	46
User Interface Design.....	47
Form Navigation Design.....	55

Output Design .....	56
Algorithm Design.....	58
Random Ordering of Multiple Choice Answers .....	58
Question Text Search .....	58
Validating Question Entry .....	58
Importing Questions from a Text File .....	60
Checking for Repeated Questions .....	60
Converting Questions to Moodle’s Gift Format .....	61
Identification of Storage Media .....	62
Testing Plan.....	63
Input and Output Testing Design.....	63
Navigation Testing Design.....	66
<b>Testing .....</b>	<b>67</b>
Input and Output Testing.....	67
Trace Tables .....	74
Navigation Testing .....	78
<b>Maintenance .....</b>	<b>79</b>
Form Navigation Overview .....	79
Class Overview .....	80
Form Overview.....	83
Subroutine and Variable Overview .....	127
frmAddEditQuestion .....	127
frmAddLog.....	129
frmAddNewClass.....	129
frmClassLogs .....	130
frmFilterDifficulty.....	131
frmFilterType .....	131
frmFilterUnitTopic.....	131
frmHome.....	132
frmTextSearch.....	137
Detailed Algorithm Design .....	138
Post-Implementation System Overview .....	148
IOPS Chart .....	148

System Flowchart.....	148
<b>User Guide .....</b>	<b>150</b>
Installation Guidelines .....	152
Installation from a USB Flash-Drive .....	153
Using the System.....	156
Creating Questions.....	156
Deleting Questions.....	158
Adding and Removing Questions from the Quiz .....	159
Adding and Deleting Classes .....	160
Adding and Deleting Class Logs .....	161
Searching Questions .....	162
Filtering Questions by Unit, Topic or Difficulty.....	163
Exporting Quizzes.....	165
Uploading Quizzes to Moodle.....	166
Troubleshooting.....	168
<b>Appraisal.....</b>	<b>170</b>
Comparison of Project Performance Against Objectives .....	170
Potential for Future Developments .....	172
Analysis of User Feedback .....	173
<b>Testing Outcome Screenshots.....</b>	<b>177</b>

# Analysis

---

## Project Problem Definition

Client: Godalming College Physics department

- Anne Lancaster (Primary)
- Phillip Morgan and Joe McCarthy-Holland (Secondary)

Contact: Anne Lancaster  
Tuesley Lane,  
Godalming,  
Surrey,  
GU7 1RS  
(01483 423526)

## Background to the Problem

The Godalming College physics department is a small department, comprising three teachers, technicians, five lower 6th and three upper 6th classes. One of the teachers - Anne Lancaster - sets exam style starter questions at the beginning of each lesson, based on the current topic she is teaching.

The current system Anne uses is the same system the department uses to generate mock exams from the AQA question bank. Because the software is not bespoke, it lacks functionality in some areas. Questions are searched for and compiled into a document which can be printed or saved, along with a mark scheme. There are only a finite number of questions, with no way to add more or change the details of the existing ones. There is also no way to export the generated quizzes to the college's Moodle, or to keep a record of the questions which have already been set.

## Interview with Anne Lancaster - The Primary Client

What's your existing system for setting classes questions?

AL: We use AQA's Exampro to select existing questions from past papers. Sometimes we write them ourselves using word processors.

What are the benefits of using the current system?

AL: It's easy to scroll through the lists of past questions and filter them by type or unit to select the ones you want. The mark schemes are easily accessible as well, and when you've chosen a set of questions you can save them to use again with a different class.

What are the drawbacks of the current system?

AL: We're limited by the questions, because they all come from past papers there isn't any way of adding new ones into the system. There aren't many questions on certain topics which can be a problem when we're making quizzes or revision booklets. Also, the search filter could be better. It would help to be able to search by more than just the unit or topic, and have more types of questions so you could search by type. The software we use also doesn't notify us if we're about to add a question to the document that's already there.

Which new features would be the most important to you?

AL: We should be able to add our own questions to the system instead of simply using past paper ones. Being able to upload the quizzes to Moodle without having to type them out again would also be useful, so that the students could complete the quizzes in their own time and have them automatically marked in seconds rather than bringing them in or us marking them through Moodle. At the same time though, we still want to be able to print normal text documents or save them to use as paper copies.

Which existing features do you find the most useful?

AL: The existing features that I'd like to see in the new software are the ability to view and print mark schemes and search by unit or topic for questions which have already been added. It would be nice to have some sort of difficulty rating for each question generated, too.

How many members of staff would be using the new system?

AL: There are three teachers in this department.

Which course and units would you like the system to work with?

AL: AS and A2 Physics - every unit.

You mentioned different types of questions, what did you mean by that?

AL: I'd like the questions to be similar to the exam style in that there are different formats; numeric calculations, simple definitions of terms and multiple choice answers.

How would you like the students to answer these questions?

AL: Currently, I display them on the interactive whiteboard or print each student a copy and they write their answers on paper. They then swap with the person sitting next to them and mark their partner's.

## Observation of Existing System

I observed two of the teachers using the existing system to get a better understanding of how the users interact with it on a daily basis, and identify the problems they face while doing this.

NB. Due to access issues, the screenshots shown below are from the GCE Computing specification rather than GCE Physics, but other than this, the systems are identical.

The current process is as follows:

1. Questions are selected from the database
  - 1.1. The teacher looks at their paper-based records to check which chapter of the textbook their class most recently covered. There is no way of keeping a record of this from within the system.
  - 1.2. Exampro is opened, and the teacher uses the search filter to search for the relevant questions by topic [figure 1] (for broader quizzes, they searched by unit of work [figure 2]). There are few other ways to filter the questions, which for a large database can be problematic and time consuming. [figure 3]
  - 1.3. If the questions are for a starter exercise, ten are selected. [figure 4] If they're for revision booklets, it could be a hundred or more. A problem arises here, because the existing system doesn't notify the user if they're adding a question which is already in the current document.

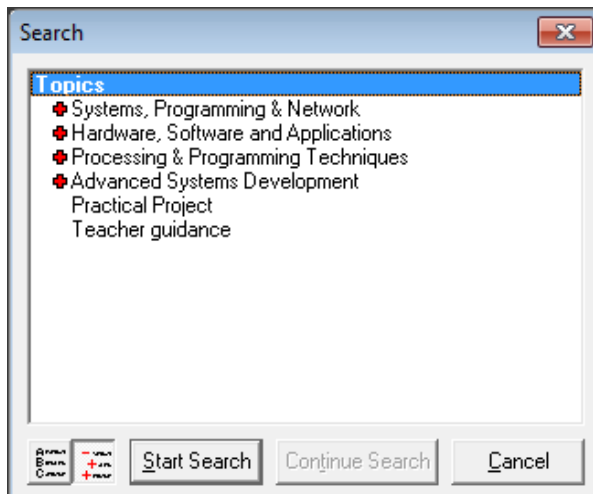


Figure 1

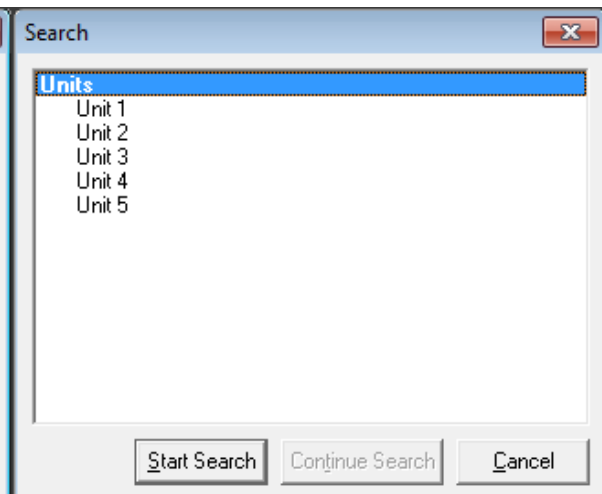


Figure 2



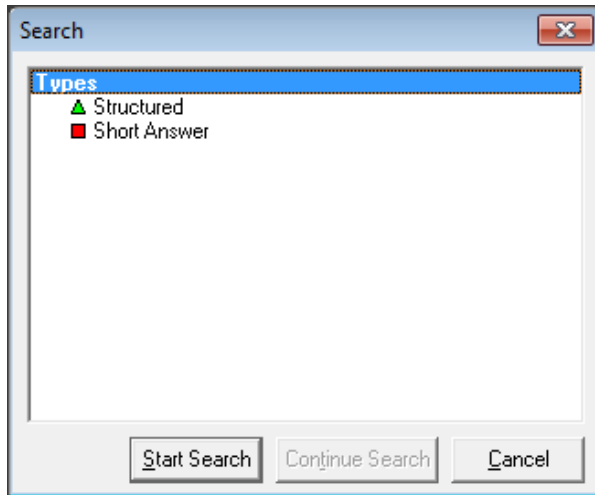


Figure 3

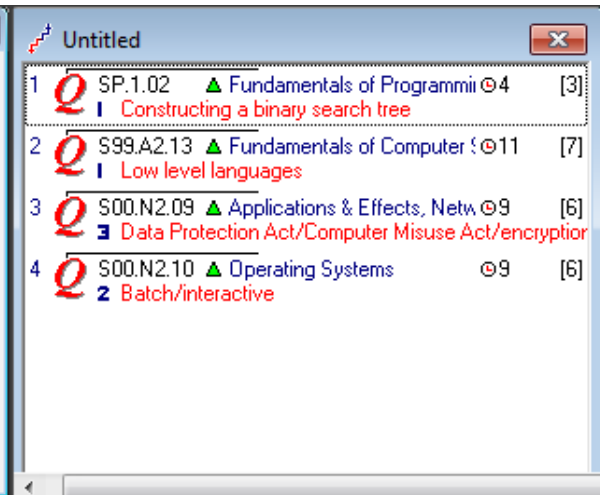


Figure 4

2. Questions are checked against existing quiz documents
  - 2.1. If the topic has taken more than one lesson to teach, the teacher opens the previous quizzes to check that none of the questions they selected have been assigned before.
  - 2.2. If there are any duplicates, the teacher replaces the questions.
  - 2.3. The questions [figure 5] and generated mark scheme [figure 6] are saved. There are alternative options, such as to print them, but no option to export them in a format compatible with the college's online Moodle.

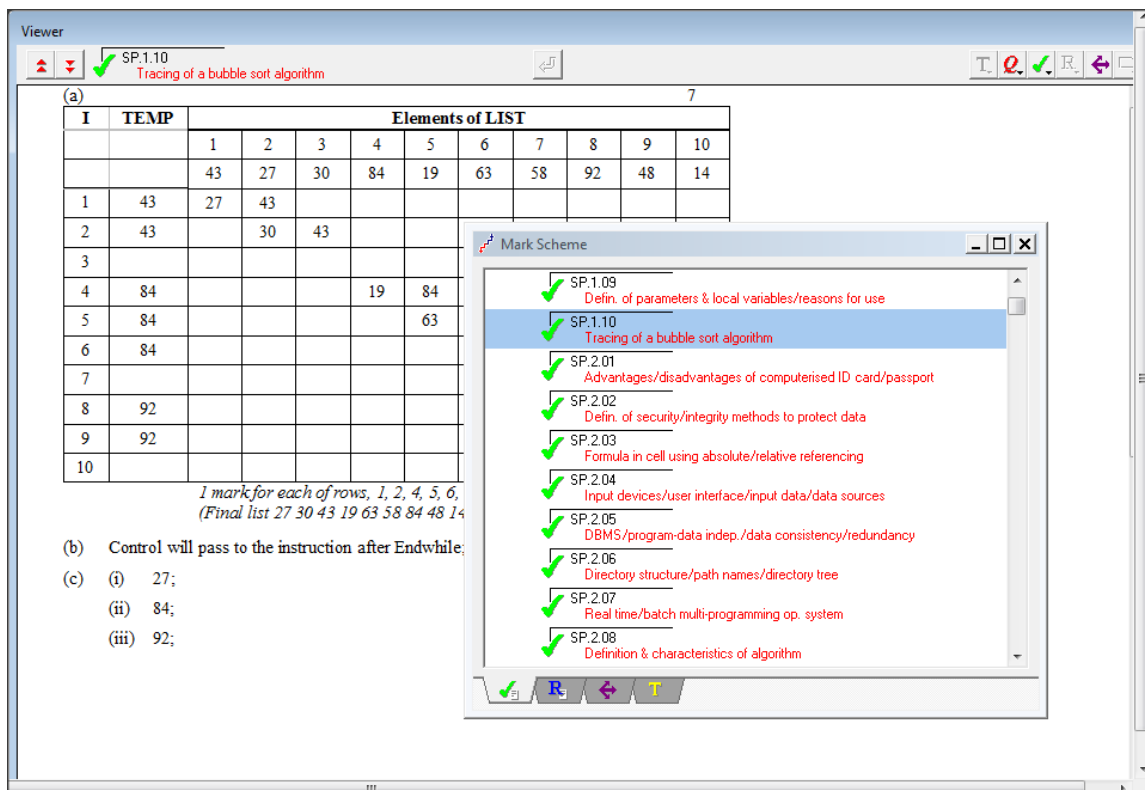


Figure 5

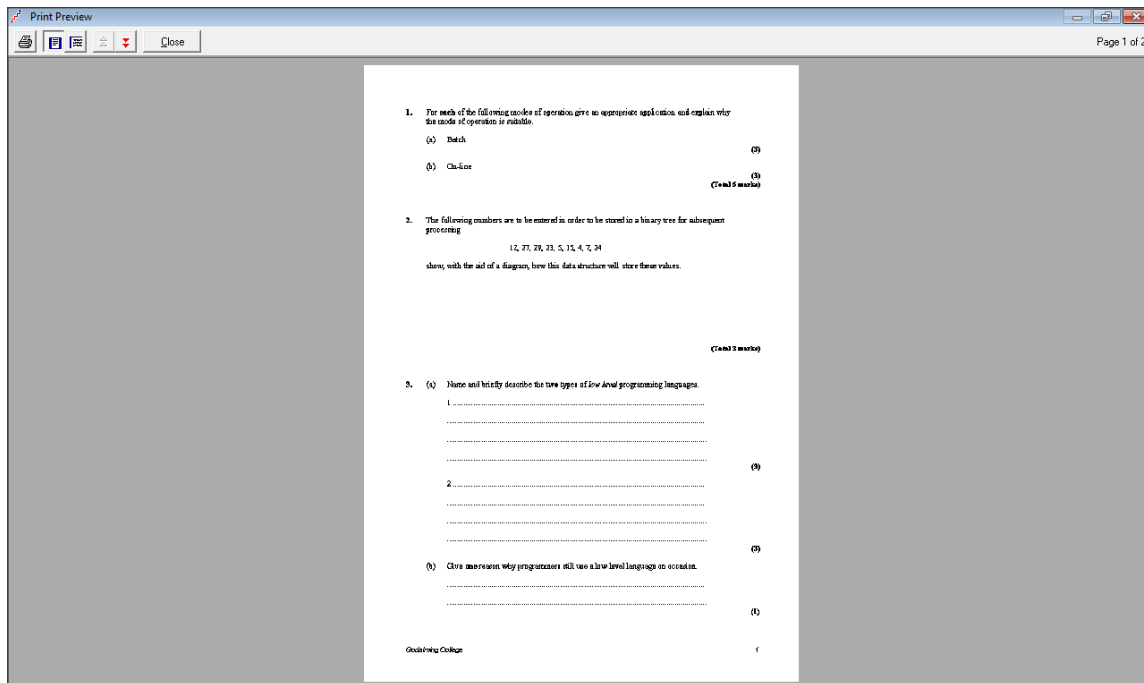
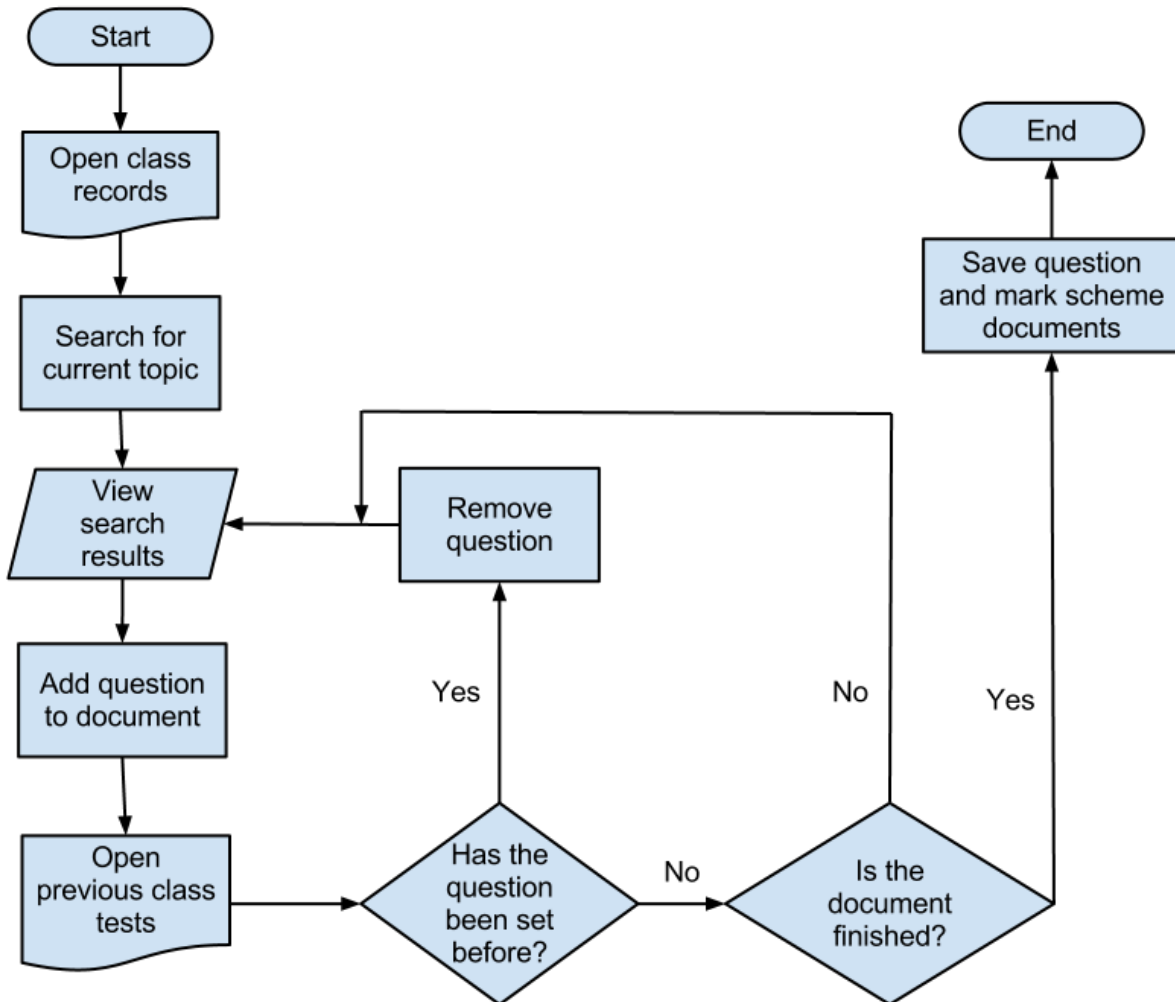


Figure 6

- Questions are set
  - The questions are shown to the class on the interactive whiteboard. The students complete them on paper. Although the eventual scores will be saved, the papers are rarely kept. There is a student database integrated into the existing system though this is rarely used, if ever.
  - The students swap quizzes and the mark scheme is displayed so they can mark them.
  - Anne collects the answers in a spreadsheet which contains every student's score for all quizzes set throughout the year. There is no way to normalize the scores in cases of particularly difficult topics or questions, so isn't always a true representation of the student's ability.

### System Flowchart



The flowchart models the system as it is now. This enables me to see the inefficient parts of the process - for example, the user has to check every time they add a question to a document that the question has not been added before.

## Prospective Users and Acceptable Limitations

At present, there would be at most three users of the system; the teachers in the physics department, with Anne Lancaster being the primary user. All of the teachers are computer literate to a level which means their skills shouldn't limit the complexity of the new system. Nevertheless, it is important for the staff to spend as little time as possible navigating the system so that it doesn't cut into lesson time, so ease-of-use is still an important factor.

Although the students aren't the primary clients, they will be directly affected by the questions produced by the system, so the question types should be of a similar format to those they are or will be familiar with from exam papers.

The limitations to the system are as follows:

- Hardware and software constraints - There is only a limited amount of hardware in the physics department for the teachers to use, and a limited amount of software available for me to design the new system. This rules out the use of mobile applications as not all teachers have smart phones or tablets.
- My skills and knowledge - The problem can't be too complex for me to be able to solve using the resources I have available. I'll be creating the solution in VB.NET and so could write the program in a number of different paradigms.
- Moodle constraints – Although complex question types and questions involving images or diagrams with labeling do feature in exam papers, I'm limited to using question types that Moodle can not only recognize and support but also automatically mark rather than send to a teacher for feedback.
- Time constraints - The system needs to be completed by Easter.

## Document Specification Sheets - Volumetrics and Data Dictionaries

The documents below are examples of those currently in different parts of the system and identify both common and format-specific data. This will allow me to identify the data and data types that will need to be included in the new system.

Density Question Sheet.

DENSITY ①

Data:	Material	Density / kg m <sup>-3</sup>
	Glass	2500
	Aluminium	2700
	Steel	7700
	Gold	19300
	<i>the densest element</i> → Osmium	22500
	Balsa Wood	150
	<i>at 0°C and 1 atmosphere</i> → Air	1.29
	<i>at 100°C and 1 atmosphere</i> → Air	0.95
	<i>at 4°C and 1 atmosphere</i> → Water	1000
	<i>at 100°C and 1 atmosphere</i> → Water	958

Volume of a cylinder  $\pi r^2 l$

Volume of a sphere  $\frac{4}{3} \pi r^3$

1.
  - a. ③ A cube of edge 8 cm is made from balsa wood. Calculate its volume, and its mass.
  - b. ④ Calculate the mass of a similar cube made from osmium.
  - c. ⑤ Divide mass (b) by mass (a). What should this equal? Does it?
  - d. ⑥ A third similar cube has a mass of 1.28 kg. Calculate its density. From what might it be made?
2. ⑦ A steel rod of diameter 15 mm has a mass of 544 g. Calculate the length of the rod.
3. ⑧ The *nucleus* of an iron atom is spherical and has a radius of  $4.6 \times 10^{-15}$  m; the mass of the nucleus is  $9.5 \times 10^{-26}$  kg. Calculate the density of the nuclear material.
4. ⑨ One litre of water is heated from 4°C to 100°C at constant pressure. Calculate the new volume.
5. ⑩ A living room measures 4 m wide x 7 m long x 3 m high. *Estimate* the mass of air in the room on a warm day. (NB. Estimate means make as good an approximate calculation as you can, stating any assumptions you need to make.)

⑬ EXTENSION QUESTIONS (Try these if you are happy you have understood the ones above)

6. ⑪ A porous metal cube of side 100 mm is made from metal of density  $8000 \text{ kg m}^{-3}$  and has a mass of 7.2 kg. Calculate the volume of the pores in the metal, assuming you can ignore the mass of the trapped air.
7. ⑫ An alloy made from osmium and gold has a density of  $20200 \text{ kg m}^{-3}$ . Calculate the percentage by volume of gold in the alloy.

Volumetrics					
Document description	System	Document	Name	Sheet	
Density Question Sheet	Physics department quizzes	Quiz	Density	1/1	
Stationery ref.	Size	Number of parts	Method of preparation		
n/a	A4	1	Manually typed		
Filing sequence		Medium	Prepared by		
n/a		Digital	Anne Lancaster		
Frequency of preparation		Retention period	Location of file		
Once		Indefinite	Digital copy on the college network		
Volume	Minimum	Maximum	Av/Abs	Growth rate/fluctuations	
	1 per lesson	1	1	No fluctuations	
Users/receipts	Purpose			Frequency of use	
Anne Lancaster Physics students	Assigning questions Answering questions			Once Once	
Data Dictionary					
Ref	Name	Data Type	Length	Occurrence	Source of data
1	Title	String	7	1	Teacher
2	Data	String	400	1	Teacher
3	Question 1a	String	80	1	Teacher
4	Question 1b	String	55	1	Teacher
5	Question 1c	String	60	1	Teacher
6	Question 1d	String	96	1	Teacher
7	Question 2	String	83	1	Teacher
8	Question 3	String	165	1	Teacher
9	Question 4	String	95	1	Teacher
10	Question 5	String	222	1	Teacher
11	Question 6	String	203	1	Teacher
12	Question 7	String	122	1	Teacher
13	Extensions	String	83	1	Teacher

Momentum Question Sheet.

**Momentum Questions** ①

② 1. For the two physical quantities, impulse and force, which one of the following is correct?

- ③ A Impulse is a scalar and momentum is a scalar.
- ④ B Impulse is a scalar and force is a vector and momentum is a vector.
- ⑤ C Impulse is a vector and momentum is a vector and force is a scalar.
- ⑥ D Impulse is a vector and force is a vector and momentum is a vector.

⑦ (Total 1 mark)

2. In a vehicle impact test, a car of mass 1200 kg travelling at a velocity of  $18 \text{ ms}^{-1}$  is stopped by a large concrete block. A force meter attached to the block is used to measure the average force of the impact. The force meter measured an average force of 240 kN. What was the duration of the impact?

- A 0.090 s
- B 0.18 s
- C 0.90 s
- D 1.8 s

(Total 1 mark)

3. Water of density  $1000 \text{ kg m}^{-3}$  flows out of a garden hose of cross-sectional area  $7.2 \times 10^{-4} \text{ m}^2$  at a rate of  $2.0 \times 10^{-4} \text{ m}^3$  per second. How much momentum is carried by the water leaving the hose per second?

- A  $5.6 \times 10^{-5} \text{ N s}$
- B  $5.6 \times 10^{-2} \text{ N s}$
- C 0.20 N s
- D 0.72 N s

(Total 1 mark)

<b>Volumetrics</b>						
Document description		System		Document	Name	Sheet
Momentum Question Sheet		Physics department quizzes		Quiz	Momentum Questions	1/1
Stationery ref.		Size		Number of parts	Method of preparation	
n/a		A4		1	Compiled in Exampro	
Filing sequence			Medium		Prepared by	
n/a			Digital		Anne Lancaster	
Frequency of preparation			Retention period		Location of file	
Once			Indefinite		Digital copy on the college network	
Volume	Minimum		Maximum	Av/Abs	Growth rate/fluctuations	
	1 per lesson		1	1	No fluctuations	
Users/receipts		Purpose			Frequency of use	
Anne Lancaster Physics students		Assigning questions Answering questions			Once Once	
<b>Data Dictionary</b>						
Ref	Name	Data Type	Length	Occurrence	Source of data	
1	Title	String	18	1	Teacher	
2	Question text	String	100-250	3	Exampro	
3	Answer A	String	5-67	3	Exampro	
4	Answer B	String	5-67	3	Exampro	
5	Answer C	String	5-67	3	Exampro	
6	Answer D	String	5-67	3	Exampro	
7	Question mark	String	5-67	3	Exampro	



## Particle Physics Moodle Questions.

Home ▶ 11-3:PHA ▶ Quizzes ▶ Particle Physics Student Quiz ▶ Attempt 1

**1** Particle Physics Student Quiz - Attempt 1**1** **2** Which of these are conserved?

Marks: -/1

Choose one answer. **3**

- a. Bosons **4**
- b. Lepton number **5**
- c. Meson Number **6**
- d. speed before and after collision **7**

**2** what are the 3 quarks that make up a proton

Marks: -/1

Choose one answer.

- a. down up strange
- b. up up down
- c. down down up
- d. up up strange

**3** what are the three quarks we need to know?

Marks: -/1

Choose one answer.

- a. Up, Down, Left.
- b. Down, Right, Crazy
- c. anti up, strange right, forwards.
- d. Strange, Up, Down.

**4** WHICH OF THE FOLLOWING ARE MESONS?

Marks: -/1

Choose one answer.

- a. Neutrino, neutron, Pion
- b. Neutrino, Kaon, Electron.
- c. Pion, Kaon, Tau
- d. Proton, Electron, Neutron.

<b>Volumetrics</b>						
Document description		System		Document	Name	Sheet
Particle Physics Moodle Questions		Physics department quizzes		Quiz	Particle Physics Student Quiz	1/7
Stationery ref.		Size		Number of parts	Method of preparation	
n/a		n/a (Web-based)		1	Manually typed into Moodle's question creator	
Filing sequence			Medium		Prepared by	
n/a			Digital		Anne Lancaster	
Frequency of preparation			Retention period		Location of file	
Once			Indefinite		College Moodle database	
Volume	Minimum		Maximum	Av/Abs	Growth rate/fluctuations	
	2 per term		5	3	Fluctuations: Jan/Feb and May/June exam sessions	
Users/receipts		Purpose				Frequency of use
Anne Lancaster Physics students		Assigning questions Answering questions				Once Once
<b>Data Dictionary</b>						
Ref	Name	Data Type	Length	Occurrence	Source of data	
1	Title	String	18	1	Teacher	
2	Question text	String	100-250	4	Teacher	
3	Mark	Integer	1	4	Teacher	
4	Answer A	String	8-30	4	Teacher	
5	Answer B	String	8-30	4	Teacher	
6	Answer C	String	8-30	4	Teacher	
7	Answer D	String	8-30	4	Teacher	

Refraction Mark Scheme.

**① Refraction Problems – Mark Scheme**

- ② 1. Calculate the refractive index for glass in which the speed of light is  $2.25 \times 10^8 \text{ ms}^{-1}$

$$\textcircled{3} \quad n = \frac{3.00 \times 10^8}{2.25 \times 10^8} = 1.33 \textcircled{5}$$

④ [1]

2. Calculate the speed of light in diamond whose refractive index is 2.42

$$v = \frac{3.00 \times 10^8}{2.42} = 1.24 \times 10^8 \text{ ms}^{-1}$$

[1]

3. Calculate the refractive index for paraffin in which the speed of light is  $2.08 \times 10^8 \text{ ms}^{-1}$

$$n = \frac{3.00 \times 10^8}{2.08 \times 10^8} = 1.44$$

[1]

4. Calculate the speed of light in perspex whose refractive index is 1.49

$$v = \frac{3.00 \times 10^8}{1.49} = 2.01 \times 10^8 \text{ ms}^{-1}$$

[1]

5. Calculate the refractive index for a medium given that the angle of incidence is  $30^\circ$  and the angle of refraction is  $24^\circ$ .

$$n = \frac{\sin 30^\circ}{\sin 24^\circ} = \frac{0.500}{0.407} = 1.23$$

[1]

6. Calculate the refractive index for glass if the angle of incidence is  $45^\circ$  and the angle of refraction is  $27^\circ$ .

$$n = \frac{\sin 45^\circ}{\sin 27^\circ} = \frac{0.707}{0.454} = 1.56$$

[1]

7. Calculate the angle of refraction in water if the angle of incidence in air is  $24^\circ$  and the refractive index is 1.33.

$$\sin r = \frac{\sin i}{n} = \frac{\sin 24^\circ}{1.33} = \frac{0.407}{1.33} = 0.306 \rightarrow r = 17.8^\circ = 18^\circ$$

[1]

8. Calculate the angle of incidence in air if the angle of refraction is  $37^\circ$  and the refractive index of the material is 1.65.

$$\sin i = n \sin r = 1.65 \sin 37^\circ = 1.65 \times 0.602 = 0.993 \rightarrow i = 83.2^\circ \dots = 83^\circ$$

[1]

9. A ray passes from air to water, calculate the angle in air if the angle in water is  $45^\circ$

$$1.33 \sin 45 = 1.00 \sin \theta_{\text{air}} \rightarrow \sin \theta_{\text{air}} = 1.33 \times 0.707 = 0.940$$

$$\theta_{\text{air}} = 70^\circ$$

[1]

10. A ray passes from air to glass, calculate the angle in the glass if the angle in air is  $60^\circ$

$$1.00 \sin 60 = 1.50 \sin \theta_{\text{glass}} \rightarrow \sin \theta_{\text{glass}} = \frac{1.00 \times 0.866}{1.50} = 0.577$$

$$\theta_{\text{glass}} = 35^\circ$$

[1]

11. A ray passes from water to glass, calculate the angle in glass if the angle in water is  $55^\circ$

$$1.33 \sin 55 = 1.50 \sin \theta_{\text{glass}} \rightarrow \sin \theta_{\text{glass}} = \frac{1.33 \times 0.819}{1.50} = 0.726$$

$$\theta_{\text{glass}} = 46.578 \dots^\circ = 47^\circ$$

[1]

12. A ray passes from glass into water. Calculate the angle in the glass if the angle in water is  $90^\circ$ . What is this angle called?

$$1.33 \sin 90 = 1.50 \sin \theta_{\text{glass}} \rightarrow \sin \theta_{\text{glass}} = \frac{1.33 \times 1.00}{1.50} = 0.887$$

$$\theta_{\text{glass}} = 62.457 \dots^\circ = 62^\circ$$

[1]

Volumetrics					
Document description	System	Document	Name	Sheet	
Refraction Mark Scheme	Physics department quizzes	Mark Scheme	Refraction Problems – Mark Scheme	1/7	
Stationery ref.	Size	Number of parts	Method of preparation		
n/a	2 sides A4	1	Compiled in Exampro		
Filing sequence		Medium	Prepared by		
n/a		Digital	Anne Lancaster		
Frequency of preparation		Retention period	Location of file		
Once		Indefinite	Digital copy on the college network		
Volume	Minimum	Maximum	Av/Abs	Growth rate/fluctuations	
	1 per quiz per lesson	1	1	No Fluctuations	
Users/receipts	Purpose			Frequency of use	
Anne Lancaster Physics students	Assigning questions Answering questions			Once Once	
Data Dictionary					
Ref	Name	Data Type	Length	Occurrence	Source of data
1	Title	String	31	1	Exampro
2	Question text	String	100-250	12	Exampro
3	Calculation	String	200-500	12	Teacher
4	Mark	Integer	1	12	Exampro
5	Answer	Integer		12	Exampro

Teacher's Mark Book.

Subject: 3PHA ①

Mechanics ③

Group: A1 ②

⑤ Date:

⑥

AVE GCSE	ALIS	Induction	IND GRADE	SI Poster	8.1 SQ 113	8.2 SQ 115	Im Not sheet	Chpt 8 ESQ
		72				21	22	32
7.8	B+	58	A	A	A	15	12	30
65	D+	/			C	13	8	
58	E+	50	C	A	A	17		30
61	D-							
7.5	B	48	C	A	A	17	17	35
64	D+	44	C	A	B	19	19	
7.0	C+	48	C	A	B			
66	C	53	B	A		15	14	24
7.5	B	50	C	A		10	20	36
68	C	40	D		10	19	18	30
62	D	50	C	A	A	16	15	
68	C	45	C	A	A	19	15	36
67	C	66	A	A	B		9	
60	D-	9	U	A	C	12		18
7.0	C+	20	U	A	A		20	28
62	D	49	C	A	A	15	14	30
7.2	C+	38	D	A	A	11	22	28
63	D	28	U	A				
60	D-	38	D	A	B			

⑦

<b>Volumetrics</b>						
Document description		System		Document	Name	Sheet
Teacher's mark book		Physics department quizzes		Mark book page	Mechanics (marks)	1/1
Stationery ref.		Size		Number of parts	Method of preparation	
n/a		1 side A4		1	Filled in by the physics teacher after every text/quiz	
Filing sequence			Medium		Prepared by	
n/a			Manual		Anne Lancaster	
Frequency of preparation			Retention period		Location of file	
Every lesson			Indefinite		Teacher's mark book, kept in physics dept.	
Volume	Minimum		Maximum	Av/Abs	Growth rate/fluctuations	
	1 column per lesson		1	1	No Fluctuations	
Users/receipts		Purpose			Frequency of use	
Anne Lancaster		Recording students' marks Checking the most recent topic covered Referring to marks when writing reports etc.			Once Once Up to 3 times/year	
<b>Data Dictionary</b>						
Ref	Name	Data Type	Length	Occurrence	Source of data	
1	Unit code	String	4	1	Timetable codes	
2	Group name	String	2	1	Timetable codes	
3	Topic title	String	9	1	Teacher/Specification	
4	Individual lesson columns	String	4-10	9	Teacher	
5	Lesson dates	Date	8	9	Teacher	
6	Students' names	String	10-30	20	Teacher/Register	
7	Individual scores/grades	String	1-2	180	Teacher	

## Data Sources and Destinations

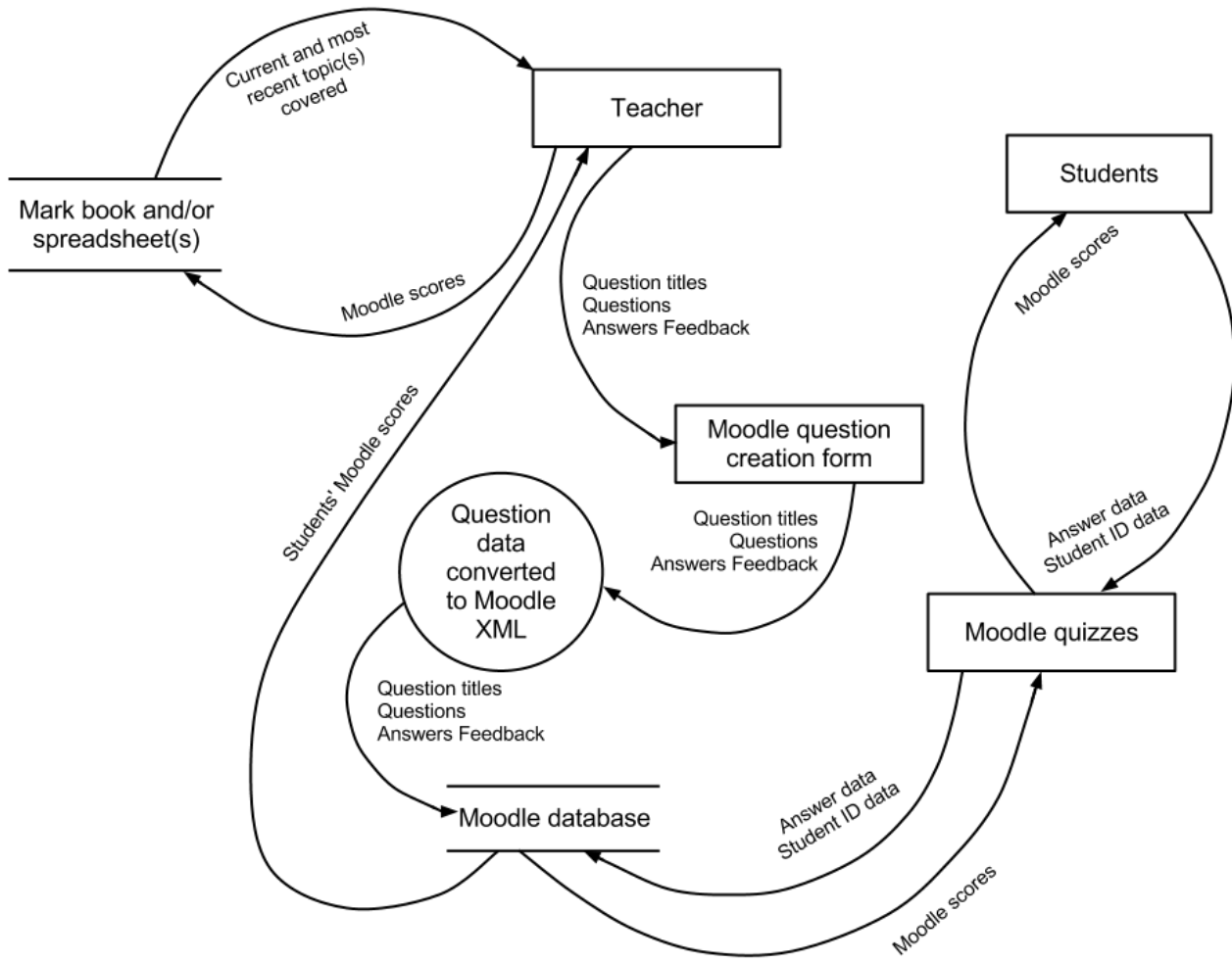
In the Existing System		
What is it?	Source	Destination
Paper-based question	Exampro database	Printed and given to students
Online Question	Created by the teacher	Stored in the Moodle database
Paper-based mark scheme	Exampro database	Printed and given to students
Online mark scheme	Created by the teacher	Stored in the Moodle database
Current topic details	Noted by the teacher	Teacher's records
Students' scores	Quiz marks/Moodle database	Teacher's records

In the Proposed System		
What is it?	Source	Destination
Paper-based question	Created by the teacher	Printed and given to students
Online Question	Created by the teacher	Converted by the system and saved on the network to be uploaded to Moodle
Paper-based mark scheme	Created by the teacher	Printed and given to students
Online mark scheme	Created by the teacher	Converted by the system and saved on the network to be uploaded to Moodle
Current topic details	Noted by the teacher	Stored in a log on the system
Students' scores	Quiz marks/Moodle database	Teacher's records
Question difficulty ratings	Created by the teacher	Stored in the database and/or printed with the mark scheme(s)

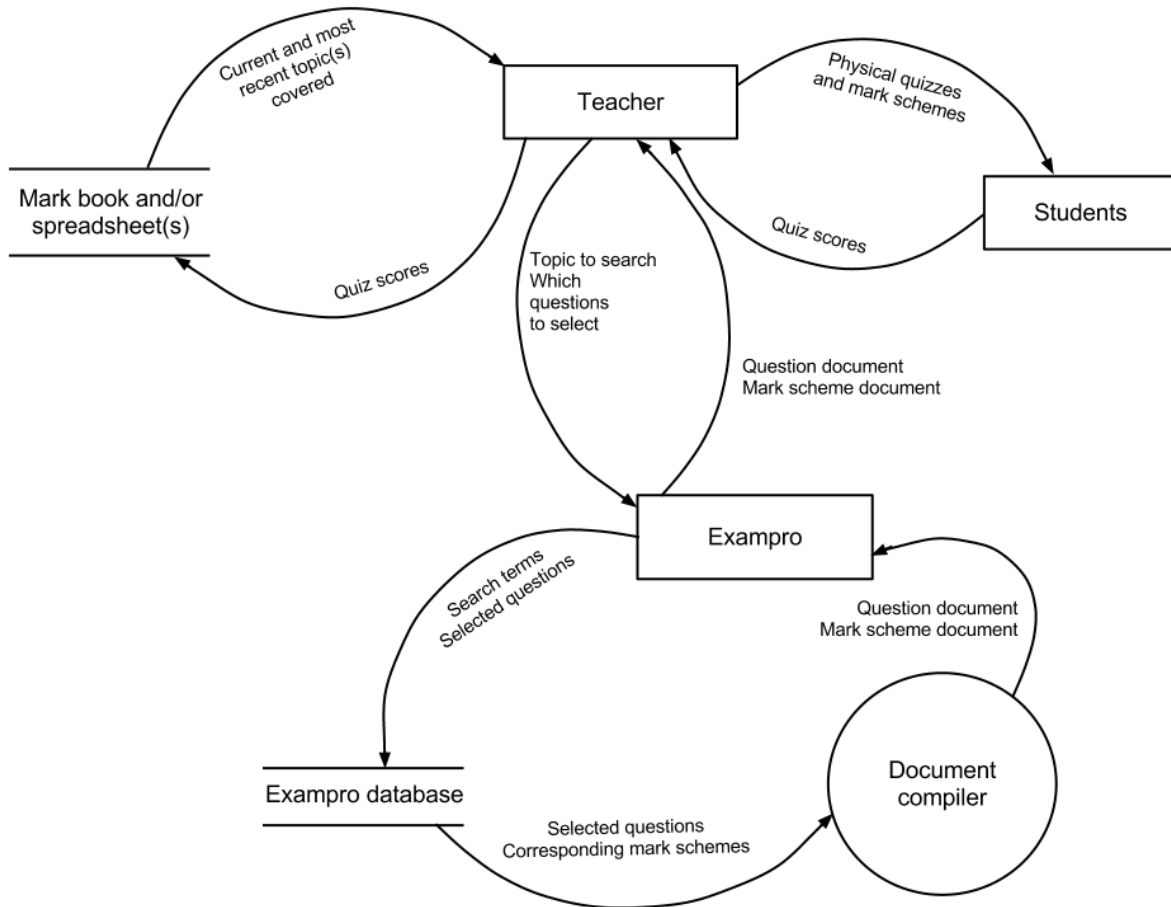


## Data Flow Diagrams

Between the Teacher, Students and Moodle.

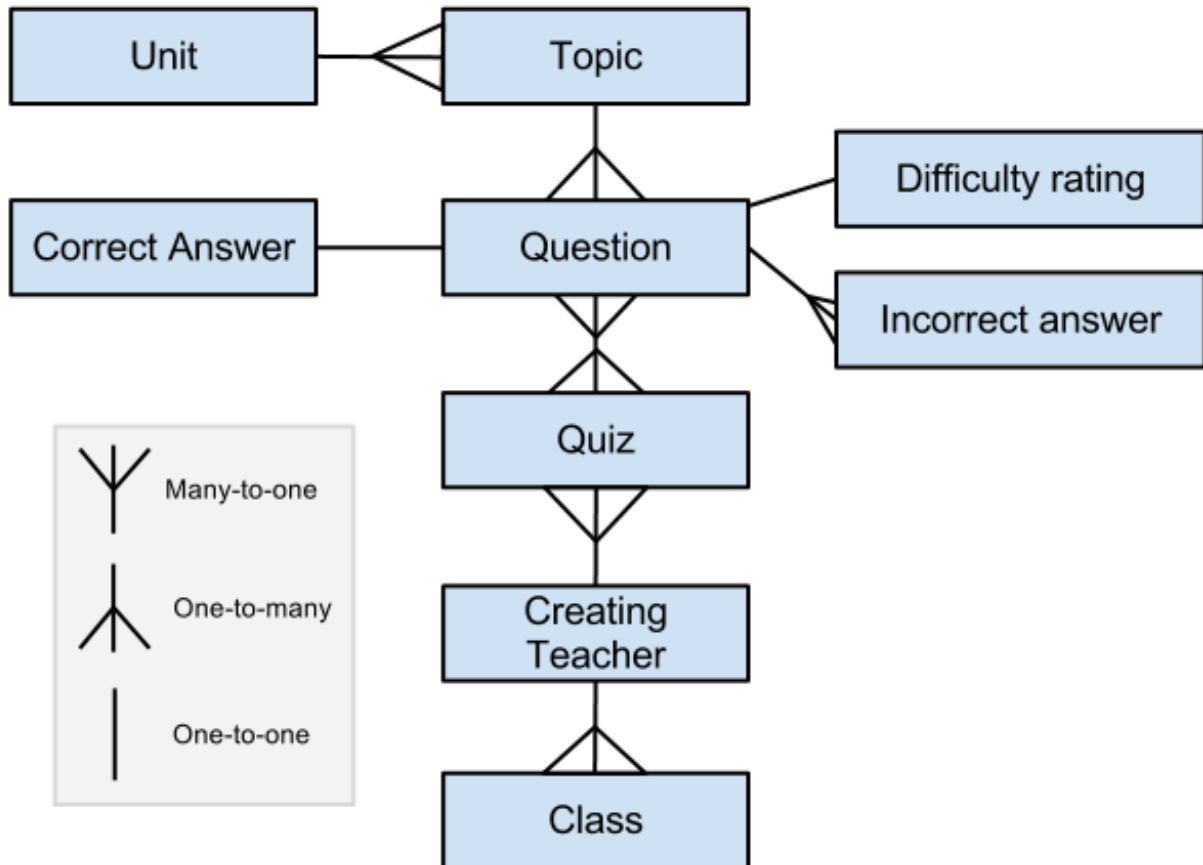


Between the Teacher, Students and Exampro.



Both the data sources and destinations tables and the data flow diagrams illustrate the processes and flow of data in the current system between the teacher, the Exampro question database, the teacher's records, and the students. They are essentially both data models of the observation I conducted of the teachers using the system, and allow me to see which parts of the system use which types of data and track its path.

## Entity Relationship Diagram



If I choose to solve the problem using a database, the diagram above identifies the relationships between the entities in the system. A question has one topic, one rating, one correct answer, and many incorrect answers. It can belong to many quizzes, and a quiz contains many questions. A topic has many questions but only one unit, and a unit has many topics.

## Tasks to be Computerised and the Benefits of Computerising

- Allowing questions to be exported in a format accepted by Moodle
- Allowing all questions to be searchable by type, difficulty, etc.
- Providing templates for teachers to create quizzes and save questions to be stored
- Allowing the user to keep a record within the system of the most recent topics covered by each class
- Allowing the user to assign questions a difficulty rating in order to normalise scores when marking

### Qualitative Benefits:

- A more user-friendly interface
- Questions won't have to be typed out manually to export Moodle
- Greater flexibility of the types of questions that can be set
- Complete flexibility of the questions themselves
- Each record can be stored separately, allowing for a changing number of classes

### Quantitative Benefits:

- A potentially unlimited number of questions can be stored in the system
- A potentially unlimited number of classes can be recorded

## Objectives for the Proposed System

I have learned a lot about the existing system both from the staff observations and by using the software myself. I was able to observe features which worked well, features that were unnecessary for the users' needs and features which the system lacked altogether.

1. The system must be able to store all the relevant details about every question entered:
  2. Question, answer, unit, topics and type should be mandatory for each entry.
  3. For multiple choice questions, the incorrect answers should also be stored.
  4. There should be a default difficulty rating for each question (out of five) which can be edited or left as the default.
  5. There should be different question types available similar to the types that feature in the exams – short answer, calculation, etc.
6. The questions must be searchable and displayable by question type, unit, topic, and difficulty rating.
7. Users must be able to search all questions for a specific string.
8. Searching questions should involve the minimum free text entry to save time and minimise errors. This could be implemented using radio buttons or drop-down menus for selection.
9. The user must be able to edit existing questions or delete them from the database.
10. The user should be able to preview the answer to each question that is displayed.
11. The user must be able to add records of new classes to the system.
12. Each class stored must have its own 'log' which would allow free text entry similar to a teacher's diary. The user must be able to record recently set questions from the textbook or chapters covered and retrieve this information when assigning new questions.
13. The user must be able to delete classes or class logs from the system.
14. The user should be able to create a quiz by selecting questions from the database and/or adding new questions.
15. The system must notify the user if they are adding a question to a quiz which already contains it to prevent unnecessary duplicates.
16. The user should be able to export text-based quizzes as text files to be emailed or printed.
17. Text-based quizzes should have separate numbered mark schemes unlike the integrated Moodle quizzes.
18. The system should be able to export quizzes with integrated answers in a format and the UTF-8 encoding recognised by Moodle.
19. Users should be able to see at all times a 'count' of how many questions they are viewing and how many are in the current quiz.
20. The system must include button shortcuts for common actions such as adding and removing questions to or from quizzes in order to reduce the time spent navigating menus.

## Potential Solutions and Justification of Chosen Solution

Potential Solutions:

1. A manual system, which would allow teachers complete flexibility with their questions as they could simply word-process them using a template created by the system. Any question type supported by Moodle could be saved, and teachers would have greater control over the formatting of the text-based quizzes and mark schemes. Questions could then be copied and pasted into the Moodle question creator form to allow students to access them online.
  - Although more than feasible in terms of my available resources and time-frame, this solution is unfeasible in terms of development.
  - This is primarily because it only makes one improvement on the existing system, while removing the features that the client found most useful, such as being able to search for particular questions by type and unit.
  - All questions and additional data would have to be manually entered which would involve a lot of unnecessary repetition of entry for fields such as units and topics. This could also lead to typographic errors.
  - A manual system would be much more time-consuming on a day-to-day basis, wasting lesson time. This is especially important to consider given that these quizzes are often set as starter activities and therefore the process of generating them must be quick and easy.
  - The process of copying and pasting individual questions and answers into the Moodle creator form would be repetitive and a waste of time for the staff; it's a far less complicated process to upload a text file with parsed questions and answers in bulk into the online question bank and let Moodle import and format them automatically.
2. A system which is part-manual and part-computerised. The questions would be entered with the answers and other relevant and stored in the system in a database or text file(s) and the user would be able to use an improved search filter to filter the questions by unit, topic, difficulty rating or type. This would then allow them to copy and paste questions of their choosing into a text file or similar in order to be saved and used as a text-based quiz or (with certain formatting) uploaded to Moodle.
  - This solution is too close to the existing system and doesn't match the needs of the client. It would be feasible both in terms of development and my abilities, but it would merely be an improvement as opposed to a solution.
  - The improvements from the previous solution would be the ability to select repetitive data such as units, topics and rating from drop-down lists rather than having to manually enter them every time and risk typographic errors.
  - In order for copied and pasted questions to be saved in text files by the user and uploaded to Moodle, the user would have to individually go through each

question and add the relevant formatting (GIFT or Moodle XML) and then save it using the correct encoding. Even the simplest of errors such as not closing an XML tag or forgetting a tilde can cause questions to be imported incorrectly, and the entire document to become useless.

- Other drawbacks are the same as with the previous solution; it would still be a time-consuming process to have to manually copy data out of the system, even if the inputs, processing and storage were fully computerised.
3. A fully computerised and part web-based system which would eliminate the need for questions to be stored on Moodle at all. Students would be able to login through the college network and answer any questions the teacher had assigned. Their scores would be stored in the database for the teacher to review and provide e-mail feedback.
- This solution isn't feasible in terms of my time constraints and abilities, and it doesn't meet the user's needs. It would take too long to design and I would need access to the College's username and password data, which compromises network security.
  - The end user wanted questions to be stored on Moodle and saved/printed, and this solution wouldn't facilitate that, meaning it meets few of the objectives for the proposed system, and is vastly over-complicated.
  - The pupils would need computers to be able to answer the questions as a starter activity in the lesson which is impractical, and both students and staff would have to adjust to a new learning environment, which is unnecessary as Moodle facilitates all of the client's initially outlined needs.

Chosen Solution:

4. A fully-computerised system which could potentially meet all of the objectives and remove all of the time-consuming manual processes from the system. Questions would be entered by the user and stored with either in a database or text file(s), along with other relevant data which would allow them to be filtered and displayed by various parameters. Quizzes could be created and then either saved or exported to Moodle, along with a separate answer sheet (for printed quizzes) or integrated answers (for Moodle.)
  - As long as all the objectives prove to be realistic and achievable, this is the most feasible potential solution. It meets the user's needs as outlined by the objectives for the proposed system and it could be updated as time progresses to deal with changing classes or styles of question.
  - The existing staff shouldn't have a problem using the system skill-wise, which means I can make it as efficient as possible without having to keep the interface at the most basic of complexity levels.
  - This solution should be achievable when taking into account my skill level and the time constraints.
  - The students wouldn't have to adjust to a new system of taking quizzes and so could concentrate more on their work.
  - Data which would be constantly repeated such as module names could, as above, be stored by the system for quick selection to minimise time spent using the system and the possibility of typographic errors.
  - Using forms to control the user's input rather than importing questions from text files means that features such as allowing the user to insert commonly used Unicode characters used in physics such as letters of the Greek alphabet and mathematical symbols can be included with relatively no difficulty.
  - The user wouldn't have to copy and paste anything from the system or system files to other documents, because the exporting of Moodle quizzes and text-based quizzes and mark schemes would be done quickly and automatically.



# Design

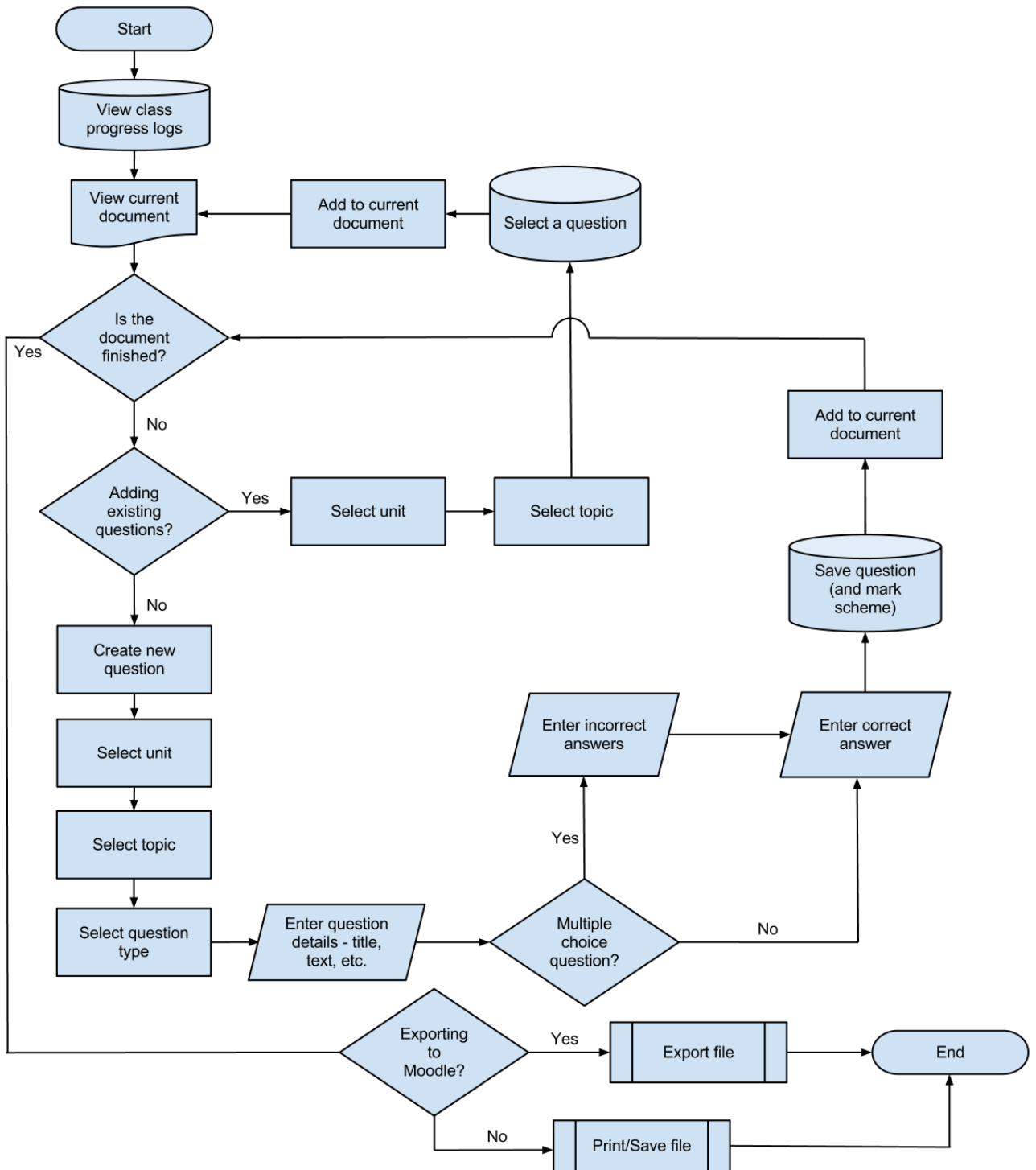
## IOPS Chart

This chart outlines what happens to the data in the new system at the most basic level, in terms of input/output, processing, and storage.

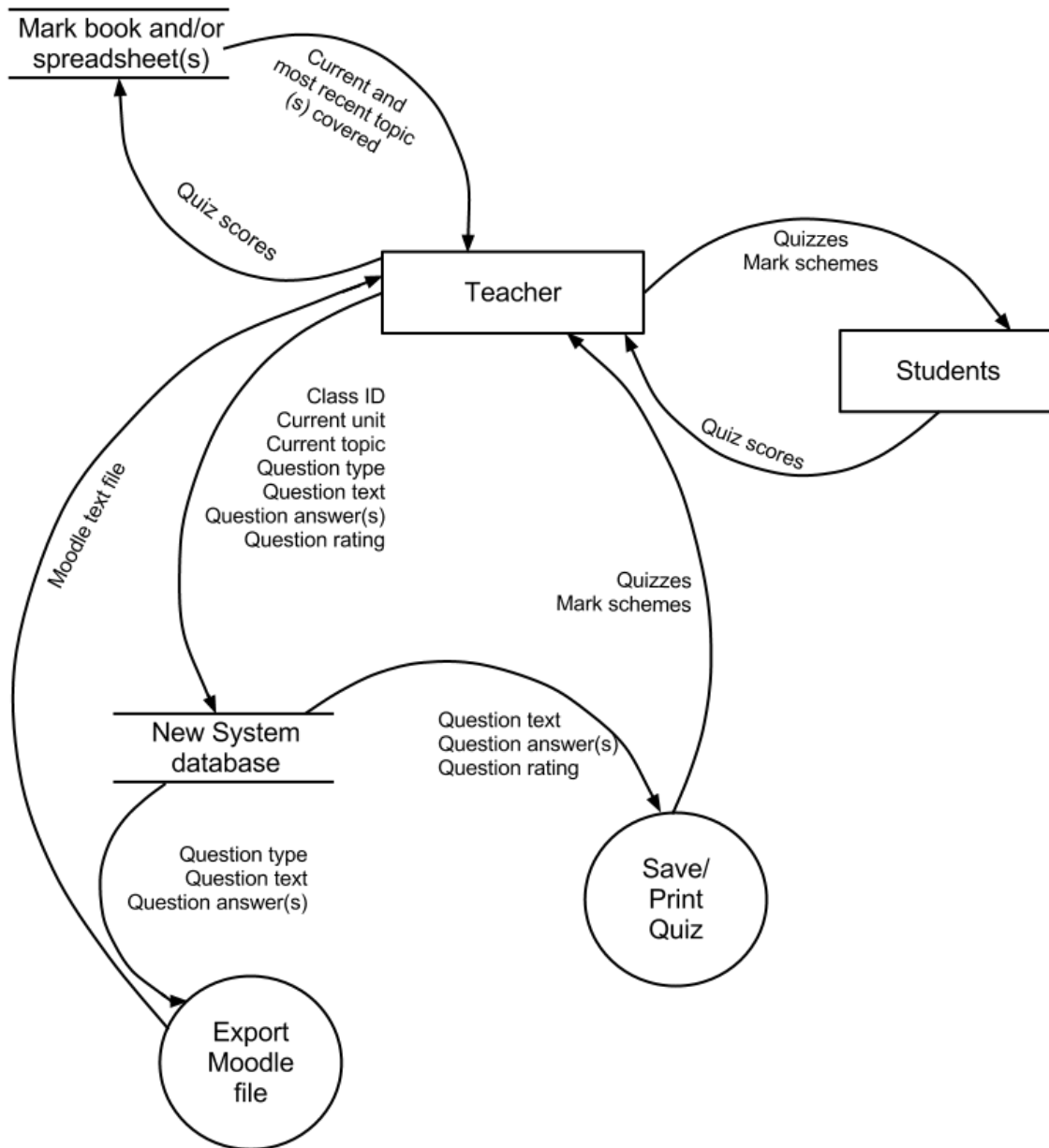
<p style="text-align: center;"><i>Input</i></p> <ul style="list-style-type: none"> <li>• Class ID</li> <li>• Class name</li> <li>• Class record (recent topics)</li> <li>• Unit of work</li> <li>• Topics</li> <li>• Question type</li> <li>• Question text</li> <li>• Correct answer</li> <li>• Incorrect answers (if multiple choice)</li> <li>• Saved quiz name (when loading)</li> <li>• File name (when saving)</li> </ul>	<p style="text-align: center;"><i>Process</i></p> <ul style="list-style-type: none"> <li>• Retrieve and display questions from database/list</li> <li>• Save questions to database/list</li> <li>• Save question documents</li> <li>• Load question documents</li> <li>• Convert questions in Moodle .gift format</li> <li>• Filter questions by unit, topic etc.</li> <li>• Print questions</li> </ul>
<p style="text-align: center;"><i>Storage</i></p> <ul style="list-style-type: none"> <li>• Question database or list</li> <li>• Class database or list</li> </ul>	<p style="text-align: center;"><i>Output</i></p> <ul style="list-style-type: none"> <li>• Moodle GIFT file</li> <li>• Digital question document</li> <li>• Printed question document</li> </ul>

## System Flowchart

The following chart is an update of the current system flowchart, outlining how the new system should operate. It looks to be more complex than the previous version, but most of the processes involved will be automated or at least easier to facilitate than the manual and time-consuming processes in the old system.



### Data Flow Diagram



The main difference between the data flow in the new and old systems is that the two previously separate flows (teacher-students-Moodle) and (teacher-students-Exampro) have been combined into one system. The current topics each class have can also be stored in the system for convenience. The interactions and flow of data between the teachers and students remain the same, because Anne and the other teachers felt this was a part of the system that didn't need to change.

## Data Dictionary and Validation

The following table contains all the data that could be entered into the system by the user, and how I plan to validate it to ensure that all fields are correctly inputted and stored. The erroneous data for most fields is a blank field, so in cases where the validation check is only allowing the user to select correct fields from a list or directory, there is no way to test this.

Field	Data Type	Length	Validation Check	Validation Description	Valid Data	Erroneous Data
Saved Quiz name (when loading)	String	1-30 characters	Lookup	Allows the user to select from only existing files.	(An existing file)	(A file which does not exist)
File name (when saving)	String	1-30 characters	Presence, length	File must have a name, and it must be 30 characters or less.	Refraction-Questions	(Blank)
Class Name	String	2 characters	List (when selecting) Length (when adding)	Allows the selection of an existing class or a class name of the correct length.	A1 (or an existing class)	A2345 (or class which does not exist)
Question text	String	1-500 characters	Presence	Question must exist.	What is mass measured in?	(Blank)
Question type	String	1-15 characters	List	Allows the user to choose one of the four existing types.	Multiple choice	(Blank or type which does not exist)
Correct answer	String	1-100 characters	Presence	Correct answer must exist.	10N	(Blank)

Field	Data Type	Length	Validation Check	Validation Description	Valid Data	Erroneous Data
Incorrect answer (multiple choice only)	String	1-100 characters	Presence	Incorrect answer must exist.	20N	(Blank)
Unit	Integer	5 characters	List	Allows selection from a list of all units.	PHYA1	(Blank)
General topic	String	1-50 characters	List	Allows selection from a list of all topics.	Physics of the ear	(Blank)
Specific topic	String	10-100 characters	List	Allows selection from a list of all topics.	Diffraction gratings	(Blank)
Rating	Integer	1 character	List	Allows selection from the integers 1-5.	4	(Blank)
Quiz title	String	1-30 characters	Presence	Quiz title must exist.	Stationary waves questions	(Blank)
Class record (most recent topics)	String	1-500 characters	Presence	Class record must exist.	Covered diffraction gratings on 10/9.	(Blank)

## Justification of Included Question types

Below are all the types of question that Moodle can recognise (as this will limit the types that the system can process) and my justification for including or not including them in the design of the new system. I've chosen them at an early stage because the layouts of the question types need to be factored into some of the early design processes.

Type	To be included?	Reasoning
Multiple choice	Yes	They're a common format of exam question, and aren't too time consuming for starter questions.
True/False	No	Although not a type of exam question, these can be answered quickly and can be used as 'filler' questions.
Short Answer	Yes	They're one of the most common types of exam question and can be applied to every topic.
Matching	No	Matching questions can't be applied to every topic, and are a very rare exam format.
Missing word	No	A large amount of text has to be entered and stored, which is time consuming and less efficient than other question types. The format also isn't used in exam questions at all.
Numerical	Yes	Calculation questions are another common exam format, and feature in every topic.
Description	No	If uploaded to Moodle, these cannot be automatically marked.
Essay	No	If uploaded to Moodle, these cannot be automatically marked.

## Stepwise Refinement

This is a more detailed list of the processes involved in the new system, mirroring the system flow chart. It allows me to break down each task into its simplest processes, and organise the data that has been inputted to store.

1. Check the current topic the class is covering
  - 1.1. Open the class records
  - 1.2. Choose the class to display
    - 1.2.1. Select class ID from the list
  - 1.3. Check the most recent topic
2. **If SELECTING EXISTING question(s)**
  - 2.1. Filter questions
    - 2.1.1. Search by unit, topic, difficulty rating or class
      - 2.1.1.1. Select options from given dialogue boxes
  - 2.2. Select questions to add
    - 2.2.1. Choose question
    - 2.2.2. (If question is already in the quiz, a warning is displayed)
2. **If ADDING NEW question(s)**
  - 2.1. Choose 'Create'
  - 2.2. Add question type
    - 2.2.1. Select question type from the list
  - 2.3. Enter the body text of the question
  - 2.4. **If MULTIPLE CHOICE**
    - 2.4.1. Enter first, second and third incorrect answers
    - 2.4.2. Enter correct answer
    - 2.4.3. Enter feedback
    - 2.4.4. Enter difficulty rating
  - 2.4. **If TRUE/FALSE**
    - 2.4.1. Indicate the correct answer
      - 2.4.1.1. Tick 'True' or 'False'
    - 2.4.2. Enter feedback
    - 2.4.3. Enter difficulty rating
  - 2.4. **If SHORT ANSWER or NUMERICAL**
    - 2.4.1. Enter correct answer
    - 2.4.2. Enter feedback
    - 2.4.3. Enter difficulty rating
3. Choose whether to export a text-based quiz or a Moodle quiz
  - 3.4. Enter file name
  - 3.5. Save file
4. Update class logs

## Database Normalisation and Design

If I choose to base the new system around a database, this needs to be normalised to avoid inconsistencies from the duplication of data, and to save space by eliminating non-atomic data.

**Bold** indicates a primary (or in the case where more than one attribute in the same table is bolded, composite) key.

*Italics* indicate a foreign key, meaning that the attribute is the primary or partial-composite key in a different table.

This is the first table, before any attempt at normalisation.

tblquestion

<b>QID</b>	QText	Unit	QType	QAns	IncAns	Rating	GTopic	STopic	Class	Quiz

The table contains non-atomic data – attributes with repeated entries; in this case, IncAns, Class and Quiz – which are not only a waste of space but can also lead to users not being able to store all the relevant data. These need to be moved to new tables.



These are the tables in First Normal Form.

tblquestion

<b>QID</b>	QText	Unit	QType	QAns	Rating	GTopic	STopic

tblclassquestion

<b>QID</b>	Class

tblincorrectanswer

<b>QID</b>	IncAns

tblquiz

<b>QUIZID</b>	QID

In 1NF, none of the data is non-atomic. All three of the new tables require composite keys, because they both represent many-to-many relationships. You couldn't tell which class had been assigned a question just from the question ID, and you couldn't tell which question a class had been assigned just from the class name. Similarly with the table of incorrect answers, you couldn't tell which question an answer related to just from the answer itself, and you couldn't tell which incorrect answer was being stored just from the question ID. The same is true for quizzes and the questions they contain.

In the tblquestion, there are attributes that don't depend on the primary key. QUnit is dependant both on GTopic and STopic (unit and general topic have a one-to-many relationship, as do general topic and specific topic.) By knowing STopic, GTopic and Unit are automatically known, so in the questions table, only STopic needs to be stored. GTopic and Unit can be stored in separate tables because they are functionally dependant on STopic, and not QID.

These are the tables in Second Normal Form. As there are no functional dependencies existing between attributes that couldn't be used as alternatives to primary keys, the tables are also in 3NF at this point.

tblquestion

<b>QID</b>	QText	<i>S</i> Topic	QAns	Rating	QType

tblclassquestion

<b>QID</b>	Class

tblincorrectanswer

<b>QID</b>	IncAns

tblunittopic

<b>G</b> Topic	Unit

tbltopics

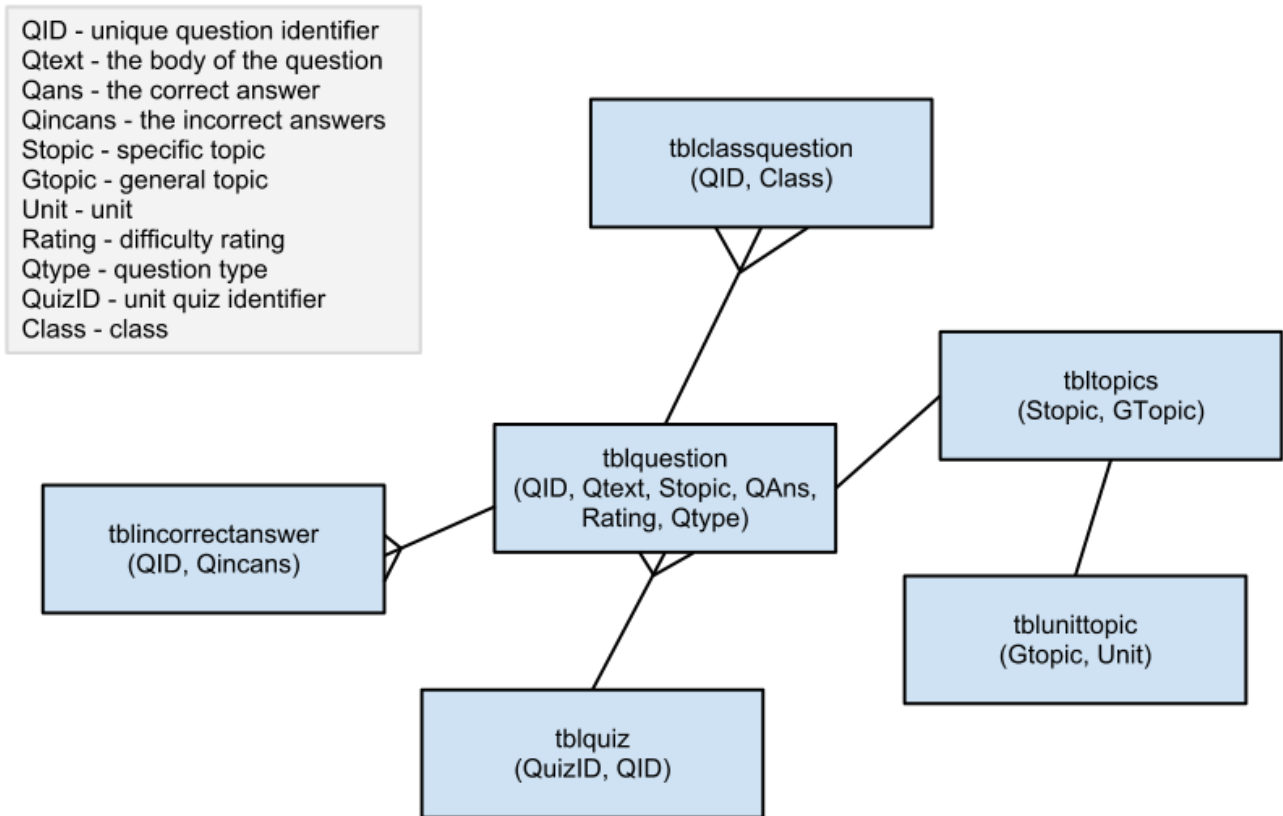
<b>S</b> Topic	<i>G</i> Topic

tblquiz

<b>QUIZID</b>	<b>QID</b>

### Table Relationship Diagram

This diagram illustrates the relationships between the normalised tables in the new system. The entity abbreviations are explained in the key. We can tell the data has been normalised because there are no many-to-many relationships.



## Sample of Possible SQL Queries

**Displaying all existing questions in the database**

```
SELECT Qtext,Qtype,Rating FROM tblquestion
```

**Displaying all questions from a topic**

```
SELECT Qtext,Qtype,Rating  
FROM tblquestion  
WHERE Topic=?
```

**Displaying all questions of particular rating**

```
SELECT Qtext,Qtype,Rating  
FROM tblquestion  
WHERE Rating=?
```

**Displaying all questions of particular type**

```
SELECT Qtext,Qtype,Rating  
FROM tblquestion  
WHERE Type=?
```

**Adding a new question to the database**

```
INSERT INTO tblquestion(QID, Qtext, STopic, QAns, Rating, QType)  
VALUES(?, ?, ?, ?, ?, ?)
```

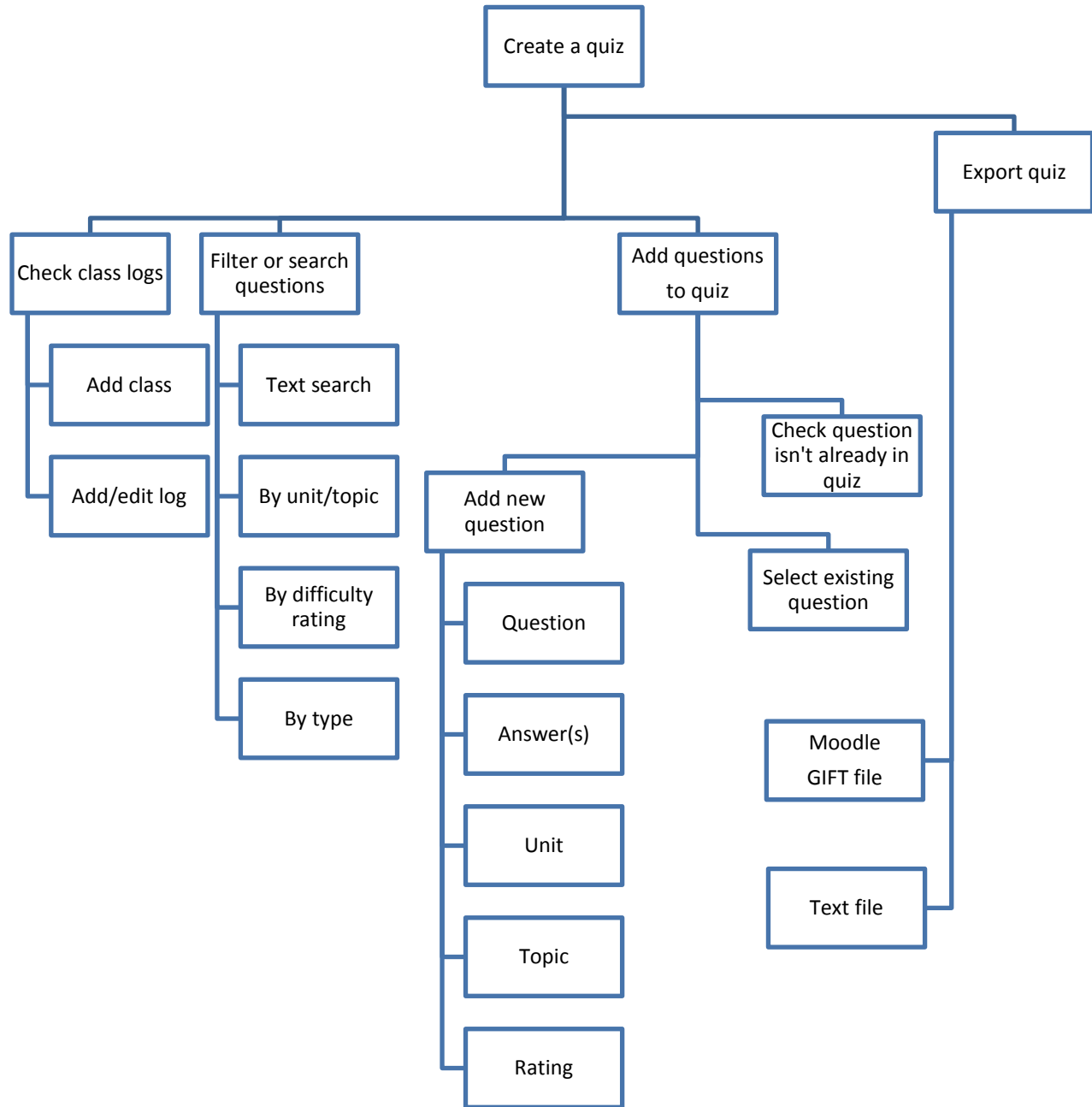
**Updating an existing question**

```
UPDATE tblquestion  
SET (changes)  
WHERE QID=?
```

**Deleting a question**

```
DELETE FROM tblquestion  
WHERE QID=?
```

### Top-Down Design



## OOP Class Design

These are the classes my system would use if I wrote in the OOP paradigm, modelling questions and groups as objects. Each question or class would be created at runtime when declared by the user. Quizzes could be modelled as objects that contain questions, but my software doesn't need to store quizzes past run-time, so for the purpose of exporting, quizzes would just be modelled as 2-dimensional arrays of question data.

### Class: Question

QID as integer (Question identifier)  
QText as string (The body of the question)  
QAns as string (The correct answer)  
QIncorrect(2) as string (Incorrect answers)  
QRating as integer (Difficulty rating)  
QTopic as string (Topic)  
QUnit as string (Unit)  
QType as string (Type)

### Class: Group

Name as string (Group name)  
Log as string (Record of current topic)

## Definition of Record Structure

I've chosen the Gift file format for converted Moodle questions because of the three supported types – the other two of which are Moodle XML and Moodle XHTML – it has the simplest syntax which should reduce errors when parsing text and also minimise file sizes. It also makes the converted questions more readable because there are no html tags or indented line breaks. Gift files have to be exported in UTF-8 variable-width encoding, but this is the default encoding for VB.NET's StreamWriter and therefore won't have to be changed.

Quizzes and mark schemes will be saved as .txt files – as they are only used once, formatting is not of great concern.

Because the data stored about each question is minimal – evidenced by the number of two column tables in the normalised database planning – and won't have to be changed or updated (either for Moodle or for printing and manually marking), I've decided to store the question list in a text file instead of a database. This means that time will be saved, because search algorithms can be used on the loaded list instead of more time-consuming SQL queries to a database that could be on an externally hosted server rather than the college network.

## Security and Integrity of Data

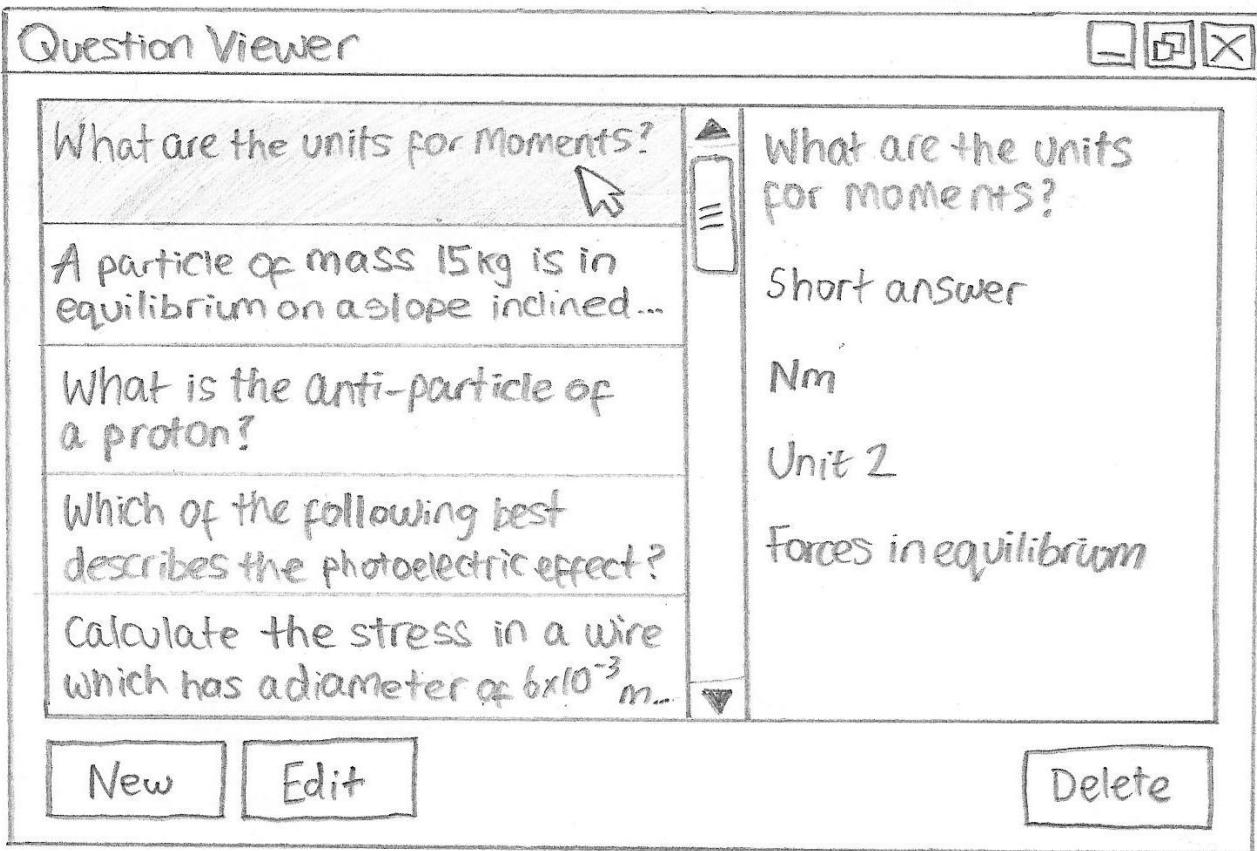
As there is no personal data stored about students or staff within the system, there is no need for any kind of encryption. Students won't have access to the system as it would only be installed on staff user areas, so having a restricted access profile is similarly not a concern. Storing the question and class data in text files rather than an externally hosted database means there is less chance a student could gain access to any answers.

To protect the integrity of the stored data, all data entry will be controlled by strict validation rules. Wherever possible – selecting question types, true/false answers, or filtering questions by certain parameters – the user will select their options from drop-down menus, radio buttons or tree diagrams. This minimises free text input which, as well as saving time, also stops typographic errors which may cause the system to crash or incorrect data to be stored.

## User Interface Design

Initial Drawings:

These drawings are my first outlines of what the interface of the system should look like. They will most likely not be the final designs, but they are the first generation of each form and a starting point for later concepts.



The 'Question Viewer' is the pane in which all existing questions in the system can be displayed. From within this pane, questions can be added, edited, or deleted. When search queries are specified, the question list will be filtered to ensure that the user only sees the relevant questions. Selecting a question on the left of the panel displays its unit, topic, type and answer(s) on the right of the pane.



This is the 'Add/Edit Question' pane. The user inputs all the data for a question including the question itself, unit and topic, and then the type. When the radio button for any type is clicked, the bottom of the pane will change to display the relevant answer inputs. Short answer and numerical answer are identical, with a text field for entry and a drop-down menu to select difficulty rating. True/False questions have another drop-down menu instead of a text field, and multiple choice questions have an additional four incorrect answer boxes.

Quizzes

What are the SI units for mass?

What is g to 2sf?

What are the SI units for length?

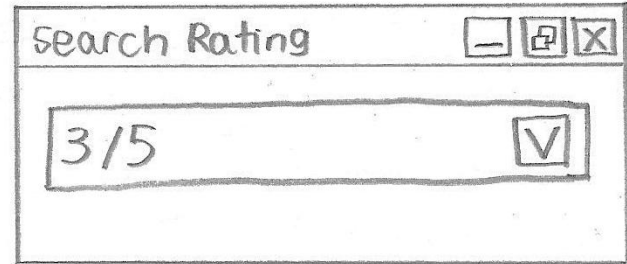
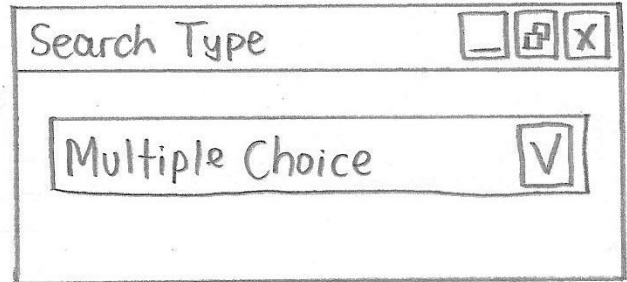
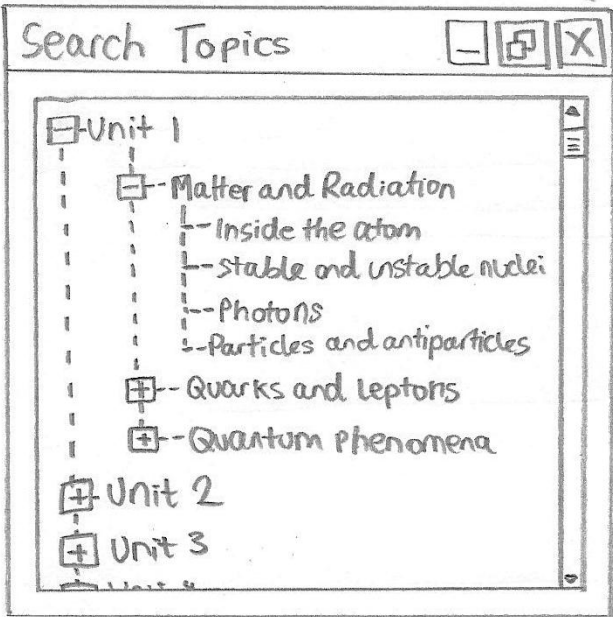
Which of the following does the constant 'c' represent?

True or false? Light years are a measure of distance.

save Export Print Delete question

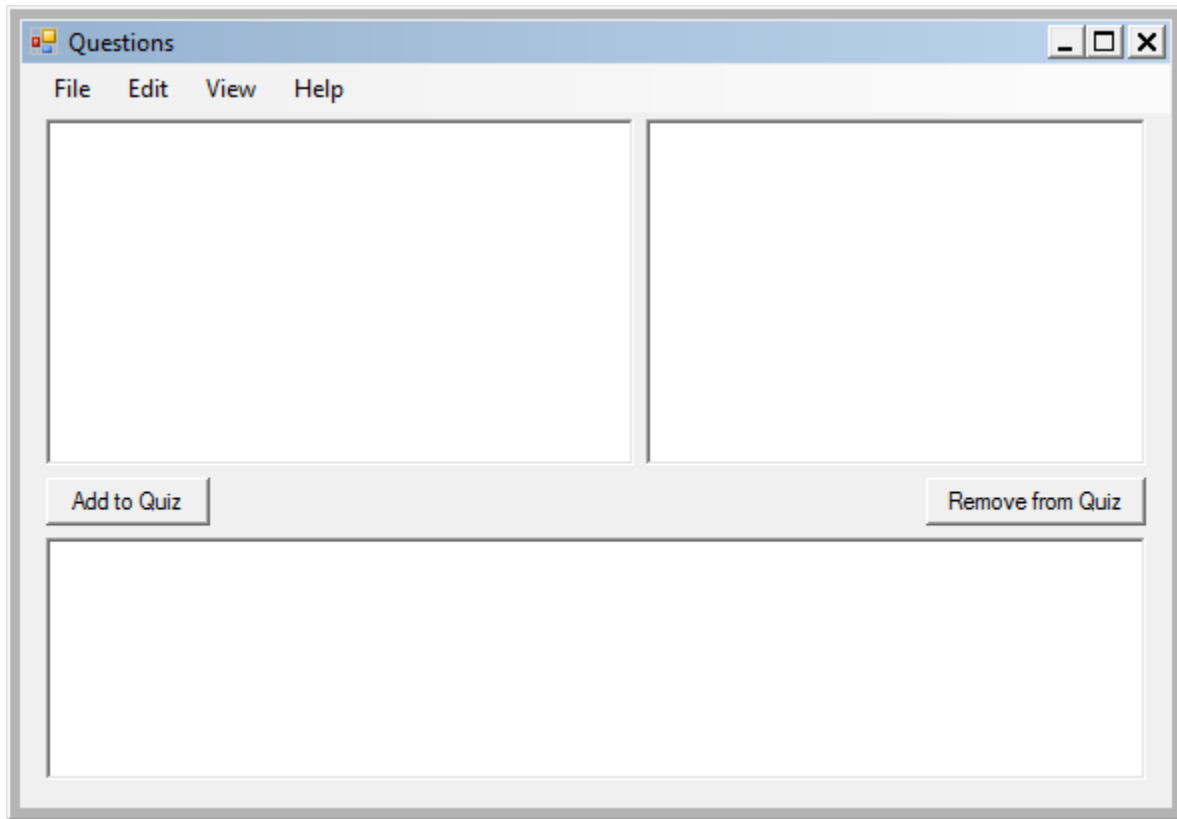
[QUESTION VIEWER PANE]

The current quiz being edited is displayed at the top of this form, along with a question viewer pane identical to the one above, below it. Questions can be selected from the viewer and added to the quiz (and from there, deleted if necessary.) The quiz can then be saved, exported or printed.

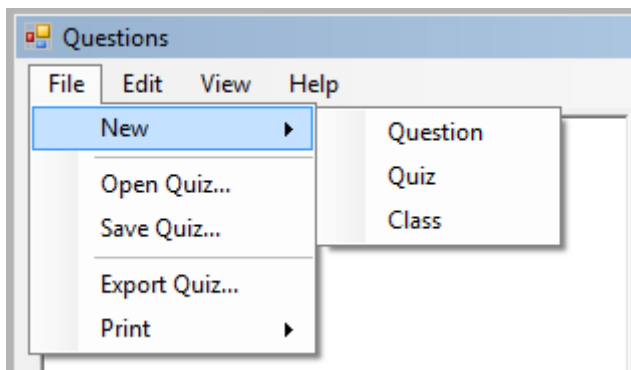


These search windows allow the user to filter the questions which are displayed to them by unit and topic, type, or difficulty rating. The unit and topic search is displayed as a tree, because presenting each unit as a parent node with broader topics as collapsible child nodes ensures that the user isn't overwhelmed with lists of text and can easily find the topic they want.

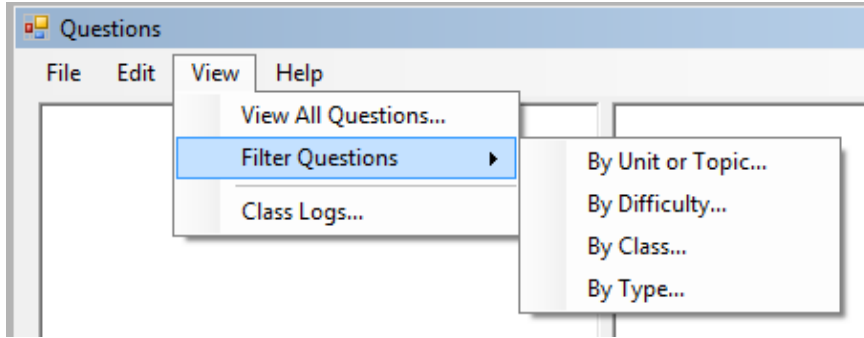
Visual Studio Form Design:



The main window comprises a question viewer pane at the top with a list of questions on the left and the answers and question information on the right. At the bottom is the quiz pane which displays all of the questions in the current quiz and allows the user to quickly add more or remove them.

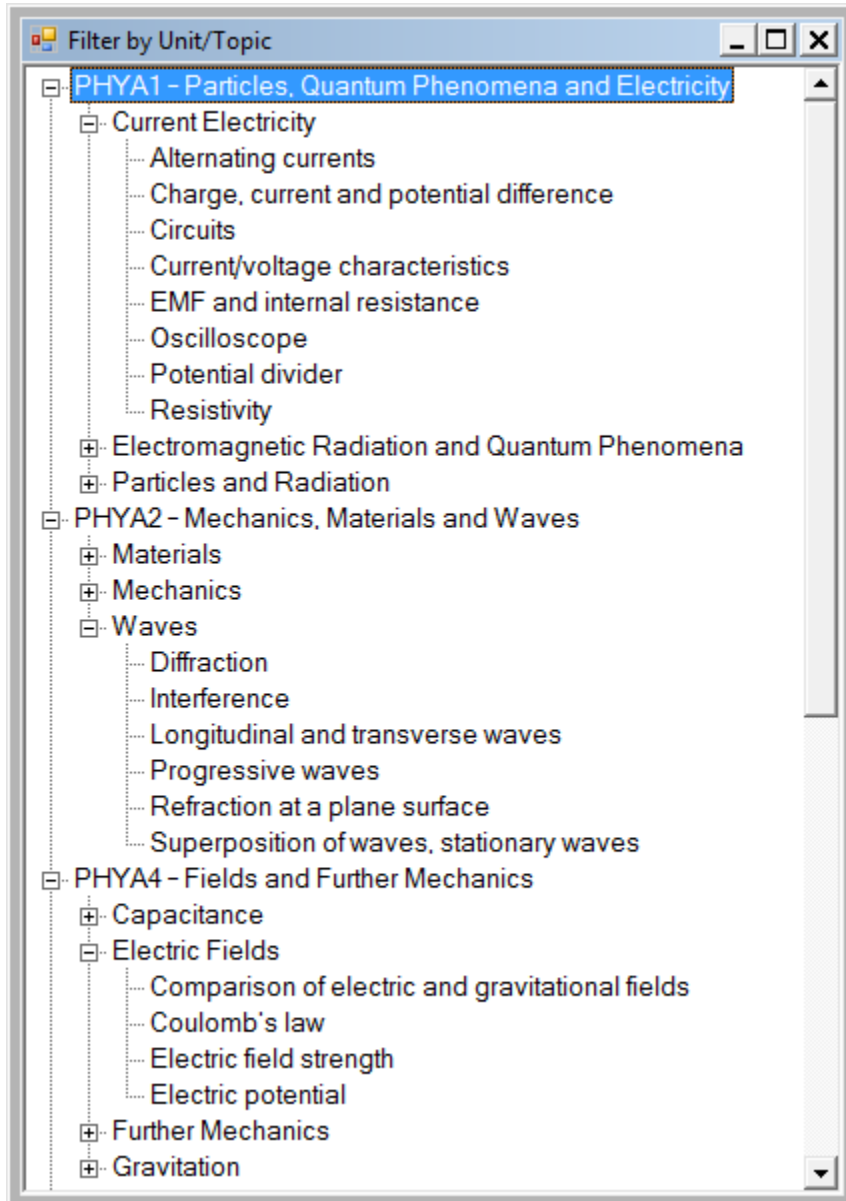


The toolbar is based on the standard Windows toolbar for familiarity of use, and menu options are grouped by type.

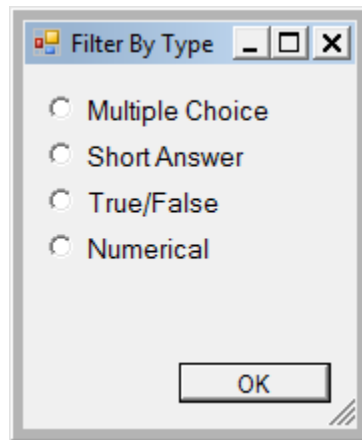
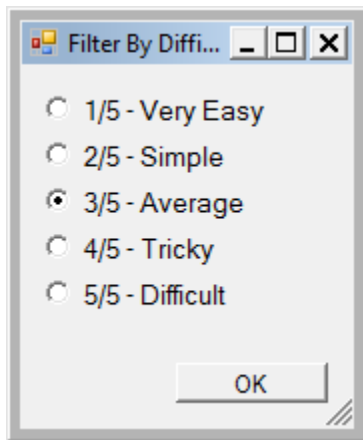


'View All Questions' would reset any search filters.

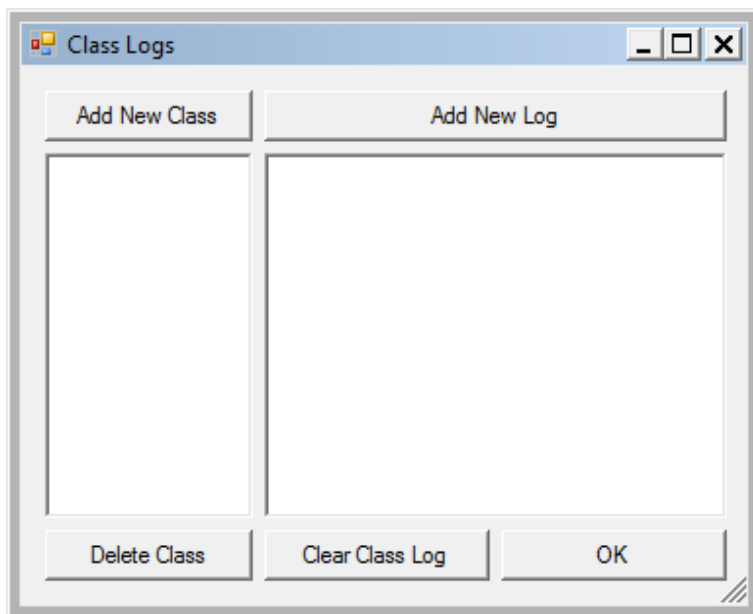
'Filter Questions' opens any of the filter dialogue boxes to enable the user to specify the filter terms.



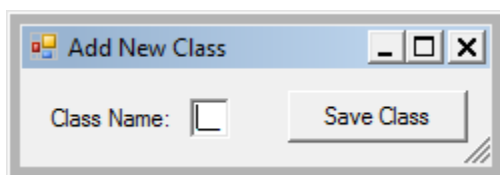
To filter by unit, general topic or specific topic the user would double-click the selected node in the tree. The first levels of nodes are units, the second are general topics and the third are specific topics. Parent nodes can be expanded or collapsed, a feature which improves navigation.



The 'Filter By Class' Dialogue would allow the user to view a list of all classes stored in the database and double-click any class to select. 'Filter By Difficulty' and 'Filter By Type' use radio buttons because the options available never change. The 'Class Log' form would show a list of all classes in the left pane, and the recent logs for the selected class in the right pane. From here, new logs or classes can be added.



Choosing 'Add New Class' opens the second class dialogue, where a new class with the name format used by the college – e.g. C1 – can be saved.



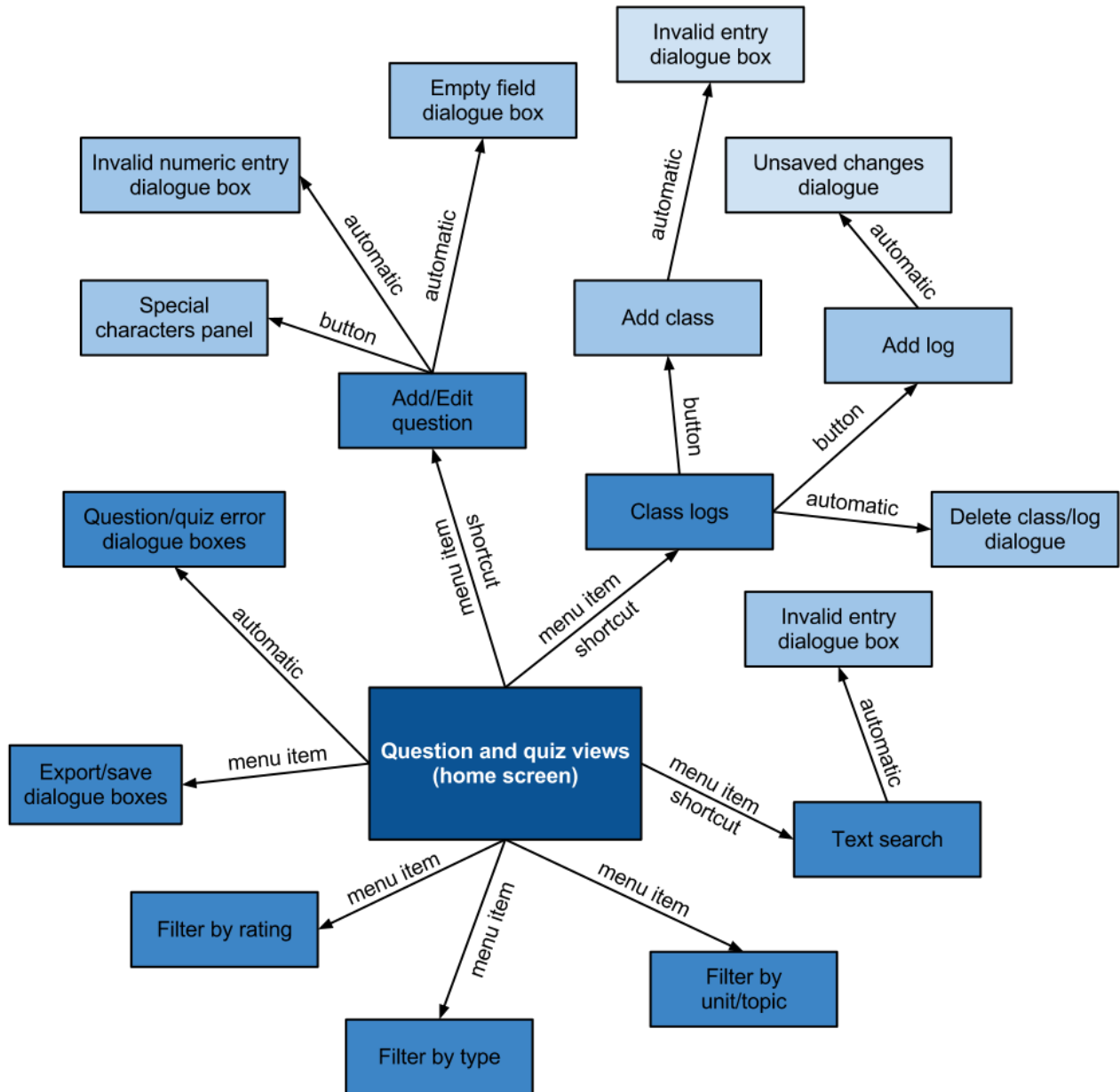
The image shows the 'Add Question' form on the left and three examples of generated quiz questions on the right. Blue arrows indicate the flow of data from the form to the generated questions.

- Add Question Form:**
  - Question: [Text input]
  - Unit: [Dropdown menu]
  - General topic: [Dropdown menu]
  - Specific topic: [Dropdown menu]
  - Type:
    - Short Answer
    - Multiple Choice
    - True/False
    - Calculation
  - Answer: [Text input]
  - Rating: [Dropdown menu]
  - Save Question [Button]
- Generated Question 1:**
  - Answer: [Dropdown menu with 'True' selected]
  - Rating: [Dropdown menu]
  - Save Question [Button]
- Generated Question 2:**
  - Answer: [Text input]
  - Rating: [Dropdown menu]
  - Save Question [Button]
- Generated Question 3:**
  - Answer: [Text input]
  - Wrong: [Text input]
  - Wrong: [Text input]
  - Wrong: [Text input]
  - Rating: [Dropdown menu]
  - Save Question [Button]

The 'Add Question' form collects all the user-inputted data about new questions added to the database. Wherever possible (unit, topics, type, rating) the user has to select the data from existing list menus or radio buttons to minimise input errors and time-consuming form completion.

## Form Navigation Design

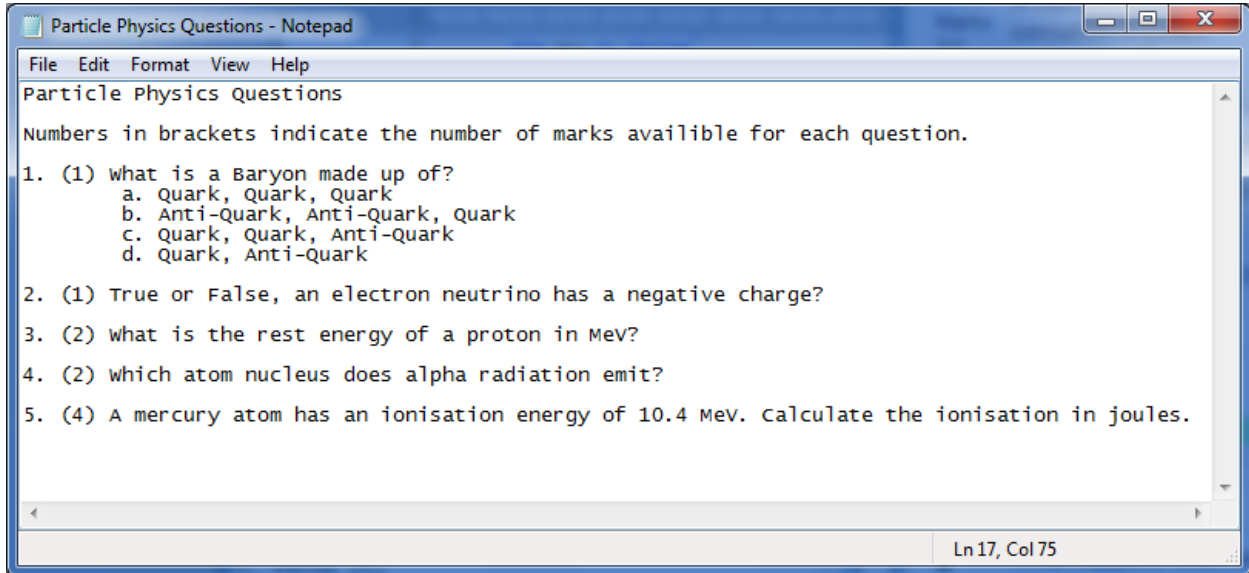
The following chart illustrates how users would navigate between all of the forms and dialogues in the new system, starting from the question/quiz viewer which is the home screen.





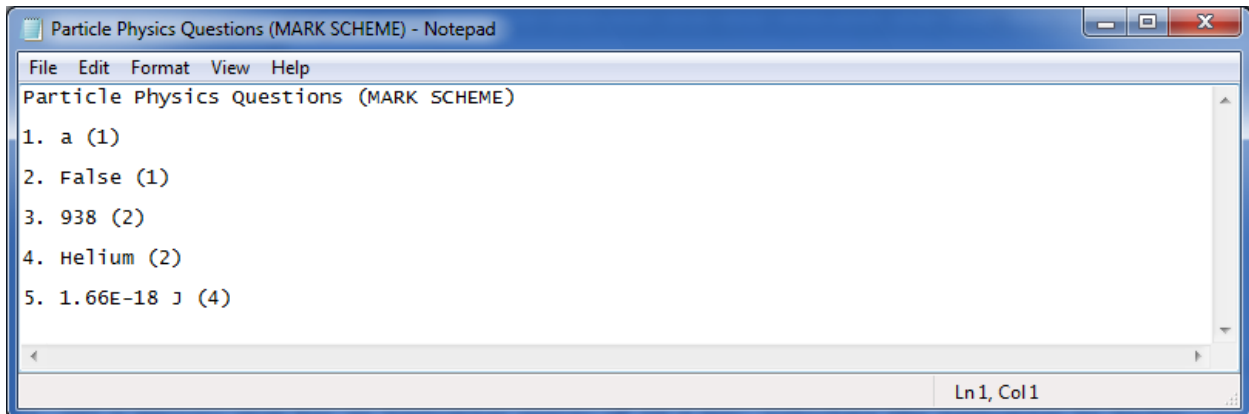
## Output Design

These text files are rough plans of the files the system should export. The first are the ‘hard copy’ files, i.e. those that won’t be uploaded to Moodle but will instead be saved or printed.



```

Particle Physics Questions
Numbers in brackets indicate the number of marks available for each question.
1. (1) what is a Baryon made up of?
    a. Quark, Quark, Quark
    b. Anti-Quark, Anti-Quark, Quark
    c. Quark, Quark, Anti-Quark
    d. Quark, Anti-Quark
2. (1) True or False, an electron neutrino has a negative charge?
3. (2) what is the rest energy of a proton in MeV?
4. (2) which atom nucleus does alpha radiation emit?
5. (4) A mercury atom has an ionisation energy of 10.4 MeV. Calculate the ionisation in joules.
    
```



```

Particle Physics Questions (MARK SCHEME)
1. a (1)
2. False (1)
3. 938 (2)
4. Helium (2)
5. 1.66E-18 J (4)
    
```

The next screenshots show the exported Gift files and how they would look when exported and uploaded to Moodle. There is no separate mark scheme file for Moodle, because it is integrated into the question file.

```

Particle Physics Moodle GIFT - Notepad
File Edit Format View Help
what is a Baryon made up of? {
=Quark, Quark, Quark
~Anti-Quark, Anti-Quark, Quark
~Quark, Quark, Anti-Quark
~Quark, Anti-Quark
}

True or False, an electron neutrino has a negative charge? {F}

what is the rest energy of a proton in MeV? {#938}

which atom nucleus does alpha radiation emit? {=Helium}

A mercury atom has an ionisation energy of 10.4 MeV.
Calculate the ionisation in joules. {#1.66*10^-18}
Ln 19, Col 51
    
```

1 What is a Baryon made up of?

Marks: --/1

Choose one answer.

- a. Quark, Quark, Quark
- b. Quark, Quark, Anti-Quark
- c. Anti-Quark, Anti-Quark, Quark
- d. Quark, Anti-Quark

Submit

2 A mercury atom has an ionisation energy of 10.4 MeV. Calculate the ionisation in joules.

Marks: --/1

Answer:

Submit

3 What is the rest energy of a proton in MeV?

Marks: --/1

Answer:

Submit

4 Which atom nucleus does alpha radiation emit?

Marks: --/1

Answer:

Submit

5 True or False, an electron neutrino has a negative charge?

Marks: --/1

Answer:

- True
- False

Submit

## Algorithm Design

### Random Ordering of Multiple Choice Answers

<b>Explanation</b>
Every multiple choice question which is either exported or printed needs to have a different order of answers, i.e. the second option shown cannot always be the correct answer. This algorithm randomises the indexes of the array of four answers (three incorrect and one correct).
<b>Pseudo-code</b>
<pre> R ← Random(1, 4) FOR EACH Question in newQuiz where Type ← "MC"   Var incorrectAnswer[2]   Var answer   X ← R.Next   IF x ← 1 Then Output answer, incorrectAnswer[0],                     incorrectAnswer[1], incorrectAnswer[2]   ELSEIF x ← 2 Then Output incorrectAnswer[0], answer,                           incorrectAnswer[1], incorrectAnswer[2]   ELSEIF x ← 3 Then Output incorrectAnswer[0],                           incorrectAnswer[1], answer,                           incorrectAnswer[2]   ELSE: Output incorrectAnswer[0], incorrectAnswer[1],               incorrectAnswer[2], answer   ENDIF NEXT </pre>

### Question Text Search

<b>Explanation</b>
As well as being able to filter the questions in the viewer, users should be able to search for a string and have all questions which contain that string returned to them. A similar algorithm is used for the filtering.
<b>Pseudo-code</b>
<pre> Var found ← False Var searchTerm FOR i ← 0 to len[allQuestions] - 1   IF question contains searchTerm THEN     Found ← True     Output question   ENDIF NEXT IF Found ← False THEN Output "No questions found." </pre>

### Validating Question Entry

**Explanation**

Question entry and editing needs to be controlled by strict validation rules in order for questions to be filtered, and saved in a way which Moodle can mark. This validation would be done before any question is saved.

This function checks for the presence of a question, answer, unit, topic, question type and rating. If the question is a multiple choice one, it also checks for the presence of incorrect answers, and if the question is numerical, it checks it is an integer and doesn't contain characters which would cause an exception.

**Pseudo-code**

```
DO
    Var validQuestion ← True
    IF question ← "" or unit ← "" or topic ← "" or answer ← ""
        or type = "" or rating ← "" then validQuestion ← False
    ELSEIF type ← "MC" and incorrectAnswer[0] ← "" or
        incorrectAnswer[2] ← "" or incorrectAnswer[3] ←
            "" then validQuestion ← False
    ELSEIF type ← "NU"
        TRY INT(answer)
        CATCH validQuestion ← False
        END TRY
    ENDIF
LOOP UNTIL validQuestion = True
```

## Importing Questions from a Text File

### Explanation

This function reads all the questions in a collection in the system. For each question, if the question body is stored on the nth line, the type will be on the (n+1)th line, the answer on the (n+2)th line and so on. Each question will be saved over nine lines.

### Pseudo-code

```
IF file exists
  Var allQuestions[]
  Var allLines[] ← file
  Var count ← 0
  FOR i ← 0 to LEN(file) - 8 STEP 9
    count ← count + 1
    question.QID ← count
    question.text ← allLines[i]
    question.type ← allLines[i+1]
    question.answer ← allLines[i+2]
    IF type = "MC" THEN
      question.incorrectAnswer[0] ← allLines[i+3]
      question.incorrectAnswer[1] ← allLines[i+4]
      question.incorrectAnswer[2] ← allLines[i+5]
    END IF
    question.unit ← allLines[i+6]
    question.topic ← allLines[i+7]
    question.rating ← allLines[i+8]
    allQuestions.add(question)
  NEXT
END IF
```

## Checking for Repeated Questions

### Explanation

The system has to be able to notify the user if they are adding a duplicate question to the quiz so it isn't accidentally assigned twice.

### Pseudo-code

```
IF listItemIndex > -1 THEN
  Var question ← allQuestions(listItemIndex)
  If newQuiz CONTAINS (question) THEN
    Output "This question is already in the quiz."
  ELSE: newQuiz.add(question)
END IF
```

## Converting Questions to Moodle's Gift Format

### Multiple Choice

General format	Example
<pre>Question { =CorrectAnswer ~IncorrectAnswer ~IncorrectAnswer ~IncorrectAnswer }</pre>	<pre>What is the SI unit for mass? { =Kilograms ~Newtons ~Metres ~Kelvin }</pre>
<b>Pseudo-code</b>	
<pre>Output question &amp; "{" &amp; newline &amp; "=" &amp; answer &amp; newline &amp; incorrectAnswer[0] &amp; newline &amp; incorrectAnswer[1] &amp; newline &amp; incorrectAnswer[2] &amp; newline &amp; "}" &amp; newline &amp; newline</pre>	

### True/False

General format	Example
<pre>Question {T} or Question {F}</pre>	<pre>Kilograms are the SI unit for mass. {T} Metres are the SI unit for mass. {F}</pre>
<b>Pseudo-code</b>	
<pre>If answer ← "true" then     Output question &amp; "{T}" Else: Output question &amp; "{F}" End if</pre>	

### Short Answer

General format	Example
<pre>Question{=CorrectAnswer}</pre>	<pre>What are kilograms the SI unit for?{=Mass}</pre>
<b>Pseudo-code</b>	
<pre>Output question &amp; "{=" &amp; answer &amp; "}"</pre>	

### Numerical

General format	Example
<pre>Question {#Answer}</pre>	<pre>How many grams in a kilogram? {#1000}</pre>
<b>Pseudo-code</b>	
<pre>Output question &amp; "{#" &amp; answer &amp; "}"</pre>	

## Identification of Storage Media

As the software has been designed for a fairly specific purpose and therefore should not be large, I calculate that the executable install file should be small, taking up less than 1 megabyte of space. When the system is used, the question data, class data and unit/topic data are all written to plain text files. These are saved in the user directory and should not exceed 5 megabytes. These small file sizes mean that I could choose to distribute the system in a number of ways.

The user could download the executable from a secure online location, but the college has a strict firewall which may make this difficult, and also means the user will have to be connected to the internet during the installation process. On a slow connection, this would be impractical. It also means the file would have to remain online and accessible indefinitely in case a reinstall is required.

The system could also be installed from a CD-ROM (CD-RW discs are unnecessary because data will only be read from and not written to the disc after the executable is initially copied) which can store 194 megabytes of data. The disc could then be kept to install the system on other devices. However, as the disc would only contain the installation file, a large amount of space that cannot be overwritten would be wasted. There is also the consideration that as the pressure on laptops and netbooks to become more portable and smaller in dimension increases, many machines will no longer have an internal CD/DVD drive.

My choice for the storage of the executable file would be on a USB flash-drive, for several reasons. The file itself wouldn't create any wasted space on the flash drive, as the rest of the drive could still be read to and written from as usual. Teachers in the departments tend to already own flash drives and could therefore keep backup copies of the executable on these instead of having to store physical CDs, and no internet connection is required during the installation process. As USB ports are an industry standard on laptops and desktops for the foreseeable future, this ensures that even as the hardware in the Physics department at the college is updated, as long as the .NET framework is kept updated, the software can still be installed. One of the machines from which I will be writing the system does not have a CD/DVD drive, and so from a programmer's perspective it is also a great deal easier to store the executable files on a USB flash drive. Installation speeds from USB 2.0 and 3.0 are both faster than installation speeds from a CD-ROM, which is another advantage.

## Testing Plan

To ensure the system will be able to correctly handle all user inputs (both correct and incorrect) as well as correctly navigate between forms and execute algorithms correctly, I have planned several testing methods.

### Input and Output Testing Design

The following table is designed to test the actual outcome vs. the expected outcome for every user input and system output in the case of typical (correct and expected) data, erroneous (would cause the system to throw an exception, e.g. Incorrect data types) data and extreme data (less expected data, e.g. Blank fields or boundary data).

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
1	Adding a question	T: All fields filled in	Question is added to the pane and form closes.		
		E: Blank field/s	'Please enter _____' dialogue. Form stays open.		
2	Editing a question	T: There is a question selected	Edit question form loads with existing data, and when saved, question updates in the question pane and, if applicable, the quiz.		
		E: None selected	'Please select a question' dialogue.		
3	Deleting a question	T: There is a question selected	'Are you sure' dialogue, and if yes then question is deleted from the pane and if applicable, the quiz.		
		E: None selected	'Please select a question' dialogue.		
4	Adding a question to the quiz	T: Question selected	Question appears in the quiz pane.		
		E: No question selected	'Please select a question' dialogue.		
		X: Question is already in quiz	'Already in quiz' dialogue. Question does not appear in the quiz pane.		



Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
5	Removing a question from the quiz	T: Question selected (in the quiz pane)	'Are you sure' dialogue. If yes, question is removed from the quiz pane.		
		E: No question selected	'Please select a question' dialogue.		
6	Filter by difficulty	T: Any radio button selected	Question pane refreshes to show all questions of that difficulty.		
		X: No questions of that difficulty	Question pane shows no questions.		
7	Filter by unit/topic	T: Any node selected	Question pane refreshes to show all questions of that unit/topic.		
		X: No questions of that unit/topic	Question pane shows no questions.		
8	Filter by type	T: Any radio button selected	Question pane refreshes to show all questions of that type.		
		X: No questions of that type	Question pane shows no questions.		
9	View class logs	T: logs may or may not already exist	All class logs are loaded from document (document is created if it doesn't exist,) log field will be blank if there is no saved log.		
10	Add class	T: Class name field is filled in	Class is added to the list on the logs form		
		E: Class name field is blank	'Please provide a class name' dialogue.		
11	Add/edit log	T: There is a class selected	Add/edit form loads (with existing log if there is one.)		
12	Delete Class	T: There is a class selected	'Are you sure' dialogue. If yes, the class is deleted.		
13	Delete Log	T: There is a class selected	'Are you sure' dialogue. If yes, the log is deleted.		

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
14	Save quiz (text-based)	T: The quiz contains questions.	Save dialogue opens and two text files are saved. One quiz and one mark scheme with the same filename as entered for the quiz suffixed by “(mark scheme.)”		
		E: The quiz contains no questions.	‘Quiz contains no questions’ dialogue. Nothing is exported.		
15	Export quiz (for Moodle)	T: The quiz contains questions.	Save dialogue opens and one text file with integrated answers is saved. This can be uploaded to Moodle with no editing.		
		E: The quiz contains no questions.	‘Quiz contains no questions’ dialogue. Nothing is exported.		
16	Loading questions	T: Question files exist and contain question data	All questions are loaded into the question pane.		
		T: Question files do not exist	‘Welcome to Benchmark’ dialogue. (Files not existing suggest this is the first time the system has been run.)		
		T: Question files exist but do not contain question data	‘You haven’t added any questions yet’ dialogue. (Blank files suggest the system has been run before.)		

## Navigation Testing Design

To be able to see at a glance which forms are linked to each other and which can only navigate through other forms, I've created a table. It is unidirectional (being able to navigate from form A to B for example does not mean that the user can directly navigate from form B back to A, it may only be able to return focus to A upon closing.)

The table can simply be filled in with ticks or crosses to test the navigation works as expected. Until implementation is complete, I cannot know for sure how many forms there will be in the system so the following table with forms A-H is merely an estimate. The finished system could have more or less, but not drastically so.

<b>Navigating from:</b> →	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>
<b>Navigating to:</b> ↓								
<b>A</b>								
<b>B</b>								
<b>C</b>								
<b>D</b>								
<b>E</b>								
<b>F</b>								
<b>G</b>								
<b>H</b>								

# Testing

---

## Input and Output Testing

This table is an overview of the basic input and output tests conducted on the system in order to make sure the expected outcome of every action is the actual outcome when using the system. Pale blue rows indicate tests that were not featured in the original plan, for features which were added during implementation.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
1	Adding a question	T: All fields filled in	Question is added to the pane and form closes.	As expected. <sup>1</sup>	None required.
		E: Blank field/s	'Please enter _____' dialogue. Form stays open.	As expected. <sup>2</sup>	None required.
		E: Invalid character in number field		As expected. <sup>3</sup>	
2	Editing a question	T: There is a question selected	Edit question form loads with existing data, and when saved, question updates in the question pane and, if applicable, the quiz.	As expected. <sup>4</sup>	None required.
		E: None selected	'Please select a question' dialogue.	As expected. <sup>5</sup>	None required.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
3	Deleting a question	T: There is a question selected	'Are you sure' dialogue, and if yes then question is deleted from the pane and if applicable, the quiz.	As expected. <sup>6</sup>	None required.
		E: None selected	'Please select a question' dialogue.	As expected. <sup>7</sup>	None required.
4	Adding a question to the quiz	T: Question selected	Question appears in the quiz pane.	As expected. <sup>8</sup>	None required.
		E: No question selected	'Please select a question' dialogue.	As expected. <sup>9</sup>	None required.
		X: Question is already in quiz	'Already in quiz' dialogue. Question does not appear in the quiz pane.	'Already in quiz' appears but question is still added to the quiz. <sup>10</sup>	Exit the IF statement before the question is added. Second test conducted (4b.)
4(b)	Adding a question to the quiz (REPEATED TEST)	X: Question is already in quiz	'Already in quiz' dialogue. Question does not appear in the quiz pane.	As expected. <sup>11</sup>	None required.
5	Removing a question from the quiz	T: Question selected (in the quiz pane)	'Are you sure' dialogue. If yes, question is removed from the quiz pane.	As expected. <sup>12</sup>	None required.
		E: No question selected	'Please select a question' dialogue.	As expected. <sup>13</sup>	None required.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
6	Text search	T: Question(s) contain the search term	Question pane refreshes to show all questions that contain the term.	As expected. <sup>14</sup>	None required.
		E: Blank search field	'Please enter a search term' dialogue.	As expected. <sup>15</sup>	None required.
		X: Valid search field but no questions contain the search term	'No question found' dialogue. Question pane shows no questions.	As expected. <sup>16</sup>	None required.
7	Filter by difficulty	T: Any radio button selected	Question pane refreshes to show all questions of that difficulty.	As expected. <sup>17</sup>	None required.
		X: No questions of that difficulty	Question pane shows no questions.	As expected. <sup>18</sup>	None required.
8	Filter by unit/topic	T: Any node selected	Question pane refreshes to show all questions of that unit/topic.	As expected. <sup>19</sup>	None required.
		X: No questions of that unit/topic	Question pane shows no questions.	As expected. <sup>20</sup>	None required.
9	Filter by type	T: Any radio button selected	Question pane refreshes to show all questions of that type.	As expected. <sup>21</sup>	None required.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
		X: No questions of that type	Question pane shows no questions.	As expected. <sup>22</sup>	None required.
10	Alphabetically sort	T: There are saved questions	Questions are sorted alphabetically	As expected. <sup>23</sup>	None required.
		E: allQuestions is empty	No action.	As expected. <sup>24</sup>	None required.
11	View class logs	T: logs may or may not already exist	All class logs are loaded from document (document is created if it doesn't exist,) log field will be blank if there is no saved log.	If the form has already been loaded, classes are loaded again without the original list being cleared. <sup>25</sup>	Clear the class list every time the form loads. Second test conducted (11b.)
11 (b)	View class logs (REPEATED TEST)	T: logs may or may not already exist	All class logs are loaded from document (document is created if it doesn't exist,) log field will be blank if there is no saved log.	As expected. <sup>26</sup>	None required.
12	Add class	T: Class name field is filled in	Class is added to the list on the logs form.	As expected. <sup>27</sup>	None required.
		E: Class name field is blank	'Please provide a class name' dialogue.	As expected. <sup>28</sup>	None required.
13	Add/edit log	T: There is a class selected	Add/edit form loads (with existing log if there is one.)	As expected. <sup>29</sup>	None required.

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
14	Delete Class	T: There is a class selected	'Are you sure' dialogue. If yes, the class is deleted.	As expected. <sup>30</sup>	None required.
15	Delete Log	T: There is a class selected	'Are you sure' dialogue. If yes, the log is deleted.	As expected. <sup>31</sup>	None required.
16	Save quiz (text-based)	T: The quiz contains questions.	Save dialogue opens and two text files are saved. One quiz and one mark scheme with the same filename as entered for the quiz suffixed by "(mark scheme.)"	As expected. <sup>32</sup>	None required.
		E: The quiz contains no questions.	'Quiz contains no questions' dialogue. Nothing is exported.	As expected. <sup>33</sup>	None required.
17	Export quiz (for Moodle)	T: The quiz contains questions.	Save dialogue opens and one text file with integrated answers is saved. This can be uploaded to Moodle with no editing.	As expected. <sup>34</sup>	None required.



Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
		E: The quiz contains no questions.	'Quiz contains no questions' dialogue. Nothing is exported.	As expected. <sup>35</sup>	None required.
18	Loading questions	T: Question files exist and contain question data	All questions are loaded into the question pane.	As expected. <sup>36</sup>	None required.
		E: Question files do not exist	'Welcome to Benchmark' dialogue. (Files not existing suggest this is the first time the system has been run.)	As expected. <sup>37</sup>	None required.
		X: Question files exist but do not contain question data	'You haven't added any questions yet' dialogue. (Blank files suggest the system has been run before.)	As expected. <sup>38</sup>	None required.
19	Random ordering of multiple choice answers when saving quizzes	T: Quiz contains multiple choice questions	For test data where answers are "Correct" "X" "Y" "Z", each of the four elements should be in a different order for each question.	As expected. Input <sup>39</sup> produces output <sup>40</sup> . However, numbering is zero-based.	Begin numbering at i=1 rather than i=0 so quizzes begin at question one. Second test conducted (19b.)

Test No.	Description	TEX (Typical, Erroneous, Extreme)	Expected Outcome	Actual Outcome	Comments and Corrective Actions
19 (b)	Random ordering of multiple choice answers when saving quizzes (REPEAT TEST)	T: Quiz contains multiple choice questions	As for (19) but numbering should begin at 1 rather than 0.	As expected. Input <sup>41</sup> produces output <sup>42</sup> .	None required.
20	Testing the saving and loading of system files on the college network	T: Saving typical question data	Question is saved and written to the text file	Question is not saved because my user profile doesn't have permission to access the directory. <sup>43</sup> Attempting to navigate to the folder also results in an error message. <sup>44</sup>	Change the "userdirectory" environment variable to the Environment.SpecialFolder.Mydocuments variable in every instance of a file being written or read. Second test conducted (20b.)
20 (b)	Testing the saving and loading of system files on the college network (REPEAT TEST)	T: Saving typical question data	As for (20)	As expected. <sup>45</sup>	

## Trace Tables

<b>Testing the output from filtering questions by difficulty rating</b>
<b>Description:</b> The search algorithm iterates through every element in allQuestions, adding any elements with a matching question rating to lstAllQuestions. I'm searching 15 questions for any that have a difficulty rating of two. The same principle would apply when searching by unit, topic or question type.
<b>Code being tested:</b> <pre> Sub filterDifficulty(ByVal rating)     lstQuestions.Items.Clear()     For i = 0 To allQuestions.Count - 1         If allQuestions.ElementAt(i).getQRating = rating Then             lstQuestions.Items.Add(allQuestions(i).getQID &amp; " " &amp; vbTab                 &amp; allQuestions(i).getQText &amp; " (" &amp;                 allQuestions(i).getQAnswer &amp; ") [" &amp;                 allQuestions(i).getQTopic &amp; ", " &amp;                 allQuestions(i).getQRating &amp; "]"")         End If     Next     Call countLists() End Sub </pre>
<b>Expected result:</b> list display of allQuestions(2) list display of allQuestions(6) list display of allQuestions(10) list display of allQuestions(11) list display of allQuestions(12) where "list display of allQuestions(i)" denotes: (allQuestions(i).getQID & " " & vbTab & allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer & ") [" & allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating & "]"")

i	allQuestions.count	rating	allQuestions(i).getQRating	Output
0	14	2	1	
1			1	
2			2	list display of allQuestions(2)
3			1	
4			1	
5			1	
6			2	list display of allQuestions(6)
7			1	
8			1	
9			3	
10			2	list display of allQuestions(10)
11			2	list display of allQuestions(11)
12			2	list display of allQuestions(12)
13			1	
14			1	

**Final output (As expected):**

List display of allQuestions(2)  
 List display of allQuestions(6)  
 List display of allQuestions(10)  
 List display of allQuestions(11)  
 List display of allQuestions(12)

<b>Testing the loading of questions into the system from a text file</b>
<b>Description:</b>
The import algorithm reads the lines of the text file into a 1-dimensional array called allLines() and then reads every 9 consecutive lines and assigns these lines to the properties of a newly instantiated question. Each one of these questions is added to the allQuestions collection before the next is looped through. For this test, I'll be importing 15 questions from a text file.
<b>Code being tested:</b>
<pre> Dim count As Integer = 0   For i = 0 To allLines.Length - 8 Step 9     Dim savedQuestion As New Question     savedQuestion.setQID(count + 1)     savedQuestion.setText(allLines(i))     savedQuestion.setType(allLines(i + 1))     savedQuestion.setQAnswer(allLines(i + 2))     If savedQuestion.getQType = "MC" Then       savedQuestion.setIncorrect(allLines(i + 3), 0)       savedQuestion.setIncorrect(allLines(i + 4), 1)       savedQuestion.setIncorrect(allLines(i + 5), 2)     End If     savedQuestion.setQUnit(allLines(i + 6))     savedQuestion.setQTopic(allLines(i + 7))     savedQuestion.setQRating(allLines(i + 8))     count = count + 1     allQuestions.Add(savedQuestion)   Next </pre>
<b>Expected result:</b>
For every incrementation of count, allQuestions(count) should contain data from lines [count * 9] to [(count + 1) * 9 - 1], starting at line 0
<b>Abbreviations used:</b>
SQ(i) – savedQuestion(i) AL(i) – allLines(i)

i	count	SQ.getQID	SQ.getQText	SQ.getQType	SQ.getQAnswer	SQ.getIncorrect(0)	SQ.getIncorrect(1)	SQ.getIncorrect(2)	SQ.getQUnit	SQ.getTopic	SQ.getQRating
0	0	1	AL(0)	AL(1)	AL(2)	AL(3)	AL(4)	AL(5)	AL(6)	AL(7)	AL(8)
9	1	2	AL(9)	AL(10)	AL(11)	AL(12)	AL(13)	AL(14)	AL(15)	AL(16)	AL(17)
18	2	3	AL(18)	AL(19)	AL(20)	AL(21)	AL(22)	AL(23)	AL(24)	AL(25)	AL(26)
27	3	4	AL(27)	AL(28)	AL(29)	AL(30)	AL(31)	AL(32)	AL(33)	AL(34)	AL(35)
36	4	5	AL(36)	AL(37)	AL(38)	AL(39)	AL(40)	AL(41)	AL(42)	AL(43)	AL(44)
45	5	6	AL(45)	AL(46)	AL(47)	AL(48)	AL(49)	AL(50)	AL(51)	AL(52)	AL(53)
54	6	7	AL(54)	AL(55)	AL(56)	AL(57)	AL(58)	AL(59)	AL(60)	AL(61)	AL(62)
63	7	8	AL(63)	AL(64)	AL(65)	AL(66)	AL(67)	AL(68)	AL(69)	AL(70)	AL(71)
72	8	9	AL(72)	AL(73)	AL(74)	AL(75)	AL(76)	AL(77)	AL(78)	AL(79)	AL(80)
81	9	10	AL(81)	AL(82)	AL(83)	AL(84)	AL(85)	AL(86)	AL(87)	AL(88)	AL(89)
90	10	11	AL(90)	AL(91)	AL(92)	AL(93)	AL(94)	AL(95)	AL(96)	AL(97)	AL(98)
99	11	12	AL(99)	AL(100)	AL(101)	AL(102)	AL(103)	AL(104)	AL(105)	AL(106)	AL(107)
108	12	13	AL(108)	AL(109)	AL(110)	AL(111)	AL(112)	AL(113)	AL(114)	AL(115)	AL(116)
117	13	14	AL(117)	AL(118)	AL(119)	AL(120)	AL(121)	AL(122)	AL(123)	AL(124)	AL(125)
126	14	15	AL(126)	AL(127)	AL(128)	AL(129)	AL(130)	AL(131)	AL(132)	AL(133)	AL(134)

**Final collection (As expected):**

The table shows a representation of each property stored in the elements of the collection; data has been loaded as expected. For every incrimination of count, allQuestions(count) contains data from lines [count \* 9] to [(count + 1) \* 9 – 1], starting at line 0.

## Navigation Testing

The following table tests the unidirectional navigation between forms which link to each other. Blue cells indicate forms are not linked. Grey cells indicate there is no need for navigation. The key is as follows:

- frmAddEditQuestion - A
- frmAddLog - B
- frmAddNewClass - C
- frmClassLogs - D
- frmFilterDifficulty - E
- frmFilterType - F
- frmFilterUnitTopic - G
- frmHome - H
- frmTextSearch - I

Navigation successful ✓

No direct navigation, but focus returned to correct form upon closing ✓

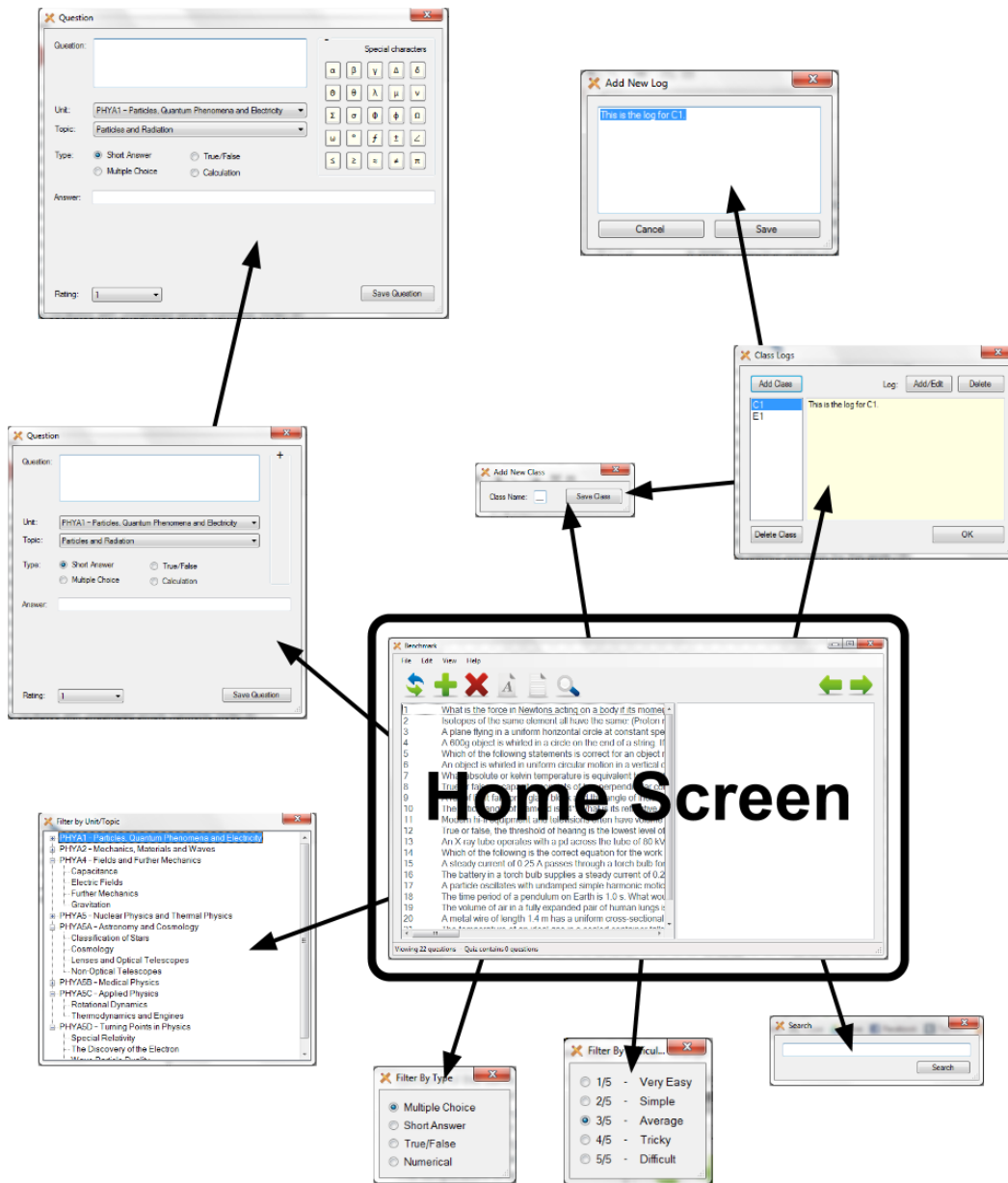
Navigation unsuccessful ✗

Navigating from: →	A	B	C	D	E	F	G	H	I
Navigating to: ↓	A	B	C	D	E	F	G	H	I
A								✓	
B				✓					
C				✓				✓	
D		✓	✓					✓	
E								✓	
F								✓	
G								✓	
H	✓			✓	✓	✓	✓		✓
I								✓	

# Maintenance

## Form Navigation Overview

The diagram below illustrates how all the forms of the system link to each other. Compare with navigation design and testing on pages 55 and 78.





## Class Overview

There are two classes from which objects are instantiated, modelled on their real counterparts.

### Group

(Every teaching group in the department)

Private properties:

```
groupName  
groupLog
```

Public methods:

```
getGroupName  
getGroupLog  
setGroupName ()  
setGroupLog ()
```

### Question

(Every question in the system)

Private properties:

```
qText  
qAnswer  
qUnit  
qTopic  
qType  
qIncorrect (2)  
qID  
qRating
```

Public methods:

```
getQText  
getQAnswer  
getQUnit  
getQTopic  
getQType  
getIncorrect ()  
getQRating  
getQID  
setQText ()  
setQAnswer ()  
setQUnit ()  
setQTopic ()  
setQType ()  
setQIncorrect ()  
setQRating
```

### Public Class Group

```
Dim groupName, GroupLog As String
Public Sub setGroupName(ByVal name)
    groupName = name
End Sub
Public Function getGroupName()
    Return groupName
End Function
Public Sub setGroupLog(ByVal log)
    GroupLog = log
End Sub
Public Function getGroupLog()
    Return GroupLog
End Function
End Class
```

---

### Public Class Question

```
Private qText, qAnswer, qUnit, qTopic, qType As String
Private qID, qRating As Integer
Private qIncorrect(2) As String

Public Sub setQText(ByVal text)
    qText = text
End Sub
Public Function getQText()
    Return qText
End Function
Public Sub setQAnswer(ByVal answer)
    qAnswer = answer
End Sub
Public Function getQAnswer()
    Return qAnswer
End Function
Public Function getIncorrect(ByVal x)
    Return qIncorrect(x)
End Function
Public Sub setIncorrect(ByVal incorrect, ByVal x)
    qIncorrect(x) = incorrect
End Sub
Public Sub setQTopic(ByVal topic)
    qTopic = topic
End Sub
Public Function getQTopic()
```

```
        Return qTopic
    End Function
    Public Sub setQUnit(ByVal unit)
        qUnit = unit
    End Sub
    Public Function getQUnit()
        Return qUnit
    End Function
    Public Sub setQID(ByVal id)
        qID = id
    End Sub
    Public Function getQID()
        Return qID
    End Function
    Public Sub setQRating(ByVal rating)
        qRating = rating
    End Sub
    Public Function getQRating()
        Return qRating
    End Function
    Public Sub setQType(ByVal type)
        qType = type
    End Sub
    Public Function getQType()
        Return qType
    End Function
```

End Class

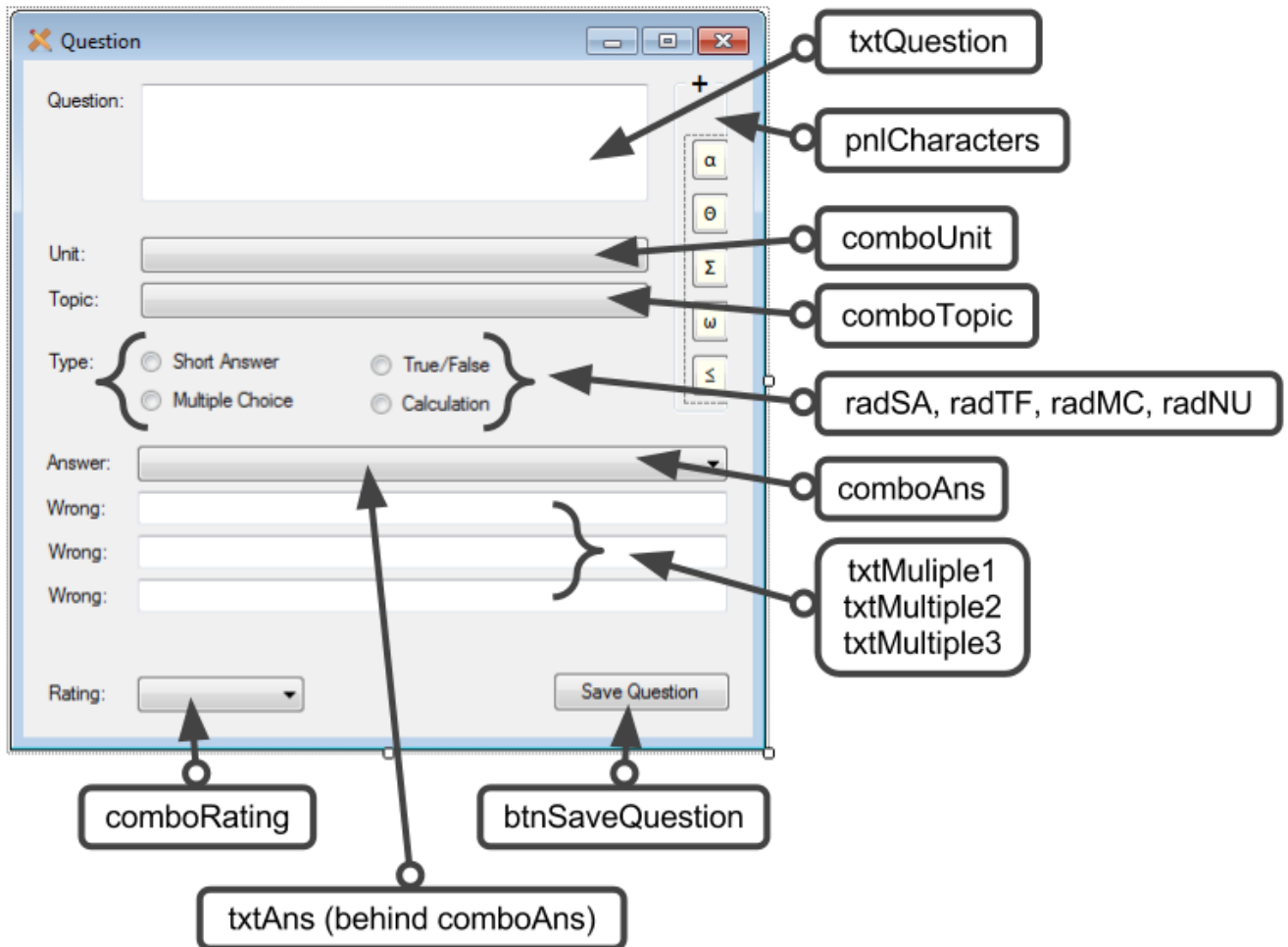
---

## Form Overview

Excluding dialogues, there are nine forms that make up the system.

### frmAddEditQuestion

Handles the input and of new question data, the editing of existing question data and the validation of all question data in the system.



### Public Class frmAddEditQuestion

```

Friend allquestions As New
System.Collections.ObjectModel.Collection(Of Question)
'Friend modifier makes the allquestions collection accessible in
this form
Dim questionType As String
Dim editing As Boolean

```

'Public questiontype variable and editing boolean are accessed by a majority of subroutines

```
Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Me.MinimizeBox = False
    Me.MaximizeBox = False
    Me.MaximumSize = New Point(450, 440)
    'Resets the size of the form to the default, with the panel
collapsed
    If editing = False Then
        'If the form has been called to add a question rather than
edit, clear the units combo box
        comboUnit.Items.Clear()
        'Gets the user directory name by retrieving the
environment variable "userprofile"
        Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
        If System.IO.File.Exists(userDirectory &
"\Benchmark\Units.txt") Then
            Dim allLines() As String =
System.IO.File.ReadAllLines(userDirectory & "\Benchmark\Units.txt")
            'If the file exists, load existing units into the
combo box
            If allLines.Length > 0 Then
                For i = 0 To allLines.Length - 1
                    'Iterate through the lines of the file,
reading a unit from each line
                    comboUnit.Items.Add(allLines(i))
                Next
                comboUnit.Update()
            End If
        End If
        comboUnit.SelectedIndex = 0
    End If
End Sub
Private Sub cancelChanges(ByVal sender As Object, ByVal e As
FormClosingEventArgs) Handles Me.FormClosing
    'In the event that the form is closing, whether or not the
user clicked 'save'
    editing = False
    'Resets the editing boolean so the form can load with empty
fields if necessary
    'Clears the text boxes
    Call clearQuestion()
End Sub
```

```
Private Sub SASelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radShortAns.CheckedChanged
    'Changes the question type of the question being created to
the short answer type
    'Hides the form controls associated with multiple choice and
true/false questions
    questionType = "SA"
    txtAns.Visible = True
    txtMultiple1.Visible = False
    txtMultiple2.Visible = False
    txtMultiple3.Visible = False
    comboAns.Visible = False
    Label8.Visible = False
    Label9.Visible = False
    Label10.Visible = False
End Sub
Private Sub NUSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radCalculation.CheckedChanged
    'Changes the question type of the question being created to
the numerical answer type
    'Hides the form controls associated with multiple choice and
true/false questions
    questionType = "NU"
    txtAns.Visible = True
    txtMultiple1.Visible = False
    txtMultiple2.Visible = False
    txtMultiple3.Visible = False
    comboAns.Visible = False
    Label8.Visible = False
    Label9.Visible = False
    Label10.Visible = False
End Sub
Private Sub TFSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RadTrueFalse.CheckedChanged
    'Changes the question type of the question being created to
the true/false type
    'Hides the form controls associated with multiple choice
questions
    questionType = "TF"
    txtAns.Visible = False
    txtMultiple1.Visible = False
    txtMultiple2.Visible = False
    txtMultiple3.Visible = False
    comboAns.Visible = True
```

```
Label8.Visible = False
Label9.Visible = False
Label10.Visible = False
comboAns.SelectedIndex = 0
End Sub
Private Sub MCSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radMultipleChoice.CheckedChanged
    'Changes the question type of the question being created to
the multiple choice type
    'Hides the form controls associated with true/false questions
questionType = "MC"
txtAns.Visible = True
txtMultiple1.Visible = True
txtMultiple2.Visible = True
txtMultiple3.Visible = True
comboAns.Visible = False
Label8.Visible = True
Label9.Visible = True
Label10.Visible = True
End Sub

Private Sub loadTopics(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles comboUnit.SelectedIndexChanged
    'Gets the unit selected in the units combo box and calls
findTopics to load the associated topics to the topics combo box
    Dim unit As String = comboUnit.SelectedItem.ToString
    Call findTopics(unit)
End Sub
Sub findTopics(ByVal unit As String)
    'Clears the topics combo box
    comboTopic.Items.Clear()
    'Gets the user directory name by retrieving the environment
variable "userprofile"
    Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    If System.IO.File.Exists(userDirectory & "\Benchmark\" & unit
& ".txt") Then
        'Open the file at the user profile directory with the name
of the selected unit
        Dim allLines() As String =
System.IO.File.ReadAllLines(userDirectory & "\Benchmark\" & unit &
".txt")
        'If the file exists, load existing topics into the combo
box
        If allLines.Length > 0 Then
```

```

        For i = 0 To allLines.Length - 1
            'Iterate through the lines of the file, reading a
            topic from each line
            comboTopic.Items.Add(allLines(i))
        Next
        comboTopic.Update()
    End If
    Else : comboTopic.Items.Add("Unknown topic")
        'If there is an error reading the file, load 'Unknown
        topic' into the box
        'The box cannot be left blank according to the validation
        rules of the form
    End If
    comboTopic.SelectedIndex = 0
End Sub

```

```

Sub clearQuestion()
    txtQuestion.Clear()
    txtAns.Clear()
    radShortAns.Checked = True
    txtMultiple1.Clear()
    txtMultiple2.Clear()
    txtMultiple3.Clear()
    txtQuestion.Select()
    'Clear all text boxes and reset to the default question type;
    short answer
    Me.Size = New Point(436, 440)
    pnlCharacters.Size = New Point(32, 208)
    pnlCharacters.Text = "+"
    pnlButtons.Visible = False
    lblCharacters.Visible = False
    comboAns.Size = New Point(351, 21)
    txtAns.Size = New Point(351, 20)
    txtMultiple1.Size = New Point(351, 20)
    txtMultiple2.Size = New Point(351, 20)
    txtMultiple3.Size = New Point(351, 20)
    btnSaveQuestion.Location = New Point(315, 363)
    Me.MaximumSize = New Point(450, 440)
    'Reset the form to its minimised size with the additional
    characters panel closed
    comboRating.SelectedIndex = 0
End Sub

```

```

Private Sub saveQuestion(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSaveQuestion.Click

```



```

Dim validquestion As Boolean = True
If txtQuestion.Text = "" Then 'No question
    validquestion = False
    MsgBox("Please enter a question.")
ElseIf comboUnit.Text = "" Then 'No unit
    validquestion = False
    MsgBox("Please select a unit.")
ElseIf comboTopic.Text = "" Then 'No topic
    validquestion = False
    MsgBox("Please select a topic.")
ElseIf (radShortAns.Checked = True Or radCalculation.Checked =
True Or radMultipleChoice.Checked = True) And txtAns.Text = "" Or
(RadTrueFalse.Checked = True And comboAns.Text =
"") Then 'No answer
    validquestion = False
    MsgBox("Please submit an answer.")
ElseIf radCalculation.Checked = True Then 'No T/F answer
    Dim integerTest As Integer
    Try
        integerTest = CInt(txtAns.Text)
    Catch
        validquestion = False
        MsgBox("Numerical answers cannot contain words or
characters.")
    End Try
    ElseIf radMultipleChoice.Checked = True And (txtMultiple1.Text
= "" Or txtMultiple2.Text = "" Or txtMultiple3.Text = "") Then
        validquestion = False
        MsgBox("Please submit three incorrect answers.") 'Missing
incorrect answer(s)
    End If
    'Save question data
    If validquestion = True Then
        Call saveQuestion()
        Call clearQuestion()
        editing = False
        Me.Close()
    End If
    frmHome.lstQuestions.ClearSelected()
End Sub
Sub saveQuestion()
    'Create a new instance of the question class
    Dim currentQuestion As New Question
    If editing = True Then
        'Do not create a new instance of the question class, edit
the existing one

```

```

    If questionType = "TF" Then
        'Rewrite the current question
        'Pass the answer as a boolean
        frmHome.updateQuestion(txtQuestion.Text, questionType,
CInt(comboRating.Text), comboAns.Text, txtMultiple1.Text,
txtMultiple2.Text, txtMultiple3.Text, comboUnit.Text, comboTopic.Text)
        Else 'Pass the answer as a string
            frmHome.updateQuestion(txtQuestion.Text, questionType,
CInt(comboRating.Text), txtAns.Text, txtMultiple1.Text,
txtMultiple2.Text, txtMultiple3.Text, comboUnit.Text, comboTopic.Text)
        End If
    Else
        currentQuestion.setQID(allquestions.Count)
        currentQuestion.setQText(txtQuestion.Text)
        currentQuestion.setQUnit(comboUnit.Text)
        currentQuestion.setQTopic(comboTopic.Text)
        currentQuestion.setQRating(CInt(comboRating.Text))
        currentQuestion.setQType(questionType)
        If questionType = "TF" Then
            currentQuestion.setQAnswer(comboAns.Text)
        Else : currentQuestion.setQAnswer(txtAns.Text)
        End If
        If questionType = "MC" Then Call
saveMultipleChoiceAnswers(currentQuestion)
        'Save the question just added as a new question
        frmHome.addNewQuestion(currentQuestion)
        frmHome.refreshList()
    End If
End Sub
Sub saveMultipleChoiceAnswers(ByRef newquestion As Question)
    newquestion.setIncorrect(txtMultiple1.Text, 0)
    newquestion.setIncorrect(txtMultiple2.Text, 1)
    newquestion.setIncorrect(txtMultiple3.Text, 2)
    newquestion.setQAnswer(txtAns.Text)
End Sub

Sub editQuestion(ByRef editQuestion As Question)
    'Set the editing boolean to true so the form controls don't
refresh
    editing = True
    comboUnit.Items.Clear()
    Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    If System.IO.File.Exists(userDirectory &
"\Benchmark\Units.txt") Then

```

```

    Dim allLines() As String =
System.IO.File.ReadAllLines(userDirectory & "\Benchmark\Units.txt")
    If allLines.Length > 0 Then
        For i = 0 To allLines.Length - 1
            comboUnit.Items.Add(allLines(i))
        Next
        comboUnit.Update()
    End If
End If
'Load the existing question text
txtQuestion.Text = editQuestion.getQText
'Try loading the existing units and topics
'If they are no longer in the system, load 'Unknown unit'
Try
    comboUnit.SelectedItem = editQuestion.getQUnit
Catch
    comboUnit.Items.Add("Unknown unit")
    comboUnit.SelectedIndex = comboUnit.Items.Count
End Try
Try
    comboTopic.SelectedItem = editQuestion.getQTopic
Catch
    comboTopic.Items.Add("Unknown topic")
    comboTopic.SelectedIndex = comboTopic.Items.Count
End Try
'Select the correct radio button based on question type
If editQuestion.getQType = "SA" Then
    radShortAns.Select()
    txtAns.Text = editQuestion.getQAnswer
ElseIf editQuestion.getQType = "TF" Then
    RadTrueFalse.Select()
    comboAns.Text = editQuestion.getQAnswer
ElseIf editQuestion.getQType = "MC" Then
    radMultipleChoice.Select()
    txtAns.Text = editQuestion.getQAnswer
    txtMultiple1.Text = editQuestion.getIncorrect(0)
    txtMultiple2.Text = editQuestion.getIncorrect(1)
    txtMultiple3.Text = editQuestion.getIncorrect(2)
ElseIf editQuestion.getQType = "NU" Then
    radCalculation.Select()
    txtAns.Text = editQuestion.getQAnswer
End If
'Set the rating
comboRating.Text = editQuestion.getQRating
End Sub

```

```

Private Sub toggleSize(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles pnlCharacters.MouseClick
    If pnlCharacters.Text = "+" Then
        'If the panel title contains the + symbol then it is in
its minimised position
        'Change the size of the form to contain the expanded
characters panel
        Me.Size = New Point(580, 440)
        pnlCharacters.Size = New Point(167, 208)
        pnlCharacters.Text = "-"
        pnlButtons.Visible = True
        lblCharacters.Visible = True
        comboAns.Size = New Point(486, 21)
        txtAns.Size = New Point(486, 20)
        txtMultiple1.Size = New Point(486, 20)
        txtMultiple2.Size = New Point(486, 20)
        txtMultiple3.Size = New Point(486, 20)
        btnSaveQuestion.Location = New Point(448, 363)
        Me.MinimumSize = New Point(580, 440)
        'Create nine new instances of the point class to resize:
        '1: the form base
        '2: the panel containing the additional characters
        '3: the true/false answer combo box
        '4: the text field answer box
        '5, 6, 7: the incorrect answer boxes for multiple choice
questions
        '8: the location of the save button
        '9: the new minimum size of the expanded form
    Else
        'If the panel title contains the - symbol then it is in
its maximised position
        'Change the size of the form to contain the contracted
characters panel
        Me.Size = New Point(436, 440)
        pnlCharacters.Size = New Point(32, 208)
        pnlCharacters.Text = "+"
        pnlButtons.Visible = False
        lblCharacters.Visible = False
        comboAns.Size = New Point(351, 21)
        txtAns.Size = New Point(351, 20)
        txtMultiple1.Size = New Point(351, 20)
        txtMultiple2.Size = New Point(351, 20)
        txtMultiple3.Size = New Point(351, 20)
        btnSaveQuestion.Location = New Point(315, 363)
        Me.MaximumSize = New Point(450, 440)
    End If
End Sub

```

```

'Create nine new instances of the point class to resize:
'1: the form base
'2: the panel containing the additional characters
'3: the true/false answer combo box
'4: the text field answer box
'5, 6, 7: the incorrect answer boxes for multiple choice
questions
'8: the location of the save button
'9: the new minimum size of the contracted form
End If
End Sub

'Call the insertcharacter subroutine and pass the relevant
character to be inserted into the question text
Private Sub btnα_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnα.Click
    Call insertCharacter("α")
End Sub
Private Sub btnβ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnβ.Click
    Call insertCharacter("β")
End Sub
Private Sub btnγ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnγ.Click
    Call insertCharacter("γ")
End Sub
Private Sub btnΔ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnΔ.Click
    Call insertCharacter("Δ")
End Sub
Private Sub btn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn.Click
    Call insertCharacter("δ")
End Sub
Private Sub btnθ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnθ.Click
    Call insertCharacter("θ")
End Sub
Private Sub btnθ2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnθ2.Click
    Call insertCharacter("θ")
End Sub
Private Sub btnλ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnλ.Click
    Call insertCharacter("λ")
End Sub

```

```
Private Sub btnμ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnμ.Click
    Call insertCharacter("μ")
End Sub
Private Sub btnν_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnν.Click
    Call insertCharacter("ν")
End Sub
Private Sub btnΣ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnΣ.Click
    Call insertCharacter("Σ")
End Sub
Private Sub btnσ2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnσ2.Click
    Call insertCharacter("σ")
End Sub
Private Sub btnΦ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnΦ.Click
    Call insertCharacter("Φ")
End Sub
Private Sub btnφ2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnφ2.Click
    Call insertCharacter("φ")
End Sub
Private Sub btnΩ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnΩ.Click
    Call insertCharacter("Ω")
End Sub
Private Sub btnω2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnω2.Click
    Call insertCharacter("ω")
End Sub
Private Sub btnDegrees_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnDegrees.Click
    Call insertCharacter("°")
End Sub
Private Sub btnf_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnf.Click
    Call insertCharacter("f")
End Sub
Private Sub btnPlusMinus_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnPlusMinus.Click
    Call insertCharacter("±")
End Sub
Private Sub btnAngle_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnAngle.Click
```

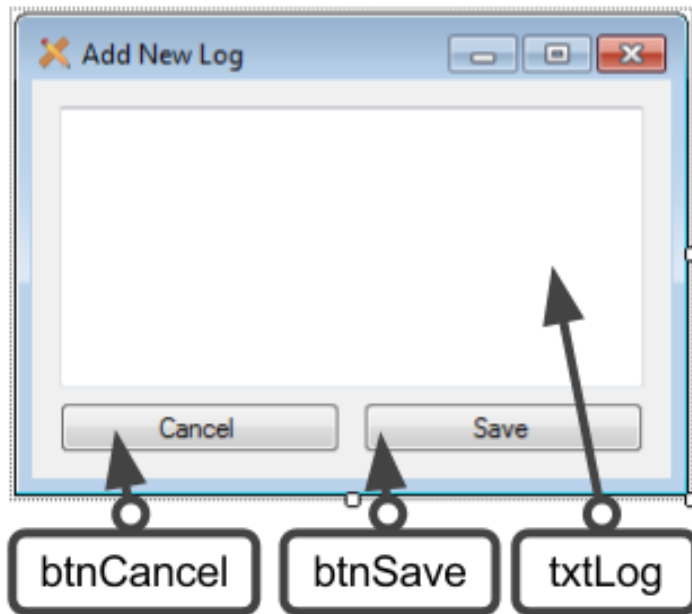
```
        Call insertCharacter("<")
    End Sub
    Private Sub btnLessEqual_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnLessEqual.Click
        Call insertCharacter("<=")
    End Sub
    Private Sub btnMoreEqual_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnMoreEqual.Click
        Call insertCharacter(">=")
    End Sub
    Private Sub btnApprox_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnApprox.Click
        Call insertCharacter("≈")
    End Sub
    Private Sub btnπ_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnπ.Click
        Call insertCharacter("π")
    End Sub
    Private Sub btnNotEqual_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnNotEqual.Click
        Call insertCharacter("≠")
    End Sub
    Sub insertCharacter(ByVal character As String)
        'Update the question text field at the position of the caret
with the character passed to it
        txtQuestion.Text =
txtQuestion.Text.Insert(txtQuestion.SelectionStart, character)
    End Sub

End Class
```

---

## frmAddLog

Handles the input of new class log data and the editing of existing class logs in the system.



```
Public Class frmAddLog
```

```
    Private Sub startup(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles MyBase.Load
```

```
        'Sets the properties determining whether the user can  
minimise or maximise the form
```

```
        'Hides the form's control bar
```

```
        Me.MinimizeBox = False
```

```
        Me.MaximizeBox = False
```

```
        Me.ControlBox = False
```

```
        Try
```

```
            'Loads the log of the class selected in the logs form to  
the text box
```

```
            txtLog.Text =
```

```
frmClassLogs.allClasses(frmClassLogs.lstAllClasses.SelectedIndex).getG  
roupLog
```

```
        Catch
```

```
            Me.Close()
```

```
        End Try
```

```
    End Sub
```

```
    Private Sub cancelChanges(ByVal sender As System.Object, ByVal e  
As System.EventArgs) Handles btnCancel.Click
```

```
        If txtLog.Text <> "This class doesn't have a log yet." Then
```



```
        'If the contents of the text box has changed since
loading, notify the user they will lose changes
        Dim result As MsgBoxResult = MsgBox("You will lose unsaved
changes. Are you sure you want to cancel?", MsgBoxStyle.YesNo)
        If result = MsgBoxResult.Yes Then
            Me.Close()
        End If
        'If the user does not choose yes on the save changes
dialog, leave the form open
        Else : Me.Close()
        End If
    End Sub

    Private Sub handleReturn(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtLog.KeyPress
        'If the user presses the return key, set the KeyPressEvent to
handled so the return is not implemented
        'This is to stop the user using the return key in logs
        If e.KeyChar = Chr(Keys.Return) Then e.Handled = True
    End Sub

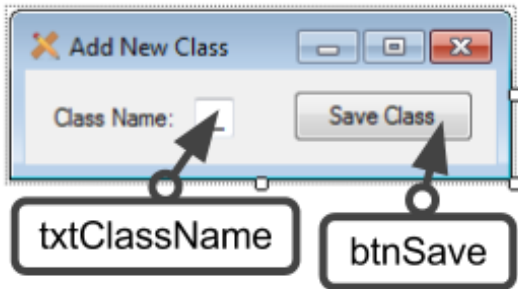
    Private Sub saveLog(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSave.Click
        If txtLog.Text <> "" And Len(txtLog.Text) <= 500 Then
            'If the log is not empty and not longer than 500
characters
            Dim log As String = txtLog.Text
            frmClassLogs.addLog(log)
            'Call the addLog subroutine, passing the contents of the
log text box
            txtLog.Clear()
            'Clear the text box ready for the next form loading event
            Me.Close()
        ElseIf Len(txtLog.Text) > 500 Then
            MsgBox("You cannot save a log more than 500 characters
long.")
            'Notify the user that their entry was too long
        Else
            MsgBox("You cannot save a blank log.")
            'Notify the user that the field cannot be left blank
        End If
        txtLog.Select()
    End Sub

End Class
```

---

frmAddNewClass

Handles the input and validation of classes being added to the system.



Public Class frmAddNewClass

```
Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

```
'Sets the properties determining whether the user can
minimise or maximise the form
```

```
Me.MinimizeBox = False
```

```
Me.MaximizeBox = False
```

```
End Sub
```

```
Private Sub saveClass(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSave.Click
```

```
If txtClassName.Text <> "" Then
```

```
'If the user has not left the class name text field blank
```

```
'Create a new instance of the group class
```

```
Dim newclass As New Group
```

```
newclass.setGroupName(txtClassName.Text)
```

```
newclass.setGroupLog("This class doesn't have a log yet.")
```

```
'Set the group's name and log
```

```
frmClassLogs.allClasses.Add(newclass)
```

```
'Add the instantiated group into the allclasses system
```

```
collection
```

```
frmClassLogs.refreshLog()
```

```
'Refresh the logs display on the class logs form
```

```
txtClassName.Text = ""
```

```
txtClassName.Select()
```

```
'Reset the form controls
```

```
Me.Close()
```

```
Else : MsgBox("Please enter a class name.")
```

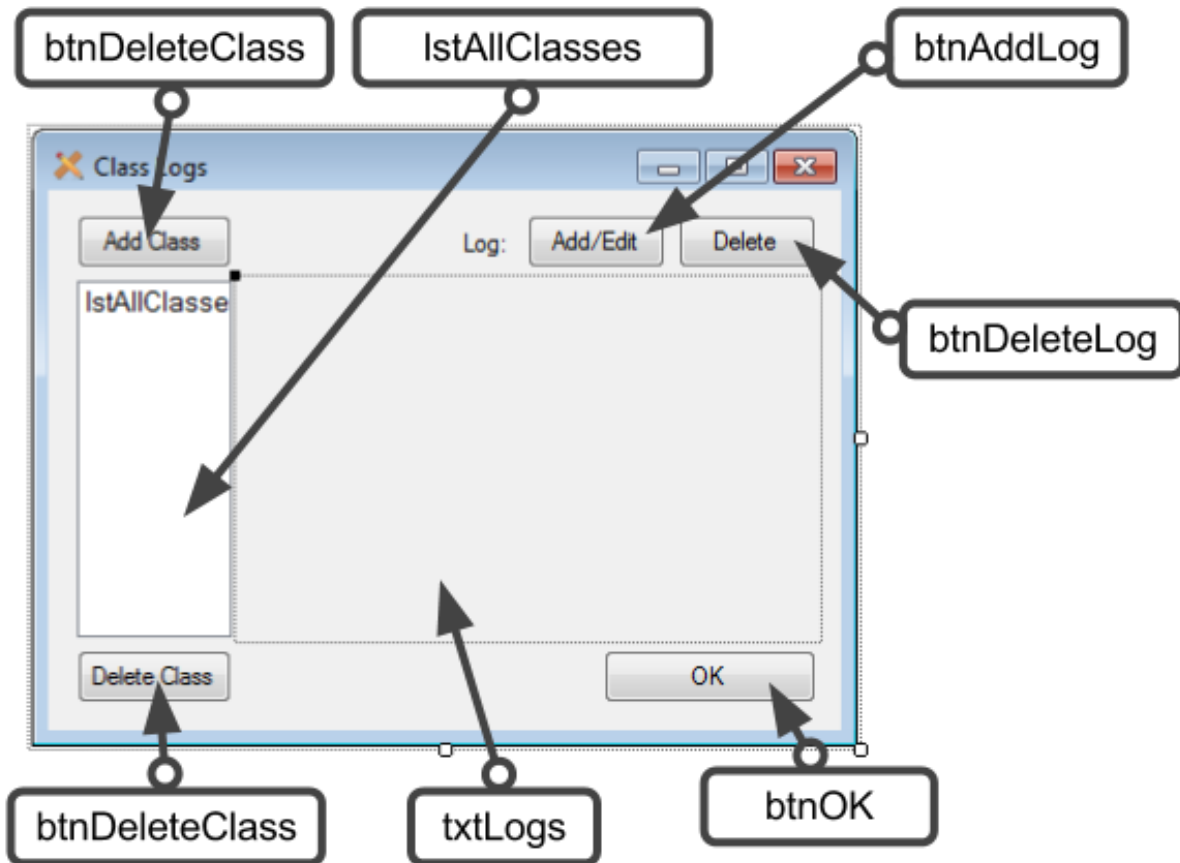
```
End If
```

```
End Sub
```

```
End Class
```

### frmClassLogs

Handles the displaying of all classes and logs in the system, links to all other class log forms.



```
Public Class frmClassLogs
```

```
    'Creates a new instance of collections of groups to store all physics groups in the system
```

```
    Public allClasses As New System.Collections.ObjectModel.Collection(Of Group)
```

```
    Private Sub startup(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
        'Sets the properties which determine whether the user can maximise and minimise the form
```

```
        Me.MinimizeBox = False
```

```
        Me.MaximizeBox = False
```

```
        'Calls the importlogs subroutine to populate the allclasses collection
```

```
        Call importLogs()
```

```
    End Sub
```

```
    Private Sub closeForm(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOK.Click
```

```
        'Close the form
        Me.Close()
    End Sub

    Function importLogs()
        'Gets the user directory name by retrieving the environment
        variable "userprofile"
        Dim userDirectory As String =
        Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
        If System.IO.File.Exists(userDirectory &
        "\Benchmark\Logs.txt") Then
            'If the file exists, load existing classes
            Dim allLines() As String =
            System.IO.File.ReadAllLines(userDirectory & "\Benchmark\Logs.txt")
            If allLines.Length > 0 Then
                For i = 0 To allLines.Length - 1 Step 2
                    'One group is read every two lines
                    Dim savedClass As New Group
                    'Create a runtime instance of the group class and
                    add data from the subsequent lines
                    savedClass.setGroupName(allLines(i))
                    savedClass.setGroupLog(allLines(i + 1))
                    allClasses.Add(savedClass)
                    'Add the imported class to the allclasses
                    collection
                Next
                lstAllClasses.Items.Clear()
                Call refreshLog()
                'Refresh the class list to update it from the
                collection
            End If
        End If
    End Function
    Sub backupLogs()
        'Gets the user directory name by retrieving the environment
        variable "userprofile"
        Dim userDirectory As String =
        Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
        If System.IO.File.Exists(userDirectory &
        "\Benchmark\Logs.txt") Then System.IO.File.Delete(userDirectory &
        "\Benchmark\Logs.txt")
        'Clear the previous backup file and rewrite it from the
        allclasses collection
        Dim backup As New System.IO.StreamWriter(userDirectory &
        "\Benchmark\Logs.txt")
```

```
'Create a new instance of the streamwriter class
For Each item In allClasses
    'For every group in the system, call the backuplog
function to write it to the file
    backup.Write(backupLog(item))
Next
backup.Close()
'Close the streamwriter
End Sub
Function backupLog(ByVal g As Group)
    'Return data held on a group to be written to the file
    backupLog = g.getGroupName & vbNewLine & g.getGroupLog &
vbNewLine
End Function

Sub refreshLog()
    lstAllClasses.Items.Clear()
    'Clear the classes list and repopulate it from the allclasses
collection
    For Each item In allClasses
        lstAllClasses.Items.Add(item.getGroupName)
    Next
    If allClasses.Count = 0 Then txtLogs.Text = ""
    Try
        lstAllClasses.ClearSelected()
        lstAllClasses.SelectedIndex = 0
        'Try selecting the first element in the class list
        'Catch the exception which will be thrown if allclasses is
empty
    Catch
    End Try
    Call backupLogs()
End Sub

Private Sub addClass(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAddClass.Click
    'Show the add class dialog
    frmAddNewClass.ShowDialog()
End Sub
Private Sub deleteClass(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDeleteClass.Click
    If lstAllClasses.SelectedIndex > -1 Then
        'If there is a group selected
```

```

        Dim result As MsgBoxResult = MsgBox("Are you sure you want
to delete this class?", MsgBoxStyle.YesNo)
        'Confirm the deletion of the group with the user
        If result = MsgBoxResult.Yes Then
            allClasses.RemoveAt(1stAllClasses.SelectedIndex)
            'Remove the group from the allclasses collection
            Call refreshLog()
            'Refresh the list display of the groups
        End If
    Else : MsgBox("Please select a class.")
    End If
End Sub

Private Sub addLog(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAddLog.Click
    If 1stAllClasses.SelectedIndex > -1 Then
        frmAddLog.ShowDialog()
        'Open the addlog form if there is a group selected
    Else : MsgBox("Please select a class.")
    End If
End Sub
Sub addLog(ByVal log)
    allClasses(1stAllClasses.SelectedIndex).setGroupLog(log)
    'Set the log of the selected group to the log passed to it
    Call refreshLog()
    'Refresh the group list with the updated log
End Sub
Private Sub deleteLog(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDeleteLog.Click
    If 1stAllClasses.SelectedIndex > -1 Then
        'If there is a group selected, confirm deletion of the log
with the user
        Dim result As MsgBoxResult = MsgBox("Are you sure you want
to clear the current log?", MsgBoxStyle.YesNo)
        If result = MsgBoxResult.Yes Then
            'Set the log back to the default text
            allClasses(1stAllClasses.SelectedIndex).setGroupLog("This class
doesn't have a log yet.")
            Call refreshLog()
            'Refresh the group list with the updated log
        End If
    Else : MsgBox("Please select a class.")
    End If
End Sub

```

```

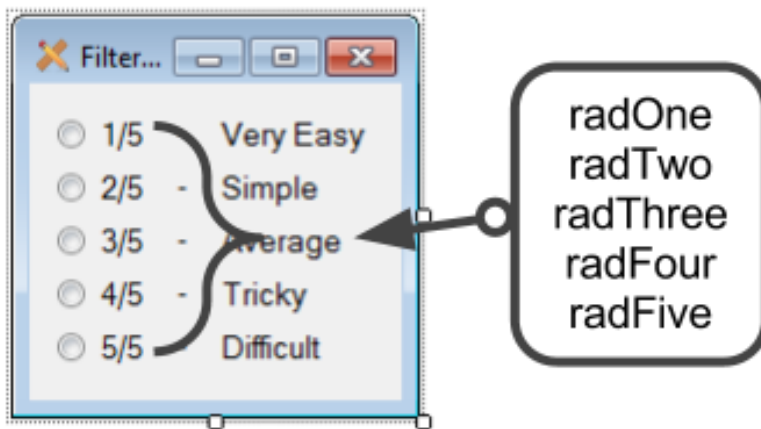
Private Sub logPreview(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstAllClasses.SelectedIndexChanged
    txtLogs.Text =
allClasses(lstAllClasses.SelectedIndex).getGroupLog
    'Show the log of the currently selected group in the panel on
the right of the form
End Sub
End Class

```

---

### frmFilterDifficulty

Handles the input of difficulty ratings to filter questions.



### Public Class frmFilterDifficulty

```

Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    'Sets the properties which determine whether the user can
maximise or minimise the form
    Me.MinimizeBox = False
    Me.MaximizeBox = False
    radThree.Select()
    'Select the default difficulty radio button to filter by
End Sub

```

```

Private Sub oneSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radOne.CheckedChanged
    frmHome.filterDifficulty(1)
    'Pass 1 to the filterdifficulty routine on the home form
End Sub

```

```
Private Sub twoSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radTwo.CheckedChanged
    frmHome.filterDifficulty(2)
    'Pass 2 to the filterdifficulty routine on the home form
End Sub
```

```
Private Sub threeSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radThree.CheckedChanged
    frmHome.filterDifficulty(3)
    'Pass 3 to the filterdifficulty routine on the home form
End Sub
```

```
Private Sub fourSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radFour.CheckedChanged
    frmHome.filterDifficulty(4)
    'Pass 4 to the filterdifficulty routine on the home form
End Sub
```

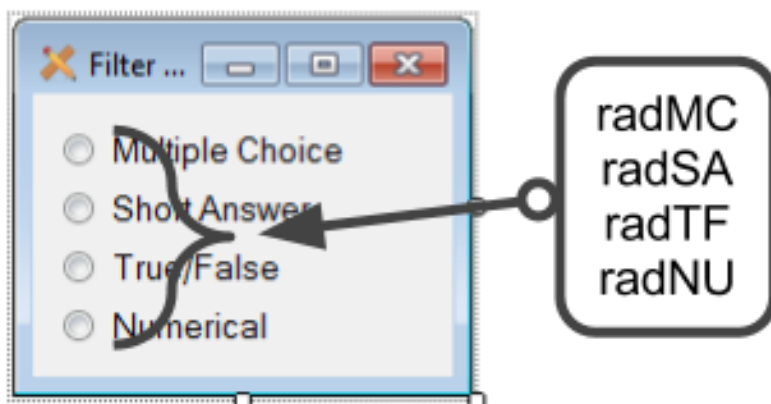
```
Private Sub fiveSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radFive.CheckedChanged
    frmHome.filterDifficulty(5)
    'Pass 5 to the filterdifficulty routine on the home form
End Sub
```

End Class

---

frmFilterType

Handles the input of question types to filter questions.



```
Public Class frmFilterType
```

```
Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```



```
'Sets the properties which determine whether the user can
maximise or minimise the form
Me.MinimizeBox = False
Me.MaximizeBox = False
radMC.Select()
'Select the default type radio button to filter by
End Sub

Private Sub MCSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radMC.CheckedChanged
    frmHome.filterType("MC")
    'Pass "MC" to the filterType routine on the home form
End Sub

Private Sub SASelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radSA.CheckedChanged
    frmHome.filterType("SA")
    'Pass "SA" to the filterType routine on the home form
End Sub

Private Sub TFSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radTF.CheckedChanged
    frmHome.filterType("TF")
    'Pass "TF" to the filterType routine on the home form
End Sub

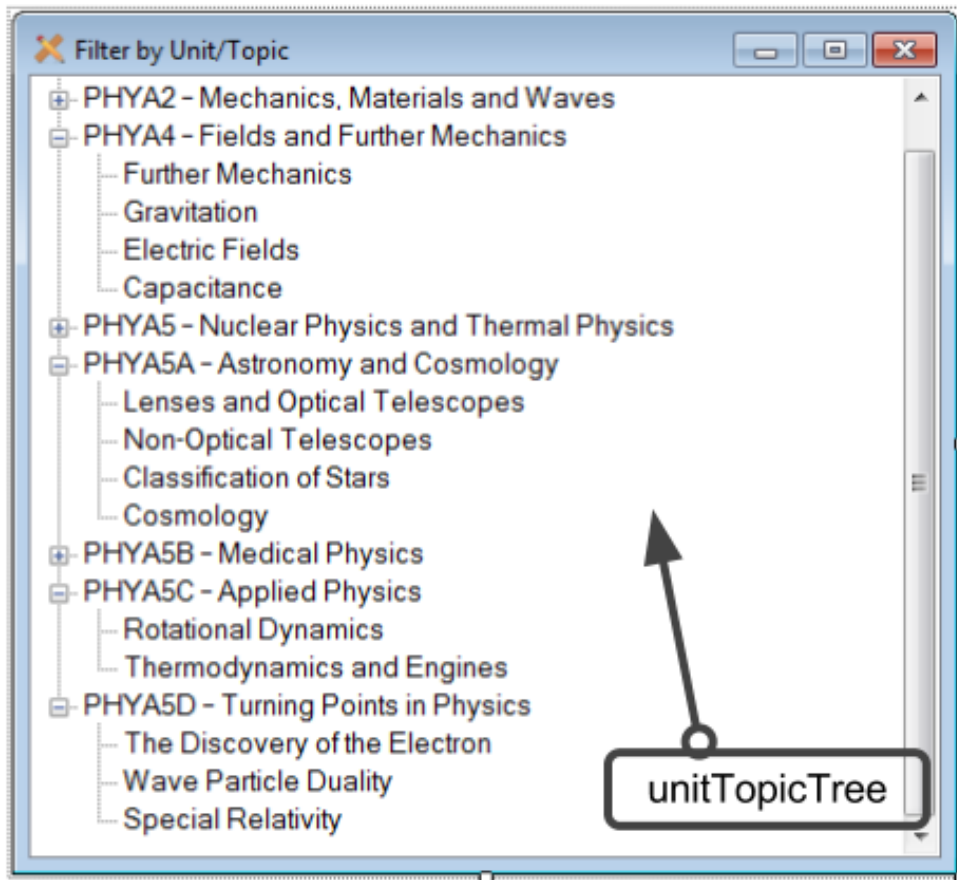
Private Sub NUSelected(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radNU.CheckedChanged
    frmHome.filterType("NU")
    'Pass "NU" to the filterType routine on the home form
End Sub

End Class
```

---

## frmFilterUnitTopic

Handles the input of units and topics to filter questions.



### Public Class frmFilterUnitTopic

```

Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Me.MinimizeBox = False
    Me.MaximizeBox = False
    'Sets the properties which determine whether the user can
    maximise or minimise the form
    unitTopicTree.Nodes.Clear()
    'Clear the data from the tree
    'Gets the user directory name by retrieving the environment
    variable "userprofile"
    Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    If System.IO.File.Exists(userDirectory &
"\Benchmark\Units.txt") Then
        'If the file exists, load existing units into an array
    
```

```

        Dim allLines() As String =
System.IO.File.ReadAllLines(userDirectory & "\Benchmark\Units.txt")
        If allLines.Length > 0 Then
            For i = 0 To allLines.Length - 1
                'For every unit read, add the unit as a new node
to the tree
                Dim parentNode As TreeNode =
unitTopicTree.Nodes.Add(allLines(i))
                Try
                    Dim topicLines() As String =
System.IO.File.ReadAllLines(userDirectory & "\Benchmark\" &
allLines(i) & ".txt")
                    'Read the contents of the file with the same
name as the unit into an array
                    If topicLines.Length > 0 Then
                        For j = 0 To topicLines.Length - 1
                            parentNode.Nodes.Add(topicLines(j))
                            'For every topic within the units file
read, add the topic as a child node
                        Next
                    End If
                Catch : parentNode.Nodes.Add("Unknown topic")
                    'If the topics file isn't found, catch the
exception
                End Try
            Next
            unitTopicTree.Update()
            'Update the tree with the new units and topics
        End If
    Else : unitTopicTree.Nodes.Add("Unknown unit")
        'If the units file isn't found, catch the exception
    End If
    unitTopicTree.CollapseAll()
End Sub

```

```

Private Sub nodeChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
unitTopicTree.AfterSelect
    'Declare a variable based on the index of the selected node in
a tree
    'Units (parent nodes) will have a treeIndex of 1
    'Topics (child nodes) will have a treeIndex of 2
    Dim treeIndex As Integer = unitTopicTree.SelectedNode.Level +
1
    Dim nodeTitle As String = unitTopicTree.SelectedNode.Text

```

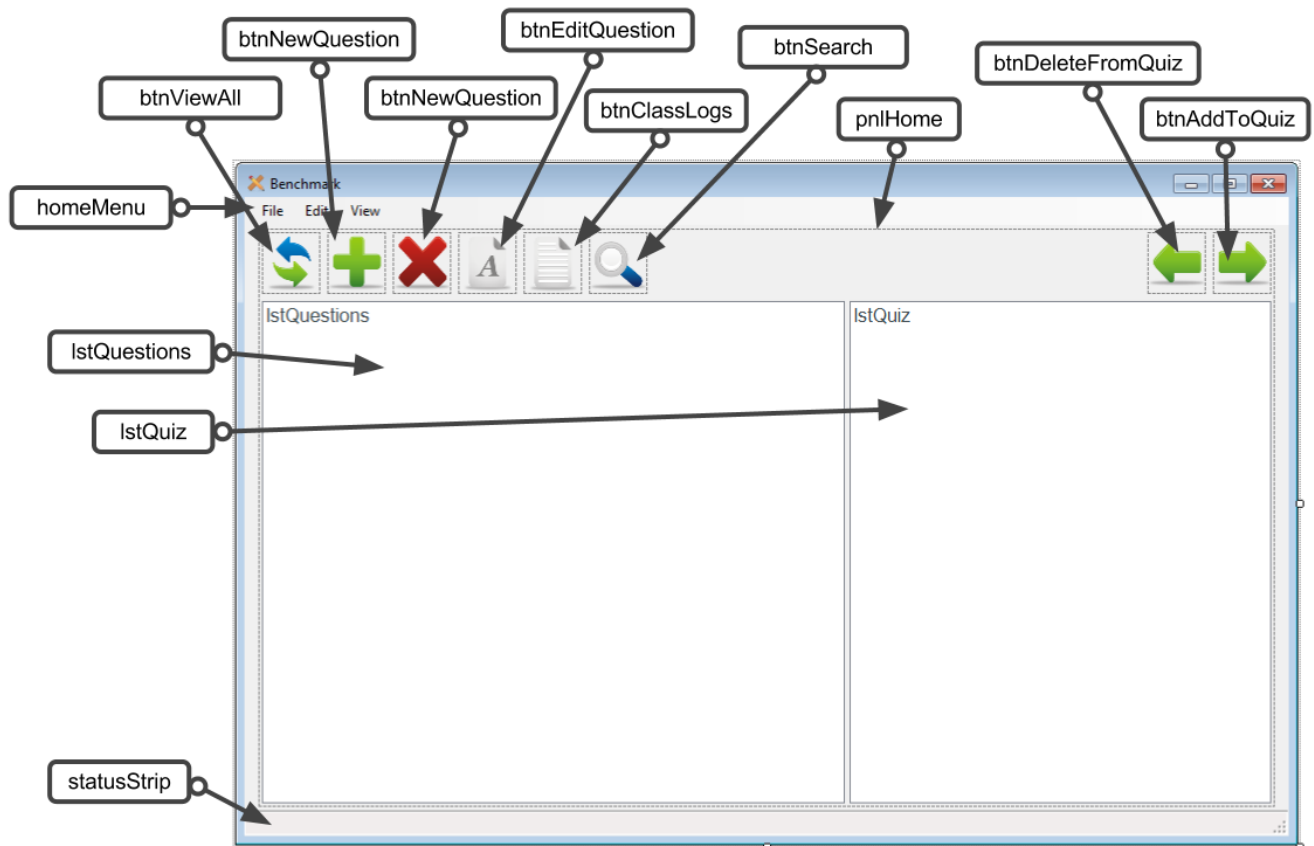
```
'Call the filterunittopic subroutine, passing the index and
title of the node
```

```
frmHome.filterUnitTopic(treeIndex, nodeTitle)
End Sub
```

End Class

frmHome

The main form of the system, displays all questions in the system and the current quiz.



Public Class frmHome

```
'Creates a new instance of collections of questions to store all
questions in the system
```

```
Dim allQuestions As New
System.Collections.ObjectModel.Collection(Of Question)
```

```
'Creates a new instance of collections of questions to store all
questions in the current quiz
```

```
Dim newQuiz As New System.Collections.ObjectModel.Collection(Of
Question)
```

```

Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    'Startup routines
    'Sets the tooltips for the shortcut bar
    'Calls the import subroutine
    'Calls the count routine to display the number of questions in
the system
    Call setToolTips()
    Call importAllQuestions()
    Call countLists()
    Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    System.IO.Directory.CreateDirectory(userDirectory &
"\Benchmark")
    'Create a folder in the user profile directory to save all
system data
    'Create will not affect the directory if it already exists
    'The following code will only be executed if no 'units' text
file exists
    'i.e. the default units have not been changed because the
system has never been run or the file has been deleted.
    If System.IO.File.Exists(userDirectory &
"\Benchmark\Units.txt") = False Then
        'Write a text file with a list of the default units
        Dim defaultUnitSetup As New
System.IO.StreamWriter(userDirectory & "\Benchmark\Units.txt")
        defaultUnitSetup.Write("PHYA1 - Particles, Quantum
Phenomena and Electricity" & vbNewLine &
                                "PHYA2 - Mechanics, Materials and
Waves" & vbNewLine &
                                "PHYA4 - Fields and Further
Mechanics" & vbNewLine &
                                "PHYA5 - Nuclear Physics and
Thermal Physics" & vbNewLine &
                                "PHYA5A - Astronomy and Cosmology"
                                & vbNewLine &
                                "PHYA5B - Medical Physics" &
                                & vbNewLine &
                                "PHYA5C - Applied Physics" &
                                & vbNewLine &
                                "PHYA5D - Turning Points in
Physics")
        defaultUnitSetup.Close()
    'Close the streamwriter and write text files with topic
lists for each unit in the units text file

```

```
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA1 - Particles,
Quantum Phenomena and Electricity.txt")
        defaultUnitSetup.Write("Particles and Radiation" &
vbNewLine &
                                "Electromagnetic Radiation and
Quantum Phenomena" & vbNewLine &
                                "Current Electricity")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA2 - Mechanics,
Materials and Waves.txt")
        defaultUnitSetup.Write("Mechanics" & vbNewLine &
                                "Materials" & vbNewLine &
                                "Waves")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA4 - Fields and
Further Mechanics.txt")
        defaultUnitSetup.Write("Further Mechanics" & vbNewLine &
                                "Gravitation" & vbNewLine &
                                "Electric Fields" & vbNewLine &
                                "Capacitance")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA5 - Nuclear
Physics and Thermal Physics.txt")
        defaultUnitSetup.Write("Radioactivity" & vbNewLine &
                                "Nuclear Energy" & vbNewLine &
                                "Thermal Physics")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA5A - Astronomy
and Cosmology.txt")
        defaultUnitSetup.Write("Lenses and Optical Telescopes" &
vbNewLine &
                                "Non-Optical Telescopes" &
vbNewLine &
                                "Classification of Stars" &
vbNewLine &
                                "Cosmology")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA5B - Medical
Physics.txt")
        defaultUnitSetup.Write("Physics of the Eye" & vbNewLine &
```

```

        "Physics of the Ear" & vbNewLine &
        "Biological Measurement" &
vbNewLine &
        "Non-Ionising Imaging" & vbNewLine
&
        "X-Ray Imaging")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA5C - Applied
Physics.txt")
        defaultUnitSetup.Write("Rotational Dynamics" & vbNewLine &
        "Thermodynamics and Engines")
        defaultUnitSetup.Close()
        defaultUnitSetup = New
System.IO.StreamWriter(userDirectory & "\Benchmark\PHYA5D - Turning
Points in Physics.txt")
        defaultUnitSetup.Write("The Discovery of the Electron" &
vbNewLine &
        "Wave Particle Duality" & vbNewLine
&
        "Special Relativity")
        defaultUnitSetup.Close()
    End If
End Sub
Sub setToolTips()
    'Sets the tooltips for all the shortcut icons on the home form
    iconsToolTip.SetToolTip(btnViewAll, "View All Questions")
    iconsToolTip.SetToolTip(btnNewQuestion, "Add New Question")
    iconsToolTip.SetToolTip(btnDeleteQuestion, "Delete Question")
    iconsToolTip.SetToolTip(btnEditQuestion, "Edit Question")
    iconsToolTip.SetToolTip(btnSearch, "Search Questions")
    iconsToolTip.SetToolTip(btnClassLogs, "View Class Logs")
    iconsToolTip.SetToolTip(btnAddToQuiz, "Add Question To Quiz")
    iconsToolTip.SetToolTip(btnDeleteFromQuiz, "Remove Question
From Quiz")
End Sub
Function importAllQuestions()
    'Gets the user directory name by retrieving the environment
variable "userprofile"
    Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
    If System.IO.File.Exists(userDirectory &
"\Benchmark\Questions.txt") Then
        'If the file exists, load existing questions into an array

```

```

        Dim allLines() As String =
System.IO.File.ReadAllLines(userDirectory &
"\Benchmark\Questions.txt")
        If allLines.Length > 0 Then
            Dim count As Integer = 0
            For i = 0 To allLines.Length - 8 Step 9
                'One question is read every nine lines
                Dim savedQuestion As New Question
                'Create a runtime instance of the question class
and add data from the subsequent lines
                savedQuestion.setQID(count + 1)
                'Count is zero-based but the list index should
begin at 1
                savedQuestion.setQText(allLines(i))
                savedQuestion.setQType(allLines(i + 1))
                savedQuestion.setQAnswer(allLines(i + 2))
                If savedQuestion.getQType = "MC" Then
                    'Read incorrect answers for multiple choice
questions
                    savedQuestion.setIncorrect(allLines(i + 3), 0)
                    savedQuestion.setIncorrect(allLines(i + 4), 1)
                    savedQuestion.setIncorrect(allLines(i + 5), 2)
                End If
                savedQuestion.setQUnit(allLines(i + 6))
                savedQuestion.setQTopic(allLines(i + 7))
                savedQuestion.setQRating(allLines(i + 8))
                count = count + 1
                allQuestions.Add(savedQuestion)
                'Add the imported question into allQuestions
collection
            Next
            Call refreshList()
            'Refresh the question list and recount the lists
        End If
    Else : MsgBox("Welcome to Benchmark. You haven't added any
questions yet.")
        ' if the files don't exist, this is the first time the
software has been used.
    End If
End Function
Sub backupAllQuestions()
    'Gets the user directory name by retrieving the environment
variable "userprofile"
    Dim userDirectory As String =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)

```



```

    If System.IO.File.Exists(userDirectory &
"\Benchmark\Questions.txt") Then System.IO.File.Delete(userDirectory &
"\Benchmark\Questions.txt")
    'Clear the previous backup and rewrite it with the contents of
allquestions
    Dim backup As New System.IO.StreamWriter(userDirectory &
"\Benchmark\Questions.txt")
    'Create a new instance of the streamwriter class
    For i = 0 To allQuestions.Count - 1
        'For every question in the system, call the backupQuestion
function to write it to the file
        backup.Write(backupQuestion(allQuestions(i)))
    Next
    backup.Close()
    'Close the streamwriter
End Sub
Function backupQuestion(ByVal q As Question)
    'Return data held on a question to be written to the file
    backupQuestion = q.getQText & vbNewLine & q.getQType &
vbNewLine & q.getQAnswer &
        vbNewLine & q.getIncorrect(0) & vbNewLine &
q.getIncorrect(1) & vbNewLine &
        q.getIncorrect(2) & vbNewLine & q.getQUnit & vbNewLine &
q.getQTopic & vbNewLine & q.getQRating & vbNewLine
End Function

Sub refreshList() Handles btnViewAll.Click, menuViewAll.Click
    'Removes all question filters currently applied and updates
the question list
    If allQuestions.Count > 0 Then
        'If there are saved questions
        lstQuestions.Items.Clear()
        For i = 0 To allQuestions.Count - 1
            'For every question: re-index and add to the list
            allQuestions(i).setQID(i + 1)
            lstQuestions.Items.Add(allQuestions(i).getQID & " " &
vbTab & allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer &
") [" & allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating
& "]"")
        Next
        Call backupAllQuestions()
        lstQuiz.ClearSelected()
        lstQuestions.ClearSelected()
        'Backup the questions and deselect items in both lists
    Else : Exit Sub

```

```

        End If
        Call countLists()
        'Count questions
    End Sub
    Sub refreshQuiz()
        'Clears and refreshes the quiz list when it has changed, in
the event of questions being added, deleted or edited
        lstQuiz.Items.Clear()
        For i = 0 To newQuiz.Count - 1
            lstQuiz.Items.Add(newQuiz(i).getQText & " (" &
newQuiz(i).getQAnswer & ")")
        Next
        Call countLists()
        'Count questions
    End Sub
    Sub countLists()
        'Updates the count toolbar at the bottom of the form by
counting the question and quiz lists
        questionStatusLabel.Text = "Viewing " &
lstQuestions.Items.Count & " questions"
        quizStatusLabel.Text = "- Quiz contains " &
lstQuiz.Items.Count & " questions"
    End Sub

    Private Sub addNewQuestion(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnNewQuestion.Click,
menuNewQuestion.Click
        'Resets fields on the form and opens it as a dialog
        frmAddEditQuestion.clearQuestion()
        frmAddEditQuestion.ShowDialog()
    End Sub
    Sub addNewQuestion(ByVal newquestion As Question)
        'Add the passed question to the allQuestions collection
        allQuestions.Add(newquestion)
        Call refreshList()
    End Sub
    Private Sub editQuestion(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEditQuestion.Click,
menuEditQuestion.Click
        Try
            If lstQuestions.SelectedIndex > -1 Then
                'If there is a question selected
                'Get the question index from the list text
                Dim line As String =
lstQuestions.SelectedItem.ToString

```

```

        Dim idIndex As Integer = CInt(line.Substring(0,
line.IndexOf(" ")) - 1
        'Edit the question with the index of the selected list
item
        'Pass the question identified from the index to
frmEditQuestion and open the form
        frmAddEditQuestion.editQuestion(allQuestions(idIndex))
        frmAddEditQuestion.ShowDialog()
        'Otherwise prompt the user to select a question
    Else : MsgBox("Please select a question.")
    End If
    Catch
    End Try
End Sub
Sub updateQuestion(ByVal qText, ByVal qtype, ByVal qRating, ByVal
qAns, ByVal qInc1, ByVal qInc2, ByVal qInc3, ByVal qUnit, ByVal
qTopic)
    Dim questionIndex As Integer = lstQuestions.SelectedIndex
    'If editing, the index of the selected question will not have
changed from before frmEditQuestion was opened
    'Update the existing question at the specified index in
allQuestions with the new data passed by frmEditQuestion
    allQuestions(questionIndex).setQText(qText)
    allQuestions(questionIndex).setQType(qtype)
    allQuestions(questionIndex).setQRating(qRating)
    allQuestions(questionIndex).setQAnswer(qAns)
    If allQuestions(questionIndex).getQType = "MC" Then
        allQuestions(questionIndex).setIncorrect(qInc1, 0)
        allQuestions(questionIndex).setIncorrect(qInc2, 1)
        allQuestions(questionIndex).setIncorrect(qInc3, 2)
    End If
    allQuestions(questionIndex).setQUnit(qUnit)
    allQuestions(questionIndex).setQTopic(qTopic)
    'Update the question and quiz lists to display the current
question data
    Call refreshList()
    Call refreshQuiz()
End Sub
Sub deleteQuestion() Handles btnDeleteQuestion.Click,
menuDeleteQuestion.Click
    Dim qDelete As Question
    'Create a new instance of the question class
    Try
        'If there is a question selected
        'Get the question index from the list text
        Dim line As String = lstQuestions.SelectedItem.ToString

```

```

        Dim idIndex As Integer = Cint(line.Substring(0,
line.IndexOf(" ")) - 1
        Dim quizIndex As Integer
        refreshList()
        'Remove all filters from the question list
        If idIndex > -1 Then
            Dim result As MsgBoxResult = MsgBox("Are you sure you
want delete this question?", MsgBoxStyle.YesNo) 'confirm deletion
            'Ask the user to confirm deletion
            If result = MsgBoxResult.Yes Then
                qDelete = allQuestions.Item(idIndex)
                'Assign qDelete the question data from the
selected question in allQuestions
                If alreadyInQuiz(qDelete) Then
                    quizIndex = newQuiz.IndexOf(qDelete)
                    lstQuiz.Items.RemoveAt(quizIndex)
                    newQuiz.Remove(qDelete)
                    'If the question is found in the quiz, remove
it from there as well
                    MsgBox("Question has also been removed from
current quiz.")
                End If
                lstQuestions.Items.RemoveAt(idIndex)
                allQuestions.Remove(qDelete)
                'Remove the question from the allquestions
collection
                Call refreshList()
            End If
        Else : MsgBox("Please select a question.")
        End If
    Catch
    End Try
End Sub

Private Sub searchQuestions(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSearch.Click, menuSearch.Click
    'Load the search form
    frmTextSearch.ShowDialog()
End Sub
Sub searchQuestions(ByVal searchTerm As String)
    Dim found As Boolean
    lstQuestions.Items.Clear()
    'Clear the question list
    For i = 0 To allQuestions.Count - 1
        'Execute for the index of every question in the system

```

```

        If
        LCase(allQuestions.ElementAt(i).getQText).contains(LCase(searchTerm))
        Then
            found = True
            'Add each matching question to the question list
            lstQuestions.Items.Add(allQuestions(i).getQID & " " &
vbTab & allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer &
") [" & allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating
& "]"")
            End If
        Next
        Call countLists()
        'Counts the search results
        If found = False Then MsgBox("No questions found.")
        'Notify the user if no questions matched the search term
    End Sub
    Private Sub filterUnitTopic(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles menuFilterUnitTopic.Click
        'Show filter form
        frmFilterUnitTopic.ShowDialog()
    End Sub
    Sub filterUnitTopic(ByVal treeIndex, ByVal nodeTitle)
        If treeIndex = 1 Then
            'Indexes of 1 indicate the node was a unit
            lstQuestions.Items.Clear()
            'Clear the question list
            For i = 0 To allQuestions.Count - 1
                'Search every question in the system for matching
units
                If allQuestions.ElementAt(i).getQUnit = nodeTitle Then
                    lstQuestions.Items.Add(allQuestions(i).getQID & "
" & vbTab & allQuestions(i).getQText & " (" &
allQuestions(i).getQAnswer & ") [" & allQuestions(i).getQTopic & ", "
& allQuestions(i).getQRating & "]"")
                End If
            Next
            Call countLists()
        ElseIf treeIndex = 2 Then
            'Indexes of 2 indicate the node was a topic
            lstQuestions.Items.Clear()
            For i = 0 To allQuestions.Count - 1
                'Search every question in the system for matching
topics
                If allQuestions.ElementAt(i).getQTopic = nodeTitle
Then

```

```

        lstQuestions.Items.Add(allQuestions(i).getQID & "
" & vbTab & allQuestions(i).getQText & " (" &
allQuestions(i).getQAnswer & ") [" & allQuestions(i).getQTopic & ", "
& allQuestions(i).getQRating & "]"")
        End If
    Next
    Call countLists()
    'Count the search results
End If
End Sub
Private Sub filterDifficulty(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles menuFilterDifficulty.Click
    'Show filter form
    frmFilterDifficulty.ShowDialog()
End Sub
Sub filterDifficulty(ByVal rating)
    lstQuestions.Items.Clear()
    'Clear the question list
    For i = 0 To allQuestions.Count - 1
        If allQuestions.ElementAt(i).getQRating = rating Then
            'Search every question in the system for matching
difficulty ratings
            lstQuestions.Items.Add(allQuestions(i).getQID & " " &
vbTab & allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer &
") [" & allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating
& "]"")
        End If
    Next
    Call countLists()
    'Count the search results
End Sub
Private Sub filterType(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles menuFilterType.Click
    'Show filter form
    frmFilterType.ShowDialog()
End Sub
Sub filterType(ByVal type)
    lstQuestions.Items.Clear()
    'Clear the question list
    For i = 0 To allQuestions.Count - 1
        If allQuestions(i).getQType = type Then
            'Search every question in the system for matching
question types
            lstQuestions.Items.Add(allQuestions(i).getQID & " " &
vbTab & allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer &

```

```

") [" & allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating
& "]"")
    End If
Next
Call countLists()
End Sub
Private Sub alphabetSort(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SortQuestionsAlphabetically.Click
    If allQuestions.Count > 0 Then
        'Unless there are questions in the system, the algorithm
won't run
        Dim questionList(allQuestions.Count - 1, 1) As String
        'Creates a 2-D array the same size as the allquestions
system collection
        For i = 0 To allQuestions.Count - 1
            questionList(i, 0) = allQuestions(i).getQText
            questionList(i, 1) = allQuestions(i).getQID & " " &
vbTab & allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer &
") [" & allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating
& "]"
        Next
        'Populates the first row of the array with the questions
from each question in the allquestions collection
        'Populates the second row of the array with the
corresponding 'list view' of each question
        Dim tempData(1) As String
        'Creates an array with two empty elements
        Dim switchValues As Boolean = True
        'Creates a boolean which will determine whether adjacent
elements are swapped
        While switchValues = True
            'Loops until all elements of allquestions have been
compared and the last two do not need to be switched
            switchValues = False
            For i = 0 To allQuestions.Count - 2
                If questionList(i, 0) > questionList(i + 1, 0)
Then
                    'The lower question text comes alphabetically
after the higher one

                    switchValues = True
                    tempData(0) = questionList(i, 0)
                    tempData(1) = questionList(i, 1)
                    'Temporarily hold the lower column of the
array in the temporary array
                    questionList(i, 0) = questionList(i + 1, 0)
                    questionList(i, 1) = questionList(i + 1, 1)

```

```

        'Move the higher column of the array to the
adjacent lower index
        questionList(i + 1, 0) = tempData(0)
        questionList(i + 1, 1) = tempData(1)
        'Move the data in the temporary array to the
higher index, completing the switch
        End If
    Next
End While
lstQuestions.Items.Clear()
'Clear the question list
For i = 0 To allQuestions.Count - 1
    'For every element in the allquestions collection
    'Add the list display of the alphabetised questions to
the question list
    lstQuestions.Items.Add(questionList(i, 1))
Next
Else
End If
End Sub

Private Sub viewClassLogs(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnClassLogs.Click, menuLogs.Click
    'Show class logs form
    frmClassLogs.ShowDialog()
End Sub
Private Sub addNewClass(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles menuNewClass.Click
    'Show the class logs form and open the add class form in front
as a dialog
    frmClassLogs.Show()
    frmAddNewClass.ShowDialog()
End Sub
Private Sub addToQuiz(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAddToQuiz.Click
    'Calls the addToQuiz subroutine when the shortcut is clicked
    Call addToQuiz()
End Sub
Private Sub shortcutAddToQuiz(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.KeyPressEventArgs) Handles
lstQuestions.KeyPress
    If e.KeyChar = Chr(Keys.Return) Then Call addToQuiz()
    'If the user presses return while an element in the question
list is selected, add the element to the quiz
End Sub

```



```

Sub addToQuiz()
    If lstQuestions.SelectedIndex > -1 Then
        'If there is a question selected
        'Get the question index from the list text
        Dim line As String = lstQuestions.SelectedItem.ToString
        Dim questionIndex As Integer = CInt(line.Substring(0,
line.IndexOf(" "))) - 1
        Try
            'Checks if the quiz contains the question before
adding it to the newquiz collection
            If Not alreadyInQuiz(allQuestions(questionIndex)) Then
                newQuiz.Add(allQuestions.Item(questionIndex))
                lstQuiz.Items.Add(newQuiz.Last.getQText & " (" &
newQuiz.Last.getQAnswer & ")")
                Call countLists()
                'Recounts the question and quiz lists
            Else
                MsgBox("Error. This question is already in the
quiz.")
                'Notifies the user that the question has not been
added
            End If
        Catch
        End Try
    Else : MsgBox("Please select a question and try again.")
    End If
End Sub

Function alreadyInQuiz(ByVal x As Question)
    'Checks if the newquiz collection already contains the
question
    If newQuiz.Contains(x) Then
        alreadyInQuiz = True
    Else
        alreadyInQuiz = False
    End If
End Function

Private Sub deleteFromQuiz(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnDeleteFromQuiz.Click
    'Calls the deletefromquiz subroutine when the shortcut is
clicked
    Call deleteFromQuiz()
End Sub

Private Sub shortcutDeleteFromQuiz(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles
lstQuiz.KeyPress
    If e.KeyChar = Chr(Keys.Back) Then Call deleteFromQuiz()

```

```
        'If the user presses backspace while an element in the quiz
list is selected, remove the element from the quiz
    End Sub
    Sub deleteFromQuiz()
        If lstQuiz.SelectedIndex > -1 Then
            'If there is a question selected
            'Get the question's index in the quiz
            Dim quizIndex As Integer = lstQuiz.SelectedIndex
            Dim result As MsgBoxResult = MsgBox("Are you sure you want
remove this question from the quiz?", MsgBoxStyle.YesNo) 'confirm
removal
            'Confirm deletion of the question from the quiz
            If result = MsgBoxResult.Yes Then
                newQuiz.RemoveAt(lstQuiz.SelectedIndex)
                lstQuiz.Items.RemoveAt(lstQuiz.SelectedIndex)
                'Remove the question at the selected index from the
quiz list and the newquiz system collection
                Call countLists()
                'Recount the question and quiz lists
                Try
                    lstQuiz.SelectedIndex = quizIndex - 1
                    'Selects the first element in the quiz
                    'Will throw an exception if newQuiz is empty which
will be caught without notification
                Catch
                End Try
            End If
            Else : MsgBox("Please select a question and try again.")
            End If
        End Sub
        Private Sub clearQuiz(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles menuClearQuiz.Click
            If newQuiz.Count > 0 Then
                'If the newquiz collection is not empty, get user
confirmation with the yes/no dialog
                Dim result As MsgBoxResult = MsgBox("Are you sure you want
to clear the current quiz?", MsgBoxStyle.YesNo)
                If result = MsgBoxResult.Yes Then
                    'Clear the newquiz collection and the quiz list
                    lstQuiz.Items.Clear()
                    newQuiz.Clear()
                End If
                Call countLists()
                'Recount the quiz and question lists
            Else : MsgBox("This quiz doesn't contain any questions.")
            End If
        End Sub
    End Class
```

End Sub

```

Private Sub exportMoodleQuiz(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles menuExport.Click
    If lstQuiz.Items.Count > 0 Then
        'The algorithm will only run if newquiz is not empty
        exportQuiz.FileName = ""
        'Clear the file name of the export form
        exportQuiz.ShowDialog()
        If exportQuiz.FileName <> "" Then
            Dim type As String
            Dim quizWriter As New
System.IO.StreamWriter(exportQuiz.FileName & ".txt")
            'Create a new instance of the streamwriter class
            For Each item In newQuiz
                'Iterate through every question in the newquiz
collection
                    type = item.getQType.ToString
                    'Write the string returned by the
exportMoodleQuestion subroutine, passing the question and its type
                    quizWriter.Write(exportMoodleQuestion(item, type))
                Next
            quizWriter.Close()
            'Close the streamwriter
        End If
    Else
        MsgBox("This quiz doesn't contain any questions.")
    End If
End Sub
Function exportMoodleQuestion(ByVal q As Question, ByVal type As
String)
    If type = "SA" Then
        'Return the Moodle format for short answer questions
        exportMoodleQuestion = q.getQText & " {" & q.getQAnswer &
"}" & vbNewLine & vbNewLine
    ElseIf type = "MC" Then
        'Return the Moodle format for multiple choice questions
        exportMoodleQuestion = q.getQText & "{" & vbNewLine &
            "=" & q.getQAnswer & vbNewLine &
            "~" & q.getIncorrect(0) & vbNewLine &
            "~" & q.getIncorrect(1) & vbNewLine &
            "~" & q.getIncorrect(2) & vbNewLine &
            "}" & vbNewLine & vbNewLine
    ElseIf type = "TF" Then

```

```

'Return the Moodle format for true/false questions
conditional on what the answer is
If q.getQAnswer = "True" Then
    exportMoodleQuestion = q.getQText & " {T}" & vbNewLine
& vbNewLine
Else
    exportMoodleQuestion = q.getQText & " {F}" & vbNewLine
& vbNewLine
End If
ElseIf type = "NU" Then
'Return the Moodle format for numerical questions
exportMoodleQuestion = q.getQText & " {" & q.getQAnswer &
"}" & vbNewLine & vbNewLine
Else
    MsgBox("Error. Please try again.")
End If
End Function
Private Sub exportTextQuiz(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles menuSaveQuiz.Click
    If newQuiz.Count > 0 Then
'Return the Moodle format for numerical questions
exportQuiz.FileName = ""
'Clear the file name of the export form
exportQuiz.ShowDialog()
exportQuiz.AddExtension = True
If exportQuiz.FileName <> "" Then
    Dim type As String
    Dim quizWriter As New
System.IO.StreamWriter(exportQuiz.FileName & ".txt")
    Dim msWriter As New
System.IO.StreamWriter(exportQuiz.FileName & " (mark scheme).txt")
'Create two new instances of the streamwriter class
'One will export the quiz and the other will export
the mark scheme
    Dim r As New Random
'Create a new instance of the random integer class
    Dim x As Integer
    For i = 0 To newQuiz.Count - 1
'For each element of the quiz, return the type
        type = newQuiz(i).getQType
'Generate a random number between 1 and 4 and
assign it to the variable x
        x = r.Next(1, 5)
'The quizwriter streamwriter writes the string
returned by the exportTextQuestion function
'It passes the one based index of the

```

```

        quizWriter.Write(exportTextQuestion(i + 1,
newQuiz(i), type, x))
        'The markscheme streamwriter writes the index of
the question in the quiz, the correct answer, and the difficulty
rating
        msWriter.Write(i + 1 & ") " &
newQuiz(i).getQAnswer & vbTab & "[" & newQuiz(i).getQRating & "]
marks" & vbNewLine & vbNewLine)
        Next
        quizWriter.Close()
        msWriter.Close()
        'Close both streamwriters
    End If
Else : MsgBox("This quiz doesn't contain any questions.")
End If
End Sub
Function exportTextQuestion(ByVal i As Integer, ByVal q As
Question, ByVal qtype As String, ByVal x As Integer)
    If qtype = "SA" Or qtype = "NU" Or qtype = "TF" Then
        'Exports the index of the question, followed by the
question text and the difficulty rating
        exportTextQuestion = i & ") " & q.getQText & "(" &
q.getQRating & " mark[s])" & vbNewLine & vbNewLine
    ElseIf qtype = "MC" Then
        'Uses the random variable passed to determine the order in
which answers are exported
        'Exports the index of the question, followed by the
question text and the difficulty rating and four answers
        If x = 1 Then
            exportTextQuestion = i & ") " & q.getQText & vbTab &
"(" & q.getQRating & " mark[s])" & vbNewLine &
                "a) " & q.getQAnswer & vbNewLine &
                "b) " & q.getIncorrect(0) & vbNewLine &
                "c) " & q.getIncorrect(1) & vbNewLine &
                "d) " & q.getIncorrect(2) & vbNewLine & vbNewLine
        ElseIf x = 2 Then
            exportTextQuestion = i & ") " & q.getQText & vbTab &
"(" & q.getQRating & " mark[s])" & vbNewLine &
                "a) " & q.getIncorrect(0) & vbNewLine &
                "b) " & q.getQAnswer & vbNewLine &
                "c) " & q.getIncorrect(1) & vbNewLine &
                "d) " & q.getIncorrect(2) & vbNewLine & vbNewLine
        ElseIf x = 3 Then
            exportTextQuestion = i & ") " & q.getQText & vbTab &
"(" & q.getQRating & " mark[s])" & vbNewLine &
                "a) " & q.getIncorrect(0) & vbNewLine &

```

```

        "b) " & q.getIncorrect(1) & vbNewLine &
        "c) " & q.getQAnswer & vbNewLine &
        "d) " & q.getIncorrect(2) & vbNewLine & vbNewLine
    Else
        exportTextQuestion = i & ") " & q.getQText & vbTab &
        "(" & q.getQRating & " mark[s])" & vbNewLine &
        "a) " & q.getIncorrect(0) & vbNewLine &
        "b) " & q.getIncorrect(1) & vbNewLine &
        "c) " & q.getIncorrect(2) & vbNewLine &
        "d) " & q.getQAnswer & vbNewLine & vbNewLine
    End If
    Else : MsgBox("Error. Please try again.")
    End If
End Function

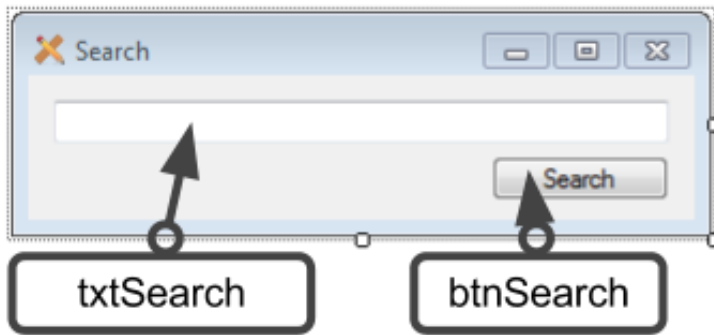
End Class

```

---

frmTextSearch

Handles the input and validation of text to search questions.



```
Public Class frmTextSearch
```

```

    Private Sub startup(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Sets the properties determining whether the user can maximise
or minimise the form
        Me.MinimizeBox = False
        Me.MaximizeBox = False
    End Sub
    Private Sub searchClicked(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnSearch.Click
        'Call the search subroutine
        Call search()
    End Sub

```

```
Private Sub shortcutSearch(ByVal sender As System.Object, ByVal e
As System.Windows.Forms.KeyPressEventArgs) Handles txtSearch.KeyPress
    'If the user presses the return key, call the search
subroutine
    If e.KeyChar = Chr(Keys.Return) Then Call search()
End Sub
Sub search()
    Dim searchTerm As String
    If txtSearch.Text = "" Then
        MsgBox("Please enter a search term.")
    Else : searchTerm = txtSearch.Text
        'If the search field is not blank, call the
searchquestions subroutine on the home form
        frmHome.refreshList()
        frmHome.searchQuestions(searchTerm)
        txtSearch.Clear()
        'Clear the search box
    End If
    txtSearch.Select()
End Sub

End Class
```

---

## Subroutine and Variable Overview

Every subroutine and variable in the system is listed, under its parent form. Where two subroutines in the same form share a name because the private sub handling a form event calls the public sub to perform a task to eliminate repetition of code, the private sub is prefaced with (Private) to differentiate between them. (Nx) before a subroutine name indicates that there are N almost identical subroutines which handle very similar events and have been grouped together.

### frmAddEditQuestion

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form. Loads the contents of the Units text file into the unit combo box.
cancelChanges	Handles the form closing event, resets the 'editing' Boolean so on the event of the form next opening to add rather than edit a question, all fields are blank.
SASelected	Handles the checking of the short answer question type radio button.
NUSelected	Handles the checking of the numerical question type radio button.
TFSelected	Handles the checking of the true/false question type radio button.
MCSelected	Handles the checking of the multiple choice question type radio buttons.
loadTopics	Gets the selected unit title and passes it to findTopics.
findTopics	Opens the file with the unit name passed to it and loads all topics from the file to the topic combo box.
clearQuestion	Resets the form sizing and fields.
(Private) saveQuestion	Handles the save question button click event, validates all the data entered and if valid, calls saveQuestion (see below.)
saveQuestion	Instantiates a new question, saves the question data and calls either addNewQuestion (see frmHome) passing the question to add it to the allQuestions collection (if editing is set to false) or updateQuestion (see frmHome) passing the question to overwrite it in the allQuestions collection (if editing is set to true.)
saveMultipleChoiceAnswers	Saves validated multiple choice answers for new and existing questions.
editQuestion	Sets the 'editing' Boolean to true to indicate that the question will be overwritten when saved, and loads the form with existing question data.



Subroutine	Description
<code>toggleSize</code>	Handles the character panel click event, and expands or contracts the form depending on the current state of the panel, showing or hiding the special characters.
<code>(Private, 25x) btn_Click</code>	Handles the clicking events of the 25 special characters on the form, calls <code>insertCharacter</code> (see below) and passes the symbol to insert it into the question body.
<code>insertCharacter</code>	Inserts the character passed to it into the body of the question.

Variable	Type	Description
<code>questionType</code>	String	Changes based on the selection of question type from the radio buttons. Used to set the question type when editing/saving.
<code>editing</code>	Boolean	Indicates whether the form is adding a new question or editing an existing one. Events conditional upon this being true are loading the form with data rather than blank, and saving to the selected question index rather than to a new question.
<code>userDirectory</code>	String	The file name of the user profile folder in C://Users, used in saving and loading.
<code>allLines()</code>	String	Array storing the contents of the units file as it is read.
<code>unit</code>	String	The unit currently selected in the units combo box.
<code>integerTest</code>	Integer	Part of a try-catch mechanism which attempts to convert an integer in a free text field to an integer data type to test its validity. It is assigned the value of the text field.
<code>currentQuestion</code>	Question	Assigned the values of the text fields containing question data whenever a new question is instantiated, and then saved to the <code>allQuestions</code> collection.

### frmAddLog

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form. Loads the log of whichever class is currently selected in frmClassLogs.
cancelChanges	Handles the form closing event, alerts the user of unsaved changes being lost.
handleReturn	Handles the keypress event for the return key, stops the user using return in the text field.
saveLog	Handles the save button click event, validates the log field and if valid, calls addLog (see frmClassLogs.)

Variable	Type	Description
result	MsgBoxResult	The result from the message box which alerts users that if they cancel they will lose unsaved changes, and asks them if they wish to continue.
log	String	Assigned the contents of the text box where users can enter the class log. Changes every time the form is opened and referencing a different class.

### frmAddNewClass

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form.
saveClass	Handles the save button click event, validates the class name field and if valid, sets the log to the default text and adds the new class to the allClasses collection and the list of classes in frmClassLogs.

Variable	Type	Description
newClass	Group	Assigned the value of the new class name and a blank log whenever a new group is instantiated.

### frmClassLogs

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form. Calls importLogs (see below.)
closeForm	Handles the OK button click event, closes the form.
importLogs	Reads the contents of the logs text file if it exists, and loads data into the allClasses collection of groups. Calls refreshlog (see below.)
backupLogs	Writes the allClasses collection to the logs text file by calling backupLog (see below) for each question. Saves to /UserDirectory/Benchmark
backupLog	Returns the name and log of the class passed to it.
refreshLog	Erases and rewrites the list of classes on the form to update it.
addClass	Handles the add class button click event, loads frmAddClass.
deleteClass	Validates list selection, confirms the deletion of the class and removes the class from allClasses. Calls refreshLog (see above.)
(Private) addLog	Handles the add class button click event. Validates list selection and loads frmAddLog.
addLog	Sets the log of the selected class to the log passed to it by frmAddLog.
logPreview	Handles the index changing on the class list. Loads an unchangeable preview of the log for the selected class.

Variable	Type	Description
allClasses	Collection (of Group)	System collection of every group in the system and their respective logs.
userDirectory	String	The file name of the user profile folder in C://Users, used in saving and loading.
allLines()	String	Array storing the contents of the log backups file as it is read.
savedClass	Group	Assigned the name and log of each class being read back from the backup file before it is added to the allClasses collection.
backup	StreamWriter	StreamWriter used to save class names and logs.
result	MsgBoxResult	The result from the message box which asks users if they are sure they wish to delete a class.
result	MsgBoxResult	The result from the message box which asks users if they are sure they wish to delete a log.

### frmFilterDifficulty

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form. Sets the default filter option.
(5x) Selected	Handles the checking event for each of the five radio buttons. Passes the respective difficulty ratings to filterDifficulty (see frmHome) to filter.
<b>No variables</b>	

### frmFilterType

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form. Sets the default filter option.
(4x) Selected	Handles the checking event for each of the four radio buttons. Passes the respective question types to filterType (see frmHome) to filter.
<b>No variables</b>	

### frmFilterUnitTopic

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form. Loads all units as parent nodes and all topics as child nodes of their respective parents from the unit and topic text files. Collapses all nodes.
nodeChanged	Handles the after select event for the tree, and passes the level and title of the selected index to filterUnitTopic (see frmHome.)

Variable	Type	Description
userDirectory	String	The file name of the user profile folder in C://Users, used in saving and loading.
allLines()	String	Array storing the contents of the units file as it is read.
parentNode	TreeNode	Each unit is temporarily declared as a parentNode while being read from the file and having topic nodes added as children.
topicLines()	String	Array storing the contents of the topics files as they are read.
nodeTitle	String	The title of the node selected.

**frmHome**

Subroutine	Description
startup	Handles the form loading event. Calls setToolTips, importAllQuestions and countLists (see below.) Creates the default unit and topic text files if none exist already.
setToolTips	Sets the tool tips for the icons on the form.
countLists	Counts the elements in allQuestions and newQuiz and displays them in the toolbar at the bottom of the form.
importAllQuestions	Reads the contents of the questions text file if it exists, and loads data into the allQuestions collection of questions. Calls refreshList (see below.)
backupAllQuestions	Overwrites the questions text file by iteratively calling backupQuestion (see below) and writing the returned value to the file.
backupQuestion	Returns all the question data associated with the question passed to it.
refreshList	Handles the refresh button click event and the view all menu click event. Erases and rewrites the question list from allQuestions, recounts them, and calls backupAllQuestions. Calls countLists.
refreshQuiz	Erases and rewrites the quiz list from newQuiz. Calls countLists.
(Private) addNewQuestion	Handles the new question button click event and the new question menu click event. Calls clearQuestion (see frmAddEditQuestion) and loads frmAddEditQuestion.
addNewQuestion	Adds the question passed to it to the allQuestions collection and updates the question list to include it. Calls countLists.
editQuestion	Validates list selection and if valid, loads frmAddEditQuestion, passing it the index of the currently selected question in allQuestions.
updateQuestion	Overwrites the question at the selected index of the question list with the data from the question passed to it by saveQuestion (see frmAddEditQuestion.) Updates the question list, then calls refreshList and refreshQuiz.

Subroutine	Description
deleteQuestion	Handles the delete button click event and the delete menu click event. Validates the list selection, and confirms the deletion of the selected question before deleting it from allQuestions and the question list. Attempts to delete the question from the quiz (if it exists there) and notifies the user if this is successful. Calls refreshList.
(Private) searchQuestions	Handles the search button click event and the search menu click event. Loads frmTextSearch.
searchQuestions	Clears the question list and iterates through allQuestions searching for the term passed to it by frmTextSearch, adding any questions which contain the term to the list and displaying it in its filtered form. If no results are found, it notifies the user. Calls countLists.
(Private) filterUnitTopic	Handles the filter by unit/topic menu click event. Loads frmFilterUnitTopic.
filterUnitTopic	Clears the list and depending on whether the node level passed to it by frmFilterUnitTopic is 1 or 2, iterates through all questions and either adds questions to the list where the unit matches the title (passed 1) or where the topic matches the title (passed 2) to display the question list in its filtered form. Calls countLists.
(Private) filterDifficulty	Handles the filter by difficulty menu click event. Loads frmFilterDifficulty.
filterDifficulty	Clears the question list and iterates through allQuestions, adding questions which have the same difficulty as was passed by frmFilterDifficulty. Calls countLists.
(Private) filterType	Handles the filter by type menu click event. Loads frmFilterType.
filterType	Clears the question list and iterates through allQuestions, adding questions which have the same type as was passed by frmFilterType. Calls countLists.
(Private) alphabetSort	Handles the sort alphabetically menu click event. Populates an array with question data and bubblesorts the question text, adding each item of the sorted alphabetical array to the question list.
viewClassLogs	Handles the class logs button click event and the class logs menu click event. Loads frmClassLogs.

Subroutine	Description
<code>addNewClass</code>	Handles the add new class menu click event. Loads <code>frmAddNewClass</code> .
<code>(Private) addToQuiz</code>	Handles the add to quiz button click event. Calls <code>addToQuiz</code> (see below.)
<code>shortcutAddToQuiz</code>	Handles the keypress event in the question list. If the keypress is the return key, it calls <code>addToQuiz</code> (see below.)
<code>addToQuiz</code>	Validates the list selection, and if valid, checks if the question is already in the quiz by calling <code>alreadyInQuiz</code> (see below.) If the question is in the quiz then the user is notified, if it's not then the selected question is added to the <code>newQuiz</code> collection and the quiz list. Calls <code>countLists</code> .
<code>alreadyInQuiz</code>	Checks for the presence of the question in <code>allQuestions</code> at the index passed to it in <code>newQuiz</code> . If it exists in <code>newQuiz</code> , returns true and otherwise returns false.
<code>(Private) deleteFromQuiz</code>	Handles the delete from quiz button click event. Calls <code>deletefromQuiz</code> (see below.)
<code>shortcutDeleteFromQuiz</code>	Handles the keypress event in the quiz list. If the keypress is the backspace key, it calls <code>deleteFromQuiz</code> (see below.)
<code>deleteFromQuiz</code>	Validates the list selection, and if valid, confirms the deletion of the question from the quiz. Removes the question from the quiz list and from <code>newQuiz</code> .
<code>clearQuiz</code>	Handles the clear quiz menu click event. Validates list selection, and confirms the clearing with the user. Clears both <code>newQuiz</code> and the quiz list. Calls <code>countLists</code> .
<code>exportMoodleQuiz</code>	Validates the presence of questions in <code>newQuiz</code> . Prompts the user for a file name and path, then calls <code>exportMoodleQuestion</code> for each question in <code>newQuiz</code> (passing the question and the question type), writing the returned values to a text file.
<code>exportMoodleQuestion</code>	Depending on the question type passed to it, the function gets question data from the question passed to it and formats it using <code>.gift</code> file formatting.

Subroutine	Description
<code>exportTextQuiz</code>	Validates the presence of questions in <code>newQuiz</code> . Prompts the user for a file name and path, then calls <code>exportTextQuestion</code> for each question in <code>newQuiz</code> (passing the index of the question in <code>newQuiz</code> , the question, the question type, and a random number between one and four), writing the returned values to a text file. For each question iterated through, it writes the same index, and the difficulty rating and answer to a mark scheme text file.
<code>exportTextQuestion</code>	Depending on the question type passed to it, the function either returns the question text and difficulty rating (short answer, numerical or true/false) or uses the random number passed to 'shuffle' the answers, preventing the consistent ordering of the correct answer within the incorrect ones.

Variable	Type	Description
<code>allQuestions</code>	Collection (of Question)	System collection of every question in the system and all associated question data.
<code>newQuiz</code>	Collection (of Question)	System collection of every question in the current quiz and all associated data.
<code>defaultUnitTopicSetup</code>	Streamwriter	Streamwriter used to save unit/topic defaults if there are no unit/topic files.
<code>userDirectory</code>	String	The file name of the user profile folder in C://Users, used in saving and loading.
<code>allLines()</code>	String	Array storing the contents of the question backups file as it is read.
<code>count</code>	Integer	Stepper variable, increments every time a new question is read from the backup file.
<code>savedQuestion</code>	Question	Assigned all of the question data of each question being read back from the backup file before it is added to the <code>allQuestions</code> collection.
<code>backup</code>	StreamWriter	StreamWriter used to save question data.
<code>line</code>	String	The currently selected list item.
<code>idIndex</code>	Integer	The integer found directly before " " in line (see above), which is the selected question's ID.
<code>questionIndex</code>	Integer	The index of the selected element in the question list.



Variable	Type	Description
qDelete	Question	A question being deleted from the question bank.
result	MsgBoxResult	The result from the message box which asks users if they are sure they wish to delete a question.
found	Boolean	Conditional on whether a specified string has been found within a question being searched.
questionList(x, 1)	(2D Array) String	(Where x = allQuestions.count) Temporarily stores question text and all other question data from every question in allQuestions while it is alphabetically sorted.
tempData(1)	(1D Array) string	Holds the data being swapped in the sorting process.
switchValues	Boolean	Conditional on whether one array element holds a greater value than its subsequent element, determines whether the 'swap' takes place.
quizIndex	Integer	The index of the selected element in the quiz.
result	MsgBoxResult	The result from the message box which asks users if they are sure they wish to remove a question from the quiz.
result	MsgBoxResult	The result from the message box which asks users if they are sure they wish to clear the quiz.
type	String	Question type of the question currently being exported.
quizWriter	StreamWriter	StreamWriter used to export quizzes.
msWriter	StreamWriter	StreamWriter used to export mark schemes.
r	Random	Used to generate a series of random numbers for ordering multiple choice answers.
x	Integer	The random number values generated by r (see above.)

**frmTextSearch**

Subroutine	Description
startup	Handles the form loading event, and the sizing options of the form.
searchClicked	Handles the search button click event.
shortcutSearch	Handles the keypress event. If the keypress is the return key, it calls search (see below.)
Search	Validates the presence of a search term.

Variable	Type	Description
searchTerm	String	The string entered by the user which will be searched for.

## Detailed Algorithm Design

<b>Algorithm:</b> Question Backup	<b>Location:</b> frmHome, backupAllQuestions()
<b>Description</b>	
The algorithm gets the file path of the user directory and writes/overwrites a question file, calling the backupQuestion function for each question in the question bank. backupQuestion returns question data over nine consecutive lines.	
<b>Pseudo-code</b>	
<pre> Var backup ← File(user\Benchmark\Questions.txt) FOR i ← 0 TO allQuestions.count - 1     WRITELINE(backup, backupQuestion(Questions(i))) NEXT  FUNCTION backupQuestion(question)     backupQuestion = question.text &amp; newline &amp; question.type &amp;     newline &amp; question.answer &amp; newline &amp; question.incorrect[0]     &amp; newline &amp; question.incorrect[1] &amp; newline &amp;     question.incorrect[2] &amp; newline &amp; question.unit &amp; newline &amp;     question.topic &amp; newline &amp; question.rating END FUNCTION </pre>	
<b>Code</b>	
<pre> Dim backup As New System.IO.StreamWriter(userDirectory &amp; "\Benchmark\Questions.txt") For i = 0 To allQuestions.Count - 1     backup.Write(backupQuestion(allQuestions(i))) Next backup.Close()  Function backupQuestion(ByVal q As Question)     backupQuestion = q.getQText &amp; vbNewLine &amp; q.getQType &amp; vbNewLine &amp;     q.getQAnswer &amp; vbNewLine &amp; q.getIncorrect(0) &amp; vbNewLine &amp;     q.getIncorrect(1) &amp; vbNewLine &amp; q.getIncorrect(2) &amp; vbNewLine &amp;     q.getQUnit &amp; vbNewLine &amp; q.getQTopic &amp; vbNewLine &amp; q.getQRating &amp;     vbNewLine End Function </pre>	

<b>Algorithm:</b> Deleting a question	<b>Location:</b> frmHome, deleteQuestion()
<b>Description</b>	
The algorithm gets the question ID of the selected list item, and after checking it is valid (i.e. That there is an item selected) it asks the user to confirm deletion. It then checks if the question is in the quiz, and removes it if it is. Finally, it removes it from the allQuestions collection and updates the quiz and question lists.	
<b>Pseudo-code</b>	
<pre> Var idIndex ← STR(quiz.item.index) </pre>	

```

IF idInxex > 0 THEN
    OUTPUT("Are you sure you want to delete this question")
    IF answer = Yes THEN
        Var qDelete ← allQuestions(idIndex)
        IF alreadyInQuiz(qDelete) THEN
            Var quizIndex ← newQuiz(qDelete.index)
            quiz.remove(quizIndex)
            OUTPUT("Question has been removed.")
        ENDIF
        newQuiz.remove(qDelete)
        quiz.items.remove(idIndex)
        allQuestions.remove(qDelete)
        Refreshlist()
    ENDIF
ELSE OUTPUT("Error. No question selected.")
ENDIF

```

**Code**

```

Dim qDelete As Question
Dim line As String = lstQuestions.SelectedItem.ToString
Dim idIndex As Integer = CInt(line.Substring(0, line.IndexOf(" "))) - 1
Dim quizIndex As Integer
If idIndex > -1 Then
    Dim result As MsgBoxResult = MsgBox("Are you sure you want delete
this question?", MsgBoxStyle.YesNo)
    If result = MsgBoxResult.Yes Then
        qDelete = allQuestions.Item(idIndex)
        If alreadyInQuiz(qDelete) Then
            quizIndex = newQuiz.IndexOf(qDelete)
            lstQuiz.Items.RemoveAt(quizIndex)
            MsgBox("Question has also been removed from current quiz.")
        End If
        newQuiz.Remove(qDelete)
        lstQuestions.Items.RemoveAt(idIndex)
        allQuestions.Remove(qDelete)
        Call refreshList()
    End If
Else : MsgBox("Error. No question selected.")
End If

```

<b>Algorithm:</b> Filtering by unit/topic	<b>Location:</b> frmFilterUnitTopic & frmHome, filterUnitTopic()
<b>Description</b> This algorithm is run every time the user selects a node in the unit/topic tree. The index of the selection (units have index = 0, topics have index = 1) and the title of the node is passed to filterUnitTopic, where it is checked against all questions in the collection and any matches are added to the question list.	
<b>Pseudo-code</b> <pre> Var treeIndex ← tree.node.level + 1 Var nodeTitle ← tree.node.text filterUnitTopic(treeIndex, nodeTitle)  PROCEDURE filterUnitTopic(treeIndex, nodeTitle)   IF treeIndex = 1 THEN     FOR i ← 0 TO LEN(allQuestions) - 1       list.items.clear       IF allQuestions(i).unit = nodeTitle THEN         list.items.add(allQuestions(i).qID &amp; " " &amp;           newline &amp; allQuestions(i).text &amp; " (" &amp;           allQuestions(i).answer &amp; " ) [" &amp;           allQuestions(i).topic &amp; ", " &amp;           allQuestions(i).rating &amp; "]"")       ENDIF     NEXT     countLists()   ELSEIF treeIndex = 2     list.items.clear     FOR i ← 0 TO LEN(allQuestions) - 1       IF allQuestions(i).topic = nodeTitle THEN         list.items.add(allQuestions(i).qID &amp; " " &amp;           newline &amp; allQuestions(i).text &amp; " (" &amp;           allQuestions(i).answer &amp; " ) [" &amp;           allQuestions(i).topic &amp; ", " &amp;           allQuestions(i).rating &amp; "]"")       ENDIF     NEXT     countLists()   END IF END PROCEDURE </pre>	

**Code**

```
Dim treeIndex As Integer = unitTopicTree.SelectedNode.Level + 1
Dim nodeTitle As String = unitTopicTree.SelectedNode.Text
frmHome.filterUnitTopic(treeIndex, nodeTitle)

Sub filterUnitTopic(ByVal treeIndex, ByVal nodeTitle)
    If treeIndex = 1 Then
        lstQuestions.Items.Clear()
        For i = 0 To allQuestions.Count - 1
            If allQuestions.ElementAt(i).getQUnit = nodeTitle Then
                lstQuestions.Items.Add(allQuestions(i).getQID & " " & vbTab
                    & allQuestions(i).getQText & " (" &
                    allQuestions(i).getQAnswer & ") [" &
                    allQuestions(i).getQTopic & ", " &
                    allQuestions(i).getQRating & "]")
            End If
        Next
        Call countLists()
    ElseIf treeIndex = 2 Then
        lstQuestions.Items.Clear()
        For i = 0 To allQuestions.Count - 1
            If allQuestions.ElementAt(i).getQTopic = nodeTitle Then
                lstQuestions.Items.Add(allQuestions(i).getQID & " " & vbTab
                    & allQuestions(i).getQText & " (" &
                    allQuestions(i).getQAnswer & ") [" &
                    allQuestions(i).getQTopic & ", " &
                    allQuestions(i).getQRating & "]")
            End If
        Next
        Call countLists()
    End If
End Sub
```

<b>Algorithm:</b> Shortcut for adding questions to the quiz	<b>Location:</b> frmHome, shortcutAddToQuiz()
<b>Description</b>	
If the question list has focus and there is a question selected, hitting the return key will call the addToQuiz function. This algorithm recognises the input. (Almost identical to the shortcutDeleteFromQuiz function)	
<b>Pseudo-code</b>	
<pre> Var e ← keyPressEventArgument IF INT(e.value) = 13 THEN     AddToQuiz() ENDIF </pre>	
<b>Code</b>	
<pre> Private Sub shortcutAddToQuiz(ByVal sender As System.Object, ByVal e As     System.Windows.Forms.KeyPressEventArgs) Handles lstQuestions.KeyPress      If e.KeyChar = Chr(Keys.Return) Then Call addToQuiz() End Sub </pre>	

<b>Algorithm:</b> Alphabetically sorting questions	<b>Location:</b> frmHome, alphabetSort()
<b>Description</b>	
This algorithm fills a 2-dimensional array with data from allQuestions temporarily to sort them. In the first column is the string being sorted, the question text of each question. In the second is the corresponding other question data in the format in which it would be added to the question list. The array elements are then bubblesorted, comparing each element in the array with the adjacent one below. If (i+1) is larger than (i), the two elements and their corresponding data switch positions in the array.	
<b>Pseudo-code</b>	
<pre> Var questionList ← [LEN(allQuestions) - 1, 1] FOR i ← 0 to LEN(allQuestions) - 1     questionList[i, 0] = allQuestions(i).text     questionList[i, 1] = allQuestions(i).qID &amp; " " &amp; newline &amp;         allQuestions(i).text &amp; " (" &amp; allQuestions(i).answer &amp; " )         [" &amp; allQuestions(i).topic &amp; ", " &amp; allQuestions(i).rating         &amp; "]" NEXT Var tempData[1] Var switchValues ← True WHILE switchValues = True     switchValues = False     FOR i ← 1 to LEN(allQuestions) - 2         IF questionList[i, 0] &gt; questionList [i+1, 0] THEN </pre>	

```

        switchValues = True
        tempData[0] = questionList[i, 0]
        tempData[1] = questionList[i, 1]
        questionList[i, 0] = questionList[i+1, 0]
        questionList[i, 1] = questionList[i+1, 1]
        questionList[i+1, 0] = tempData[0]
        questionList[i+1, 1] = tempData[1]
    ENDIF
NEXT
END WHILE

```

**Code**

```

Dim questionList(allQuestions.Count - 1, 1) As String
For i = 0 To allQuestions.Count - 1
    questionList(i, 0) = allQuestions(i).getQText
    questionList(i, 1) = allQuestions(i).getQID & " " & vbTab &
        allQuestions(i).getQText & " (" & allQuestions(i).getQAnswer & ") [" &
        allQuestions(i).getQTopic & ", " & allQuestions(i).getQRating & "]"
Next
Dim tempData(1) As String
Dim switchValues As Boolean = True
While switchValues = True
    switchValues = False
    For i = 0 To allQuestions.Count - 2
        If questionList(i, 0) > questionList(i + 1, 0) Then
            switchValues = True
            tempData(0) = questionList(i, 0)
            tempData(1) = questionList(i, 1)
            questionList(i, 0) = questionList(i + 1, 0)
            questionList(i, 1) = questionList(i + 1, 1)
            questionList(i + 1, 0) = tempData(0)
            questionList(i + 1, 1) = tempData(1)
        End If
    Next
End While

```



<p><b>Algorithm:</b> Loading units and topics into the tree from which questions are filtered</p>	<p><b>Location:</b> frmFilterUnitTopic, startup()</p>
<p><b>Description</b></p> <p>The tree is first cleared of all nodes, and then the units text file is opened. For every unit (line in the file) read, it is first added to the tree as a parent node, and then the text file associated with that unit is opened and all topics (lines in the topic file) are added as child nodes. Finally, all nodes are collapsed.</p>	
<p><b>Illustrative Diagram</b></p> <p>The diagram illustrates the process of loading units and topics into a tree structure. It shows a vertical list of units (Unit 1, Unit 2, Unit 3) on the left, with indices <math>i=0</math>, <math>i=1</math>, and <math>i=2</math> next to them. Arrows point from each unit to its corresponding topic file (Unit 1.txt, Unit 2.txt, Unit 3.txt). Each topic file contains a list of topics (e.g., Topic 1.1, Topic 1.2, etc.). On the right, indices <math>j=0</math>, <math>j=1</math>, <math>j=2</math>, and <math>j=3</math> are shown next to the topics. Below the diagram, a tree view shows the resulting nodes and their corresponding code:</p> <pre> + ..... Unit 1  unitTopicTree.Nodes.Add(parentNode)   ..... Topic 1.1  parentNode.Nodes.Add(node)   ..... Topic 1.2  parentNode.Nodes.Add(node)   ..... Topic 1.3  parentNode.Nodes.Add(node)   ..... Topic 1.4  parentNode.Nodes.Add(node)  + ..... Unit 2  unitTopicTree.Nodes.Add(parentNode)   ..... Topic 2.1  parentNode.Nodes.Add(node)   ..... Topic 2.2  parentNode.Nodes.Add(node)   ..... Topic 2.3  parentNode.Nodes.Add(node)   ..... Topic 2.4  parentNode.Nodes.Add(node) </pre>	

**Code**

```
unitTopicTree.Nodes.Clear()
Dim userDirectory As String =
Environment.GetEnvironmentVariable("userprofile")
If System.IO.File.Exists(userDirectory & "\Benchmark\Units.txt") Then
    Dim allLines() As String = System.IO.File.ReadAllLines(userDirectory &
"\Benchmark\Units.txt")
    If allLines.Length > 0 Then
        For i = 0 To allLines.Length - 1
            Dim parentNode As TreeNode =
unitTopicTree.Nodes.Add(allLines(i))
            Dim topicLines() As String =
System.IO.File.ReadAllLines(userDirectory & "\Benchmark\" &
allLines(i) & ".txt")
            If topicLines.Length > 0 Then
                For j = 0 To topicLines.Length - 1
                    parentNode.Nodes.Add(topicLines(j))
                Next
            End If
        Next
        unitTopicTree.Update()
    End If
Else : unitTopicTree.Nodes.Add("Unknown unit")
End If
unitTopicTree.CollapseAll()
```

<p><b>Algorithm:</b> Exporting quizzes in the Moodle GIFT format</p>	<p><b>Location:</b> frmHome, exportMoodleQuiz()</p>
<p><b>Description</b></p> <p>If the quiz contains more than zero questions, a save dialog is opened and the user is prompted to select a file location and name the export file. A StreamWriter is opened and for each item in the quiz, the exportMoodleQuestion function is called to return the relevant data in the GIFT format in order for it to be written to the text file. After all questions have been written, the StreamWriter is closed.</p>	
<p><b>Illustrative Diagram</b></p> <pre>     graph TD       subgraph newQuizList [newQuiz]         direction TB         n0["i=0 newQuiz(0)"]         n1["i=1 newQuiz(1)"]         n2["i=2 newQuiz(2)"]         ndots["... newQuiz(...)"]       end        n0 -.-&gt; "newQuiz(0) \"SA\""  B1["The body of the question {=The correct answer}"]       n1 -.-&gt; "newQuiz(1) \"NU\""  B2["The body of the question {#The correct answer}"]       n2 -.-&gt; "newQuiz(2) \"MC\""  B3["The body of the question { =The correct answer ~An incorrect answer ~An incorrect answer ~An incorrect answer }"]       ndots -.-&gt; "newQuiz(...) \"TF\""  B4["A statement {T} OR {F}"]        B3 --&gt; qw["quizWriter"]       B4 --&gt; qw        subgraph qwOutput [quizWriter]         direction TB         qw1["..."]         qw2["..."]         qw3["The body of the question { =The correct answer ~An incorrect answer ~An incorrect answer ~An incorrect answer }"]         qw4["..."]       end   </pre>	
<p><b>Code</b></p> <pre> If lstQuiz.Items.Count &gt; 0 Then 'there are questions in the quiz   exportQuiz.FileName = ""   exportQuiz.ShowDialog()   If exportQuiz.FileName &lt;&gt; "" Then     exportQuiz.FileName = exportQuiz.FileName &amp; ".txt"     Dim type As String     Dim quizWriter As New   </pre>	

```

        System.IO.StreamWriter(exportQuiz.FileName)
    For Each item In newQuiz
        type = item.getQType.ToString
        quizWriter.Write(exportMoodleQuestion(item, type))
    Next
    quizWriter.Close()
End If
Else
    MsgBox("This quiz doesn't contain any questions.")
End If

Function exportMoodleQuestion(ByVal q As Question, ByVal type As
String)
    If type = "SA" Then
        exportMoodleQuestion = q.getQText & " {" & q.getQAnswer &
        "}" & vbNewLine & vbNewLine
    ElseIf type = "MC" Then
        exportMoodleQuestion = q.getQText & "{" & vbNewLine &
        "=" & q.getQAnswer & vbNewLine &
        "~" & q.getIncorrect(0) & vbNewLine &
        "~" & q.getIncorrect(1) & vbNewLine &
        "~" & q.getIncorrect(2) & vbNewLine &
        "}" & vbNewLine & vbNewLine
    ElseIf type = "TF" Then
        If q.getQAnswer = "True" Then
            exportMoodleQuestion = q.getQText & "{T}" & vbNewLine
            & vbNewLine
        Else
            exportMoodleQuestion = q.getQText & "{F}" & vbNewLine
            & vbNewLine
        End If
    ElseIf type = "NU" Then
        exportMoodleQuestion = q.getQText & {"#" & q.getQAnswer &
        "#"} & vbNewLine & vbNewLine
    Else
        MsgBox("Error. Please try again.")
    End If
End Function

```

## Post-Implementation System Overview

### IOPS Chart

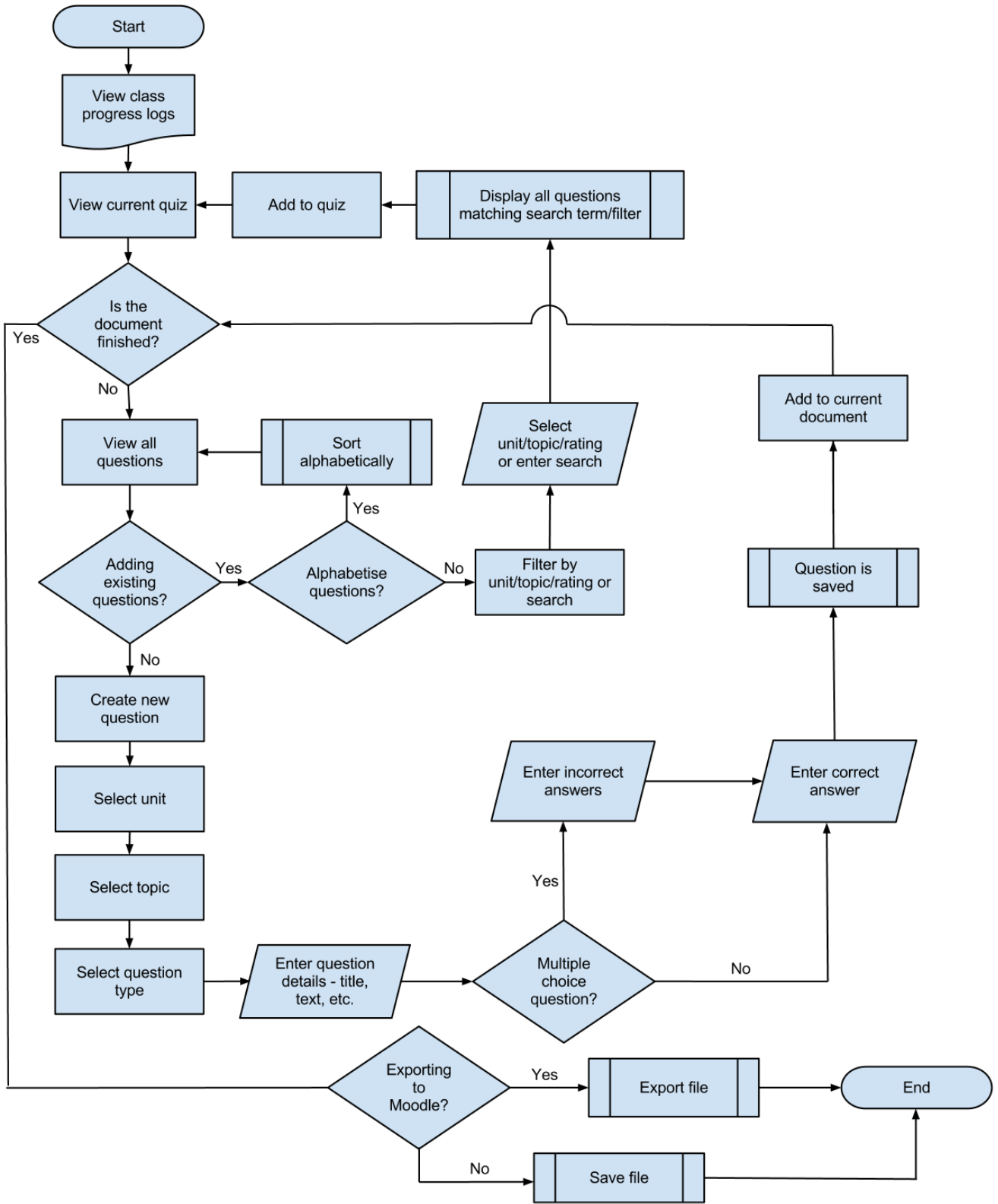
The following chart outlines the input, output, process and storage components of the new system in the most concise way. This can be cross-referenced with the original IOPS design on page 32.

Data flow remains the same as in design, which can be seen on page 34.

<p style="text-align: center;"><i><b>Input</b></i></p> <ul style="list-style-type: none"> <li>• Class record (recent topic)</li> <li>• Class name</li> <li>• Unit of work</li> <li>• Topic</li> <li>• Question type</li> <li>• Question text</li> <li>• Correct answer</li> <li>• Incorrect answers (if multiple choice)</li> <li>• File name (when saving)</li> </ul>	<p style="text-align: center;"><i><b>Process</b></i></p> <ul style="list-style-type: none"> <li>• Retrieve and display questions from text file</li> <li>• Save questions to text file</li> <li>• Save question documents</li> <li>• Convert questions to Moodle .gift format</li> <li>• Sort questions alphabetically</li> <li>• Search for strings within questions</li> <li>• Filter questions by unit, topic etc.</li> </ul>
<p style="text-align: center;"><i><b>Storage</b></i></p> <ul style="list-style-type: none"> <li>• Question text-file</li> <li>• Class log text-file</li> <li>• Units text file</li> <li>• Individual topic text files</li> </ul>	<p style="text-align: center;"><i><b>Output</b></i></p> <ul style="list-style-type: none"> <li>• Moodle GIFT file</li> <li>• Digital question document</li> </ul>

### System Flowchart

On the following page is an update of the system flowchart from design, which can be seen on page 32. Processes such as searching and filtering have been included in more depth, and features added at the end of design such as alphabetising questions have been included.



# User Guide

---





## Installation Guidelines

## Installation from a USB Flash-Drive





## Using the System

### Creating Questions



## Deleting Questions

## Adding and Removing Questions from the Quiz



## Adding and Deleting Classes

## Adding and Deleting Class Logs

## Searching Questions

## Filtering Questions by Unit, Topic or Difficulty



## Exporting Quizzes

## Uploading Quizzes to Moodle





## Troubleshooting



# Appraisal

## Comparison of Project Performance Against Objectives

The following table compares the performance of the completed system to the objectives originally laid out on page 28, to assess whether all the user’s needs have been met.

Key:

Objective not met	Objective partially met	Objective met	Objective exceeded
-------------------	-------------------------	---------------	--------------------

Original Objective	Completed System
Question, answer, unit, topics and type should be mandatory for each [question entered].	Each question in the system stores question text, a correct answer, a unit, a topic, and a question type. The idea of entering general more specific topics was remodelled during implementation to allow for more time-saving question entry.
For multiple choice questions, the incorrect answers should also be stored.	Multiple choice questions store three incorrect answers. However, other question types still have incorrect answer properties which are simply not used.
There should be a default difficulty rating for each question (out of five) which can be edited or left as the default.	The default rating which the form loads with is 1 (out of five.) Other ratings can be selected from the drop-down combo box.
There should be different question types available similar to the types that feature in the exams – short answer, calculation, etc.	Questions can be one of four types; short answer, multiple choice, true/false, or numerical answer.
The questions must be searchable and displayable by question type, unit, topic, and difficulty rating.	The user can search for questions by selecting a question type, unit, topic or difficulty rating. Questions can also be alphabetised.
Users must be able to search all questions for a specific string.	Users can search all questions for a specific string and all questions containing that string are returned.
Searching questions should involve the minimum free text entry to save time and minimise errors. This could be implemented using radio buttons or drop-down menus for selection.	To search by question type or difficulty rating, the user selects their choice using radio buttons. To choose a unit or topic, they choose a node on a tree.

Original Objective	Completed System
The user must be able to edit existing questions or delete them from the database.	Every detail of a question can be edited, and questions can be deleted. If a question is deleted from the system, it is also deleted from the quiz if present there too.
The user should be able to preview the answer to each question that is displayed.	The (correct) answer to each question is displayed in brackets after each question in the question list, but there is no way to hide it.
The user must be able to add records of new classes to the system.	New classes can be added to the system and logs can be added to those classes.
Each class stored must have its own 'log' which would allow free text entry similar to a teacher's diary. The user must be able to record recently set questions from the textbook or chapters covered and retrieve this information when assigning new questions.	Users have free text entry (excluding use of the return key) of up to 500 characters to store information on the current progress of each class. These can be accessed at any time.
The user must be able to delete classes or class logs from the system.	Both logs and classes can be deleted from the system.
The user should be able to create a quiz by selecting questions from the database and/or adding new questions.	The user can add new questions or select existing questions and add them to a quiz.
The system must notify the user if they are adding a question to a quiz which already contains it to prevent unnecessary duplicates.	The system notifies the user with a dialog box if they are adding a duplicate question to the quiz.
The user should be able to export text-based quizzes as text files to be emailed or printed.	The system can export text-based quizzes, but cannot email or print them from within the system.
Text-based quizzes should have separate numbered mark schemes unlike the integrated Moodle quizzes.	Text based quizzes are saved with a numbered mark scheme of the same name suffixed by (Mark scheme) in the same directory, which also contains the marks each question carries.
The system should be able to export quizzes with integrated answers in a format and the UTF-8 encoding recognised by Moodle.	The system exports GIFT files which can be uploaded directly to Moodle.
Users should be able to see at all times a 'count' of how many questions they are viewing and how many are in the current quiz.	The status strip at the bottom of the home screen shows a live count of how many questions are in the system, and how many are in the current quiz.
The system must include button shortcuts for common actions such as adding and removing questions to or from quizzes in order to reduce the time spent navigating menus.	The system contains shortcuts for refreshing any searches, adding and deleting questions, editing questions, viewing the class logs form, searching questions, and adding/removing questions to/from quizzes.

## Potential for Future Developments

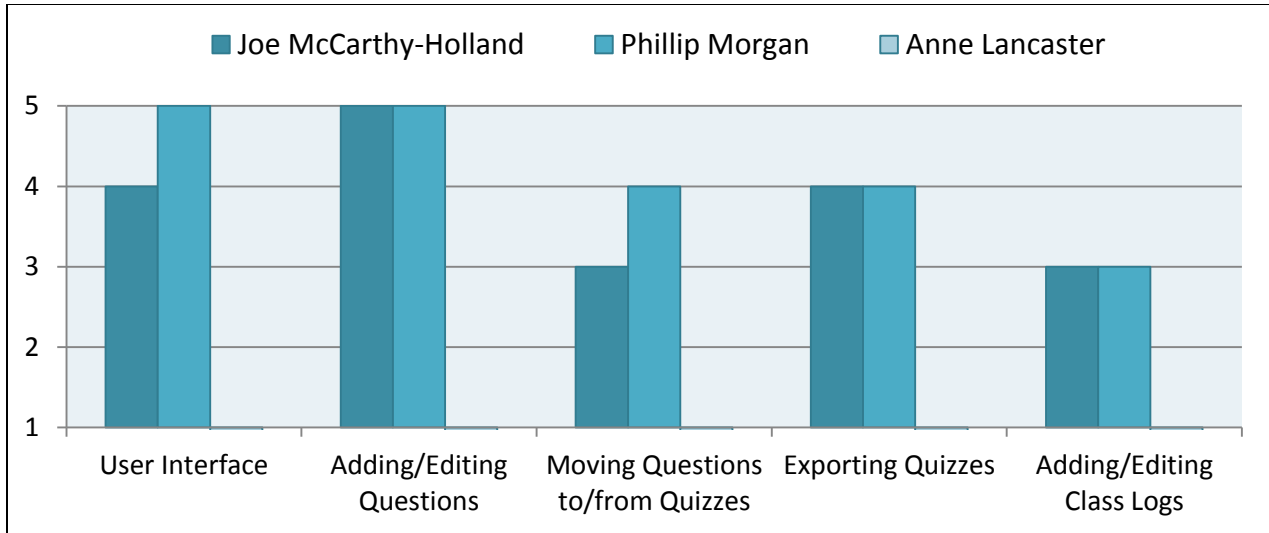
As a default, the units and topics read into the system are those from the AQA Physics A specification (AS - 2009 onwards; A2 - 2010 onwards.) As the specification is subject to change, it follows that the unit and topic data read by the system will change as well. At present, new units and topics can be added by adding the unit to C:\\Users\\UserProfile\\Benchmark\\Units.txt on its own line, and creating a UTF-8 encoded text file with the same name as the unit, containing a list of its topics on separate lines. This can all be done in a simple text editor such as Notepad. However, if existing units or topics are deleted, questions from those topics will not be searchable or editable. A potential development would include a separate category for old-specification units or topics, so they are still searchable but can be hidden when not required.

Another development which could be implemented is the inclusion of images or diagrams in questions, which Moodle can process. This feature would be more complex than any of the question features currently in the system and I'm limited by my time-frame and own ability, and as such unable to include it. For someone maintaining the system with more time available however, this would be possible, and allow an even wider range of questions to be added as diagrams are fairly common in A-Level physics exams.

If teachers wished to import questions from old Moodle quizzes directly into the system, this feature could also be implemented with use of existing classes, subroutines and functions. However, in the original interview this was never considered a desirable feature so at the time of design, it wasn't necessary to add to the system.

## Analysis of User Feedback

I gave the installation files, user guide and a feedback questionnaire to each of the three teachers in the physics department. In order to get an objective assessment of how the users rated the system after the initial use, I asked them to rate the interface, adding/removing questions process, moving questions process, exporting process, and class logs system out of five (five being the highest.) I then asked for additional feedback on any aspect of the system, and have summarised their thoughts below.



Desirable features/changes		
Joe	Phillip	Anne
<ul style="list-style-type: none"> <li>• Multiple question selection</li> <li>• Not allowing multiple classes with the same name</li> <li>• Labeled icons rather than tooltips</li> <li>• Cumulative filtering rather than resetting the filter each time the user searches</li> </ul>	<ul style="list-style-type: none"> <li>• More of an explanation of quizzes in the user manual</li> <li>• Linking class lists with Moodle class lists on Godalming Online</li> </ul>	

Changes such as switching tooltips for labels, class validation to ensure groups aren't added twice, and cumulative filtering can be implemented as part of the next release of Benchmark. These would be the objectives for the second pass through the systems development life cycle.

# Benchmark

## User Feedback Form

Please rate the following aspects of the system out of five:

User interface

Adding and editing questions

Moving questions to and from the quiz


Exporting and saving quizzes

Adding and editing class logs

5: Unusable	4: Poor	3: Neutral	2: Good	1: Excellent
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Additional feedback on features such as clarity of user guide, ease of use of the system, etc.

It would have been nice to multi select questions when moving to quiz.  
 Doesn't validate against multiple class names  
 All of the buttons are a mystery until rollover.  
 Filtering isn't cumulative -> only one class at a time

Signed: 

Date: 26/2/17

# Benchmark

## User Feedback Form

Please rate the following aspects of the system out of five:

User interface

Adding and editing questions

Moving questions to and from the quiz

Exporting and saving quizzes

Adding and editing class logs

	5: Unusable	4: Poor	3: Neutral	2: Good	1: Excellent
User interface	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Adding and editing questions	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Moving questions to and from the quiz	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Exporting and saving quizzes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Adding and editing class logs	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Additional feedback on features such as clarity of user guide, ease of use of the system, etc.

A thorough, effective and nice looking system. I had some trouble with the installation, but suspect that was to do with network rights. Anyway, I was able to run from USB. The adding and editing of questions was very straight-forward and the marking facilities were excellent. I found the creation of quizzes a little more cryptic - it was easy to do, but perhaps could have benefited from more of an introduction in the user guide. Didn't really get anywhere with class logs - is it possible to link that with existing classes, or would that be done on Godalming Online. Export was fine and I managed to upload some questions to GOL.

Signed: *P. J. ...*

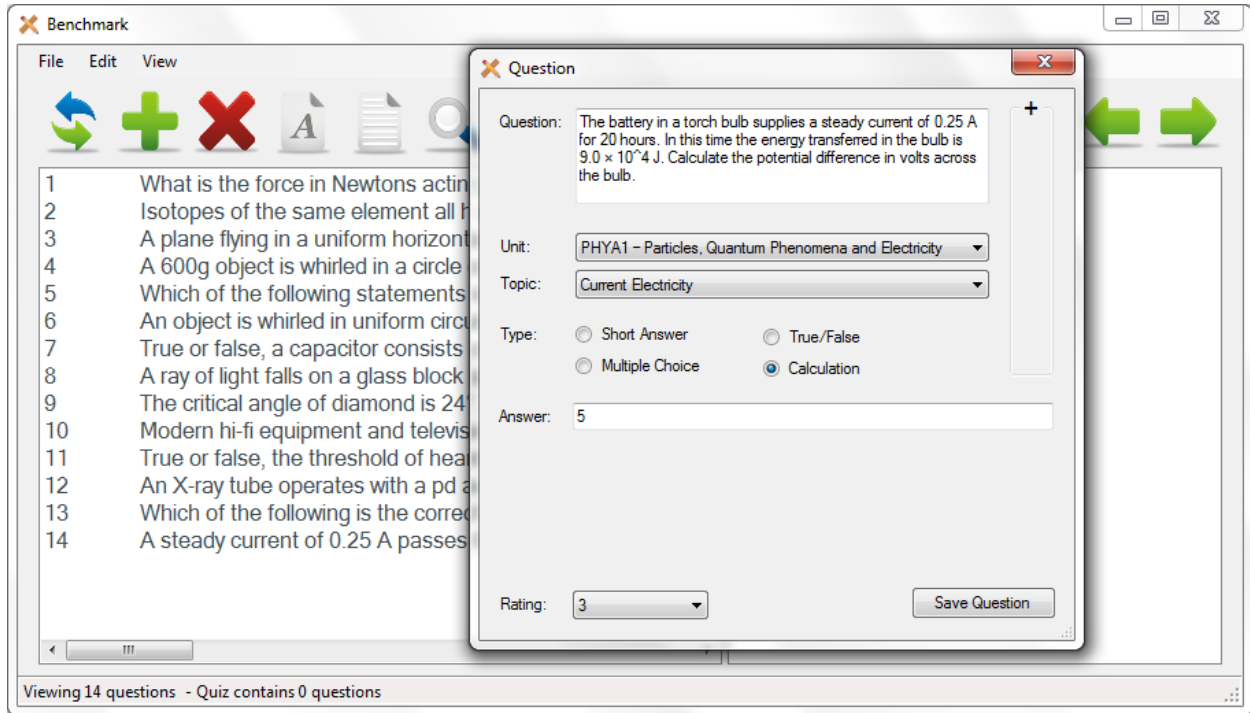
Date: 1/3/13

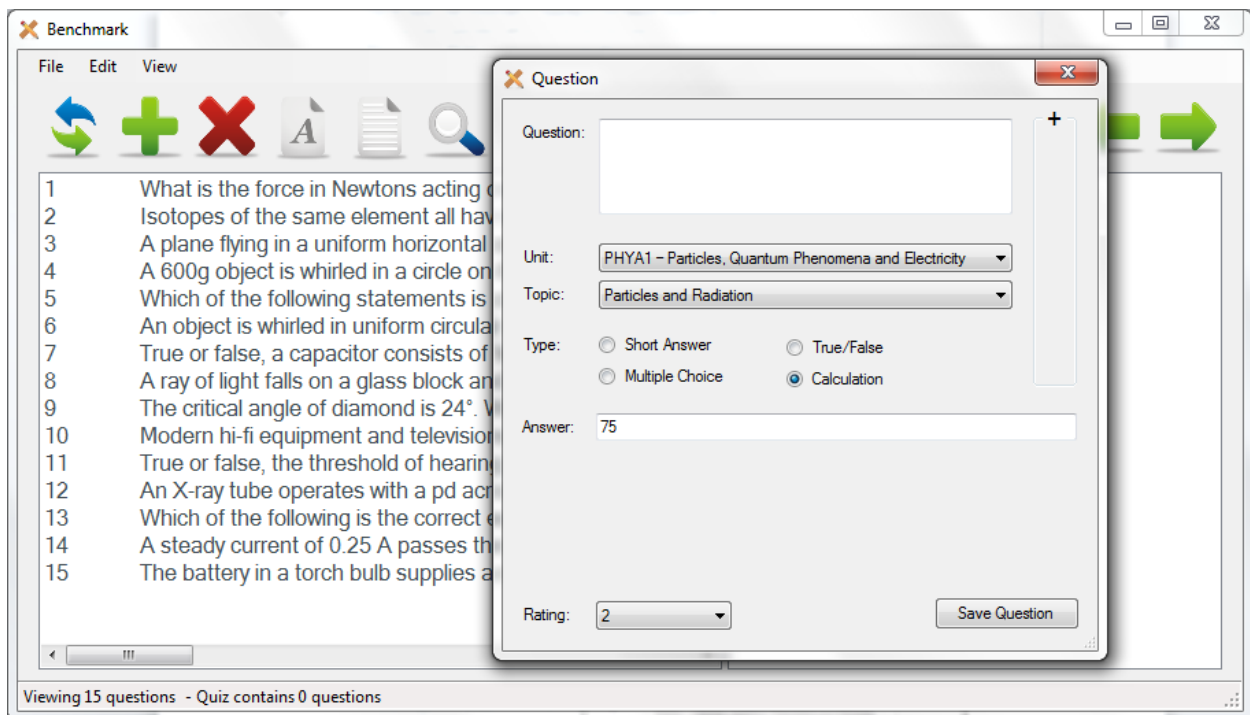
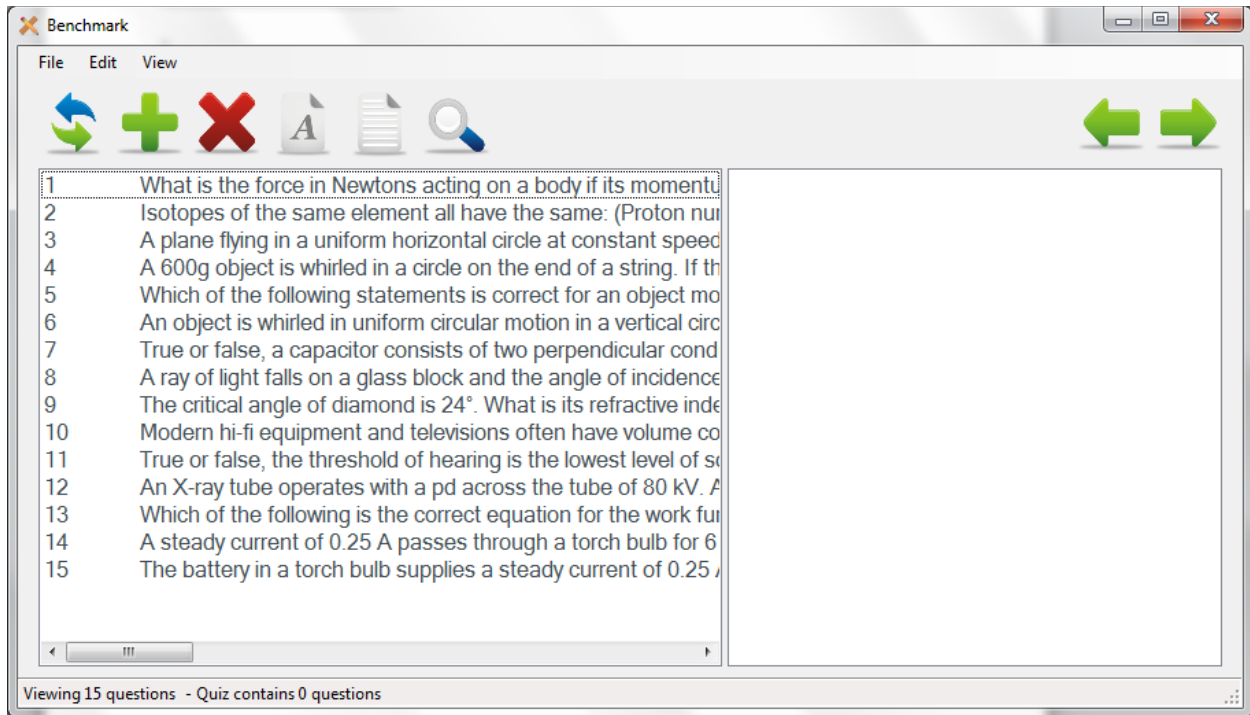


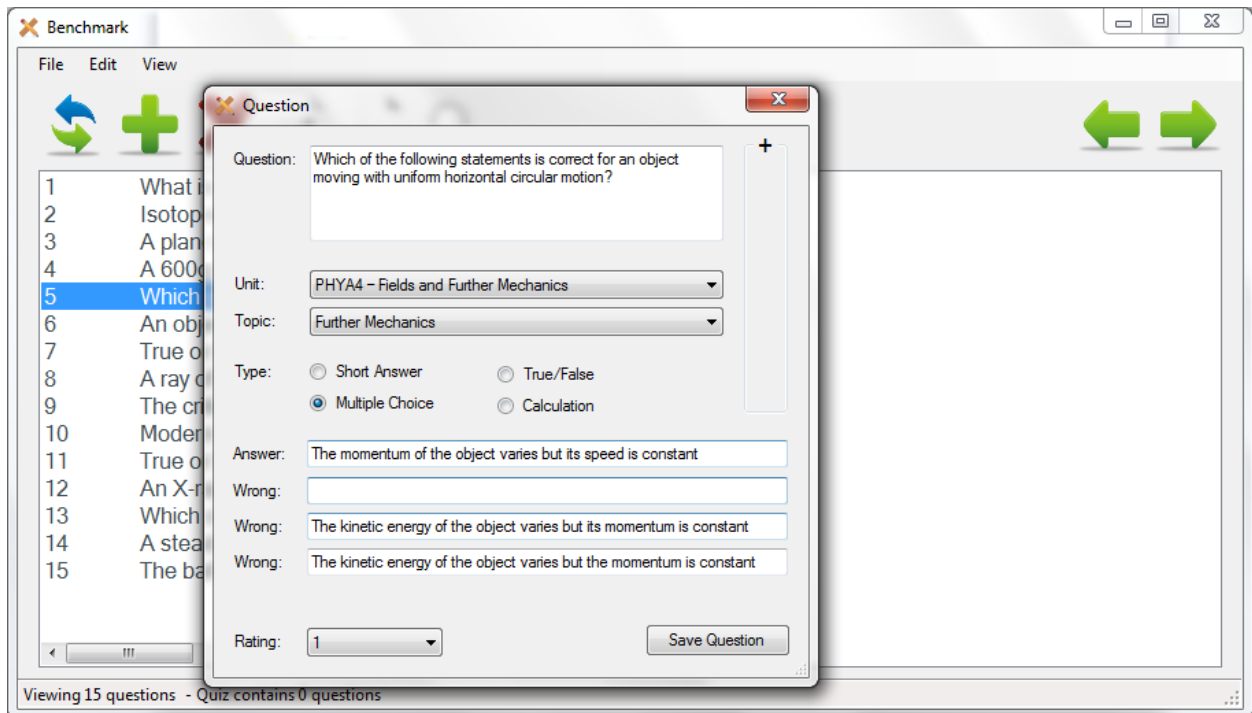
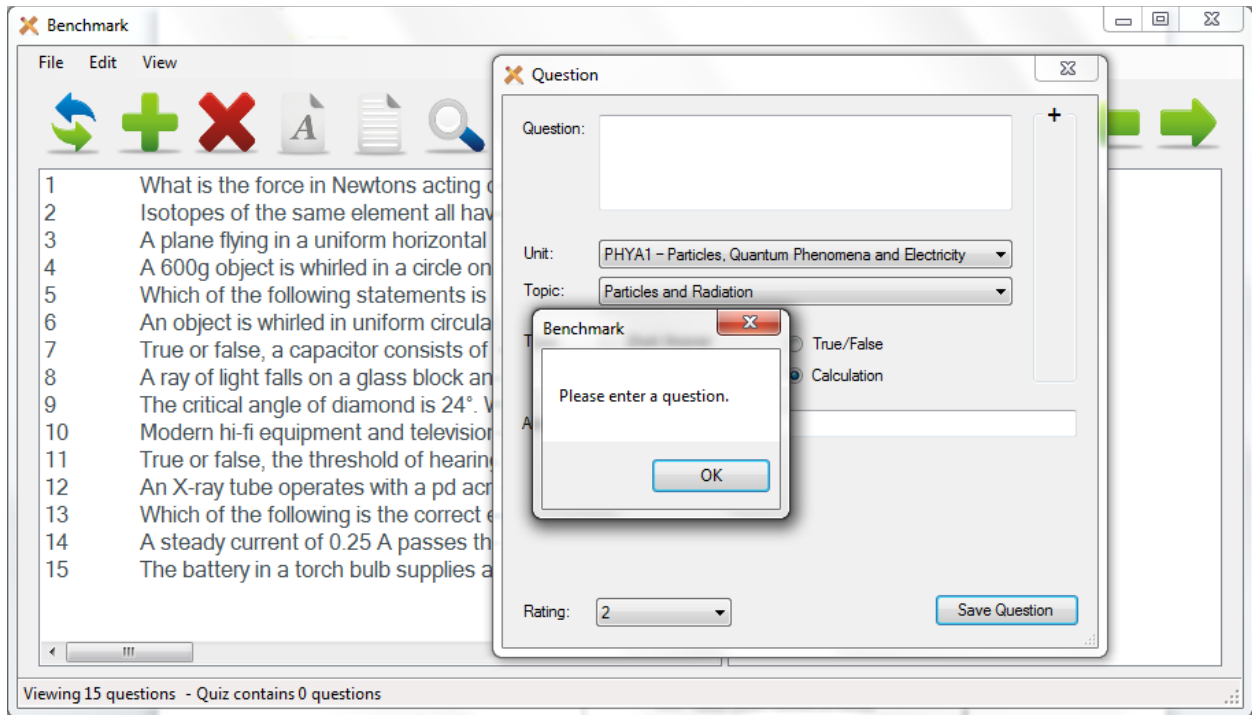


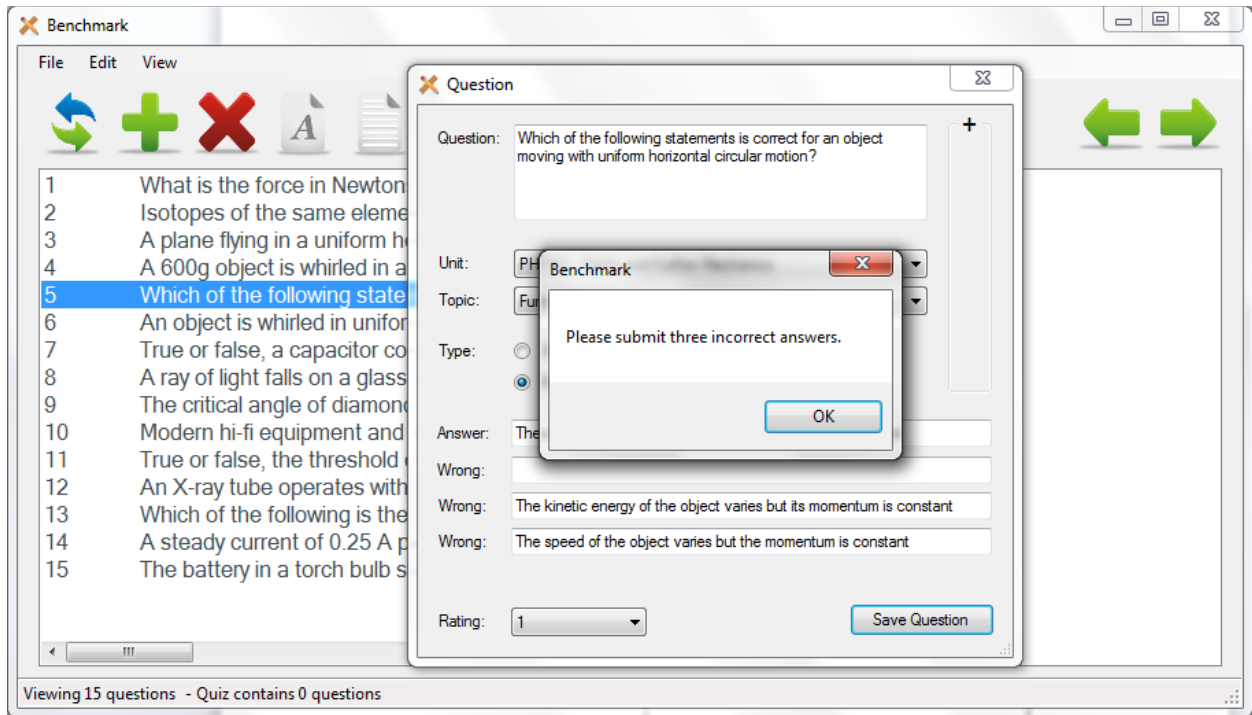
# Testing Outcome Screenshots

1

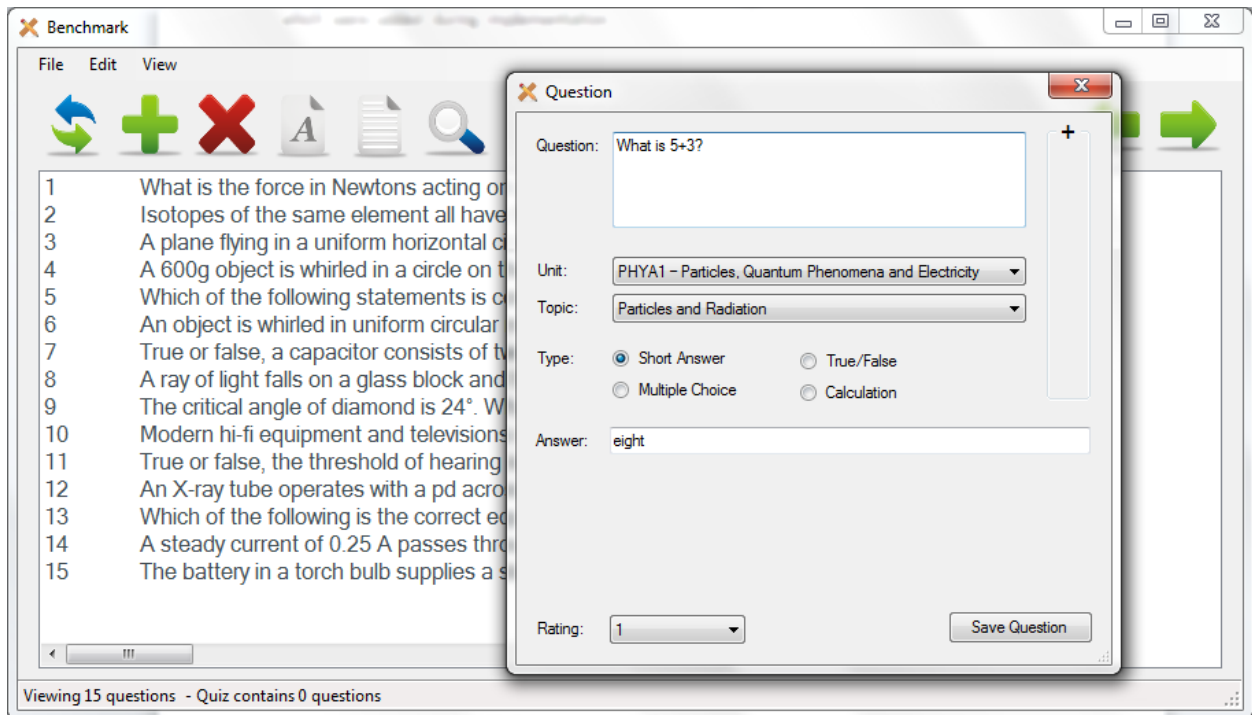


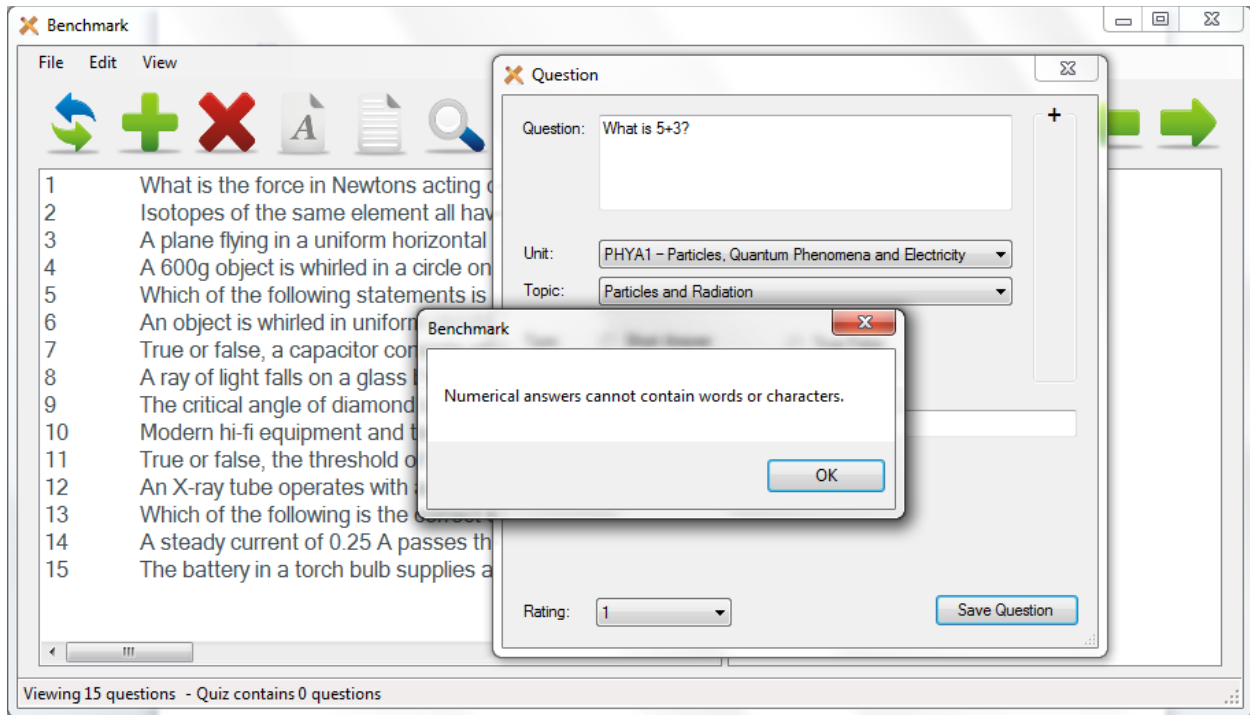




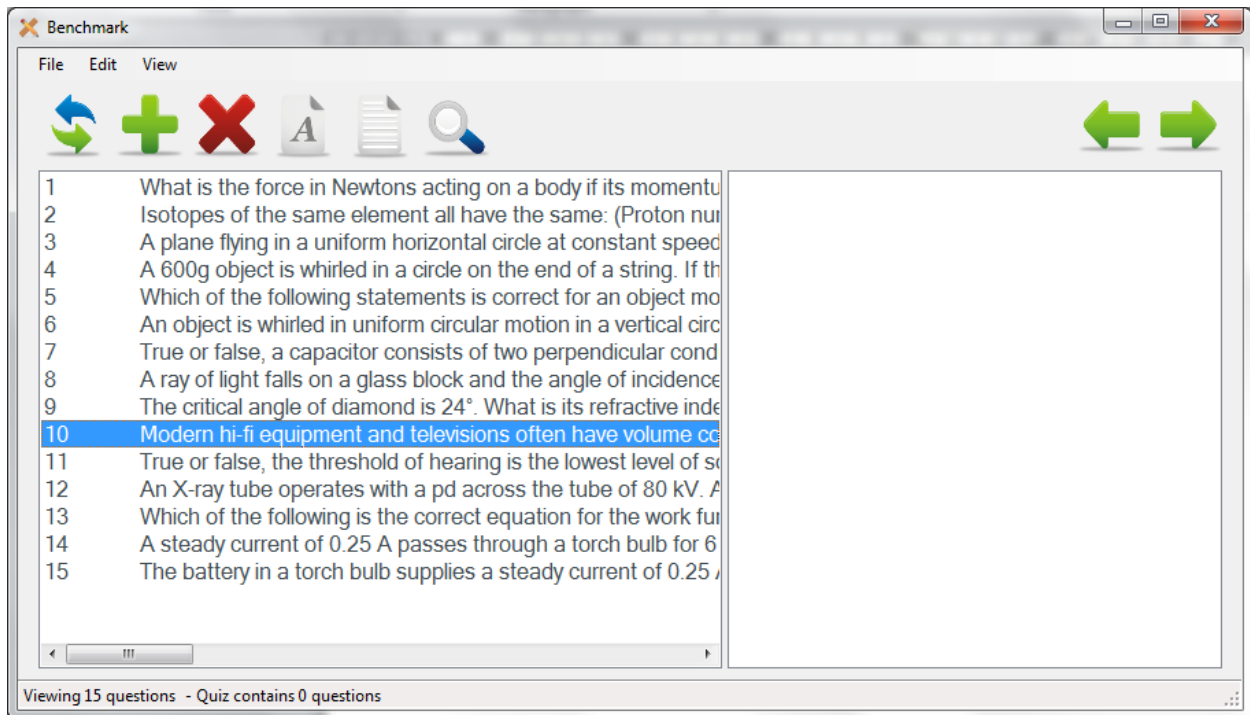


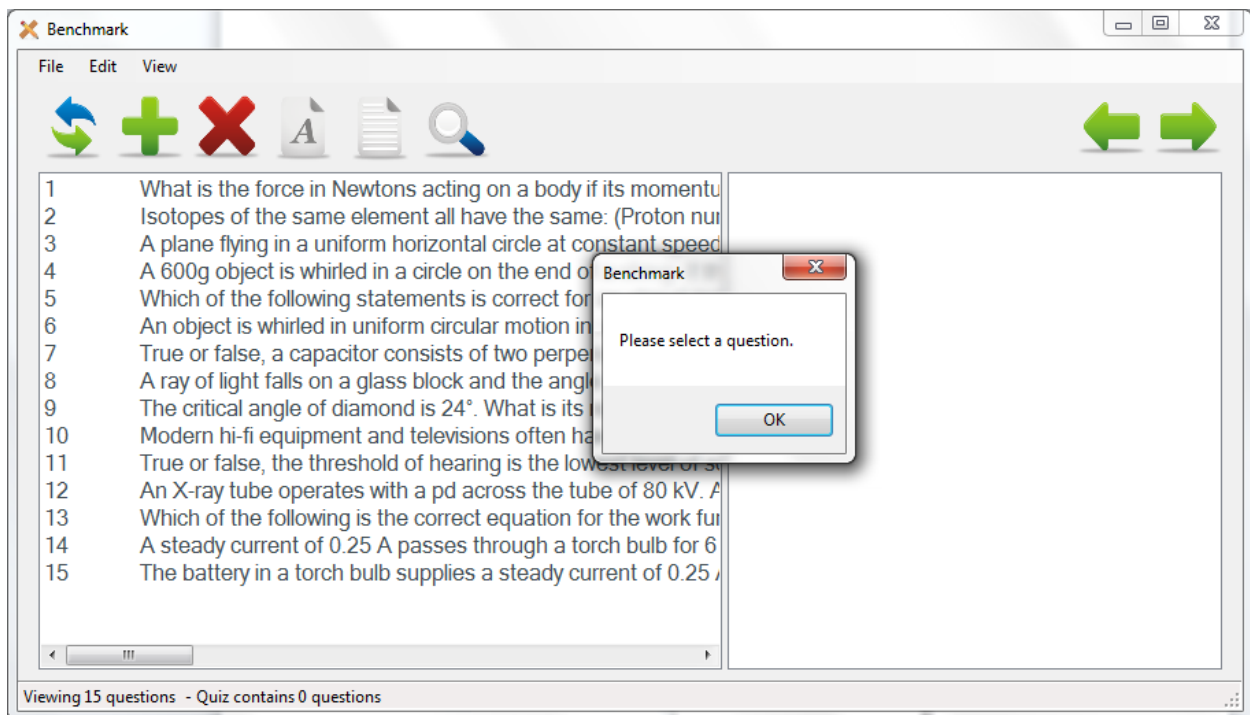
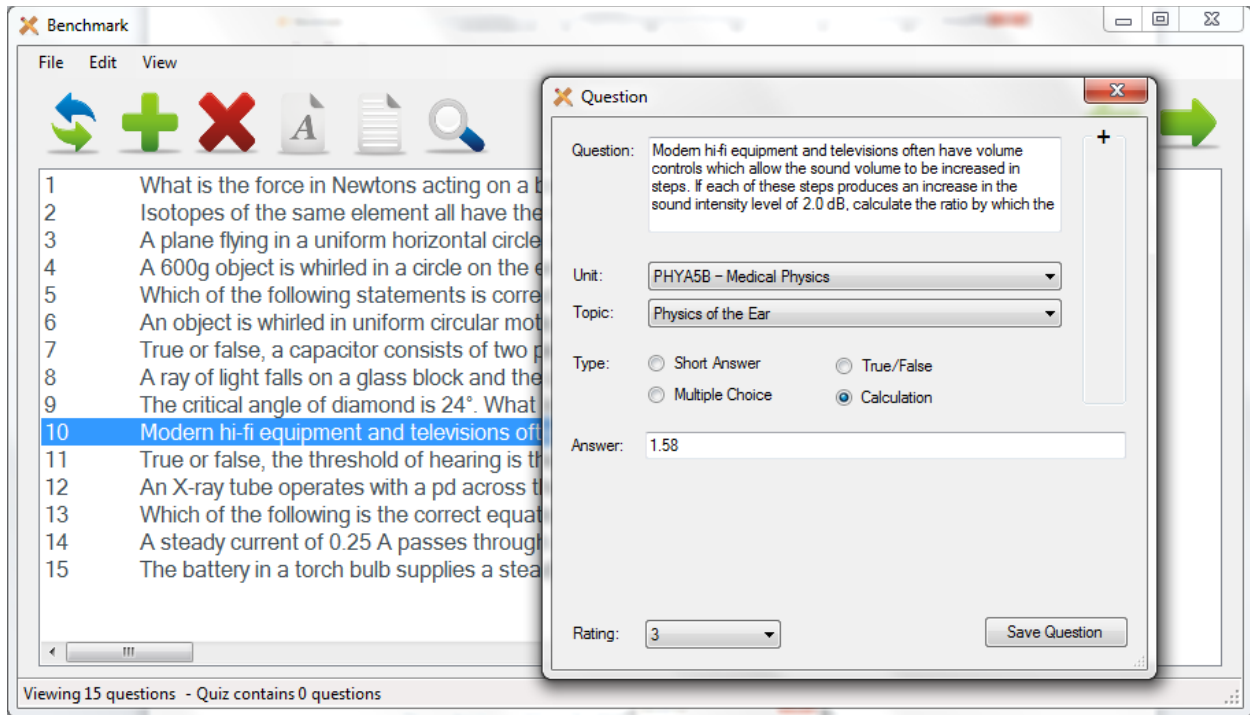
3



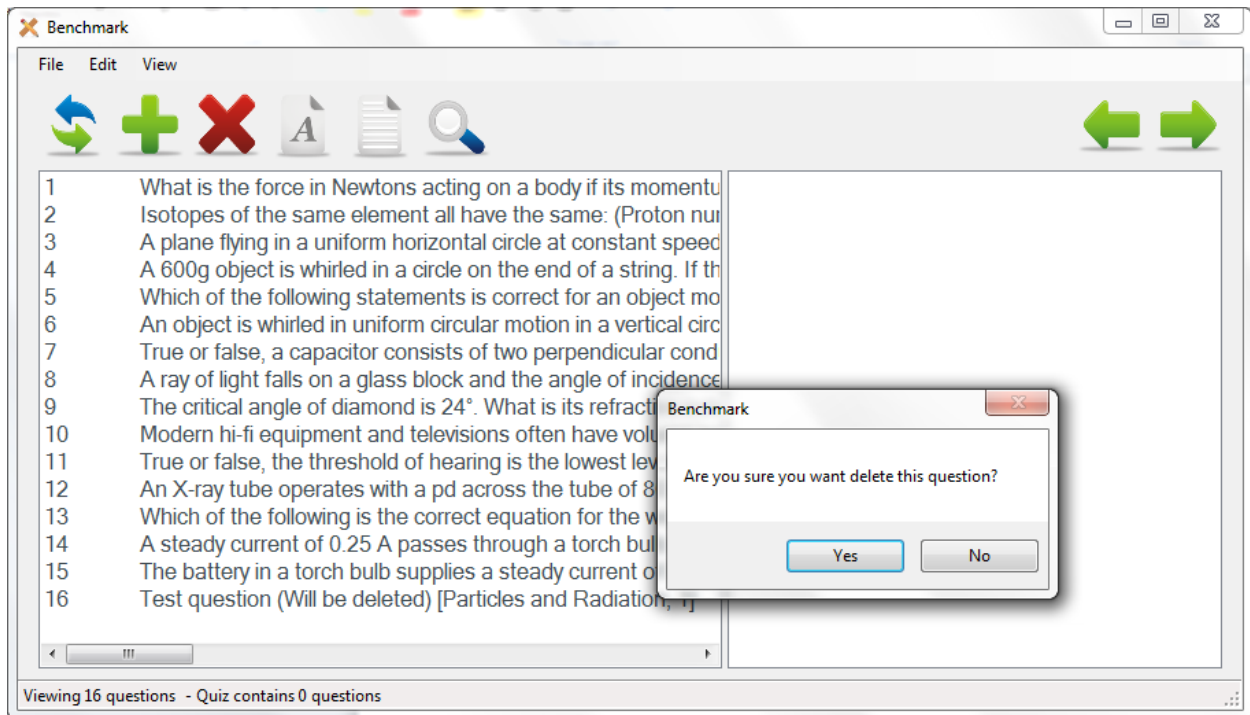
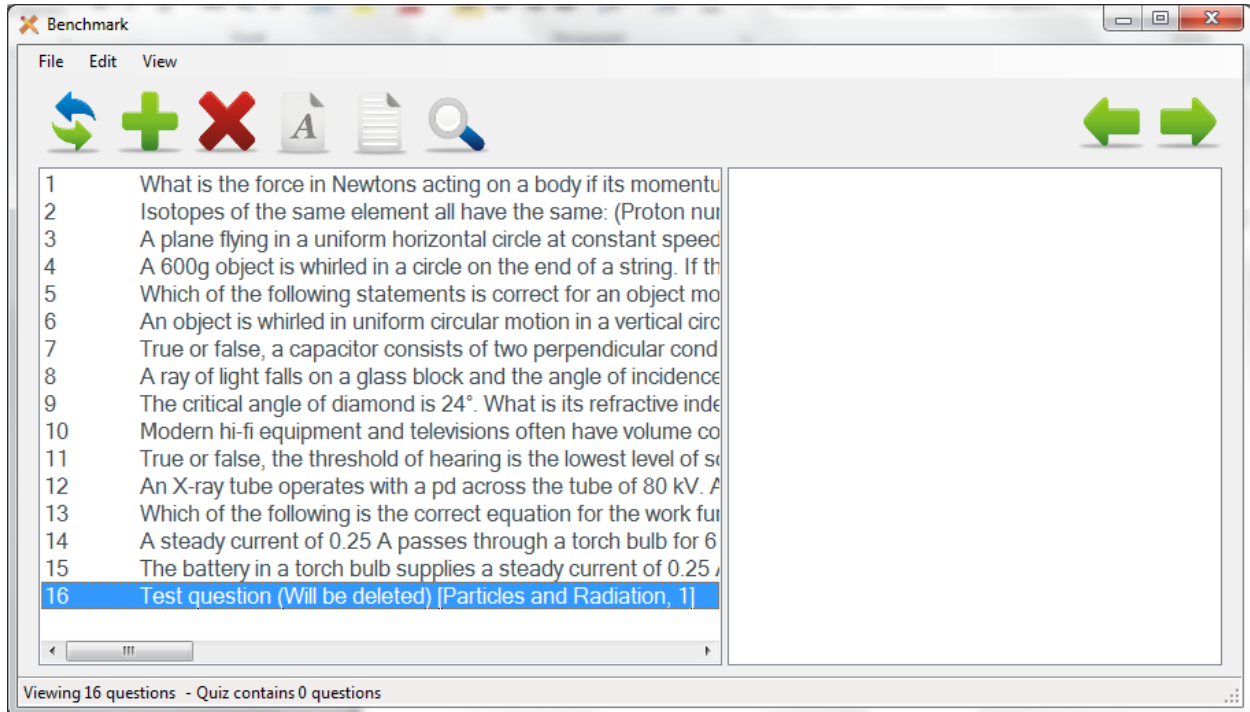


4

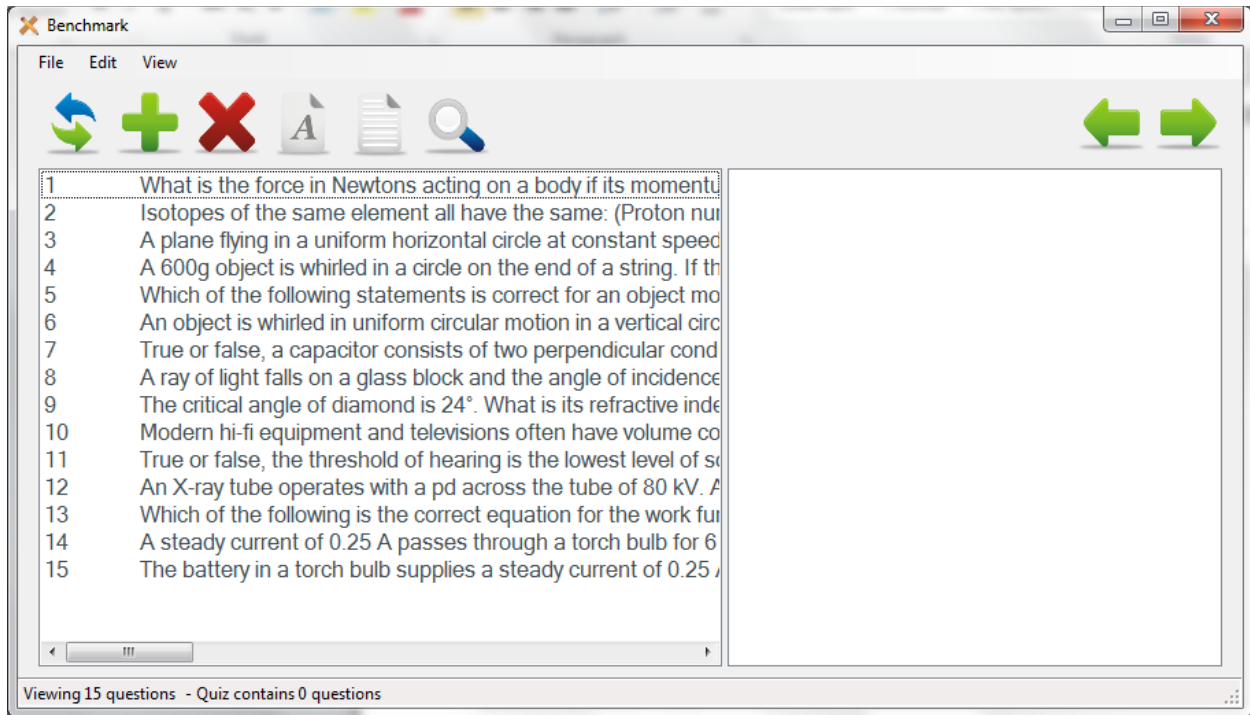




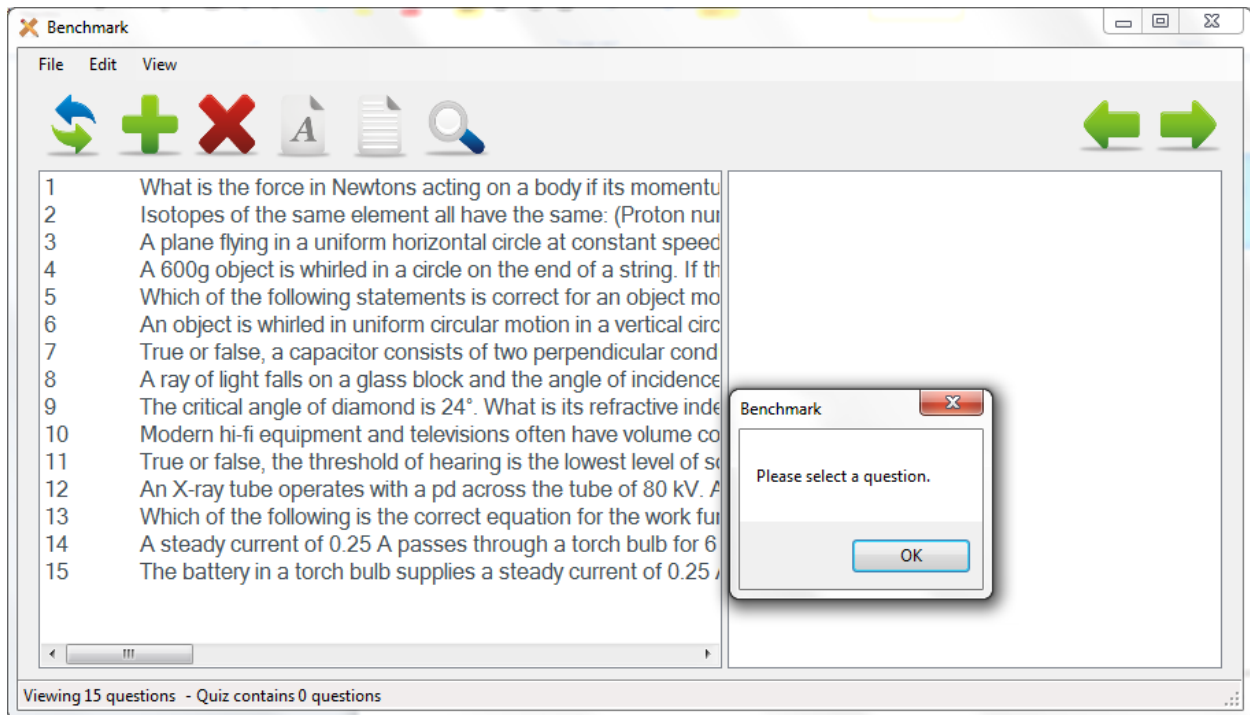
6



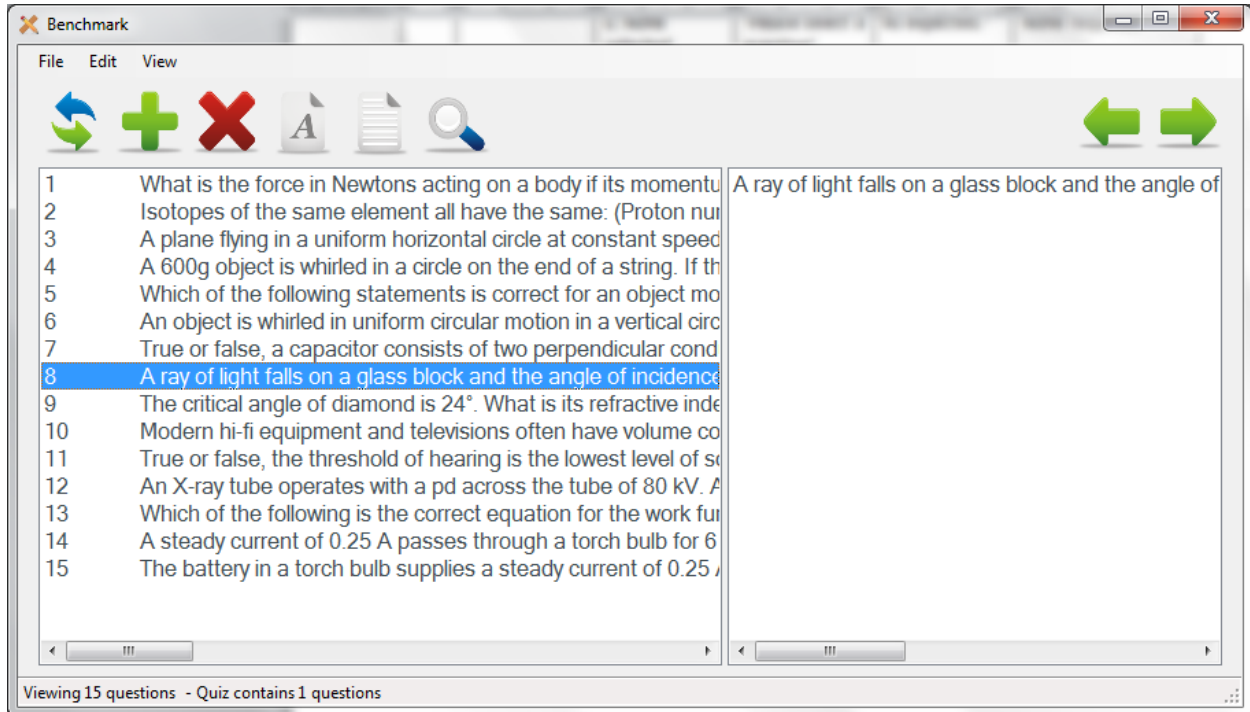




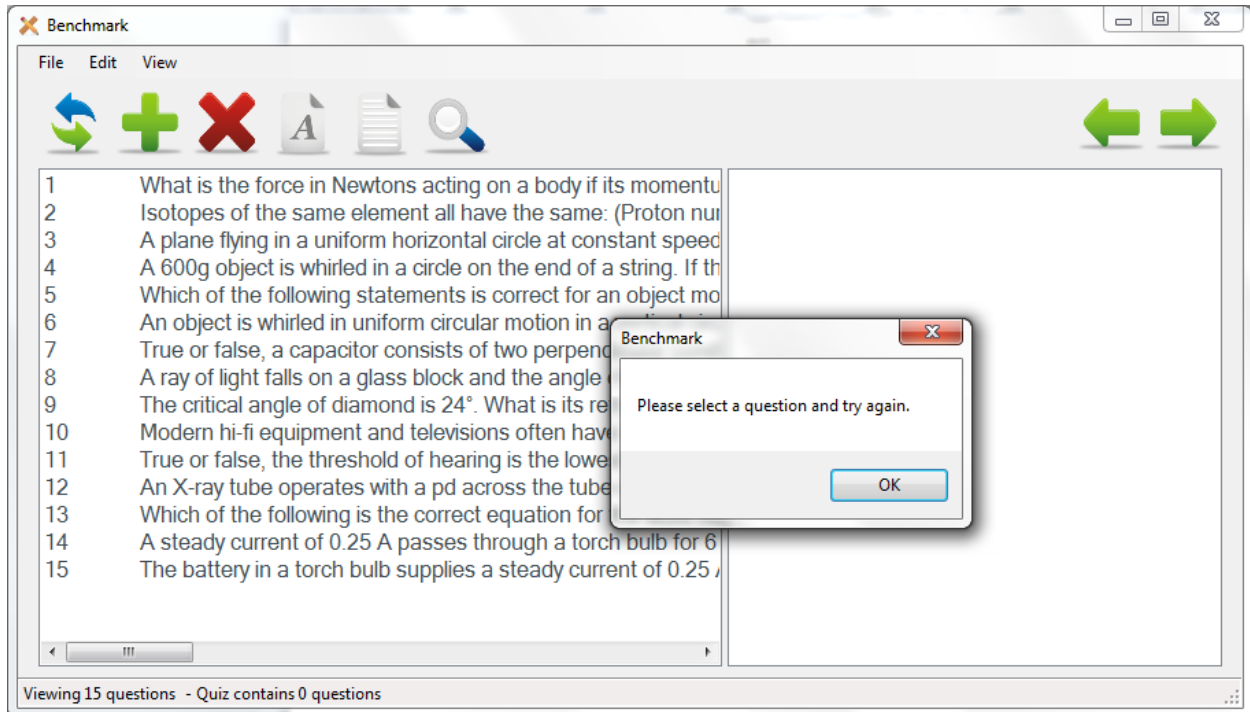
7



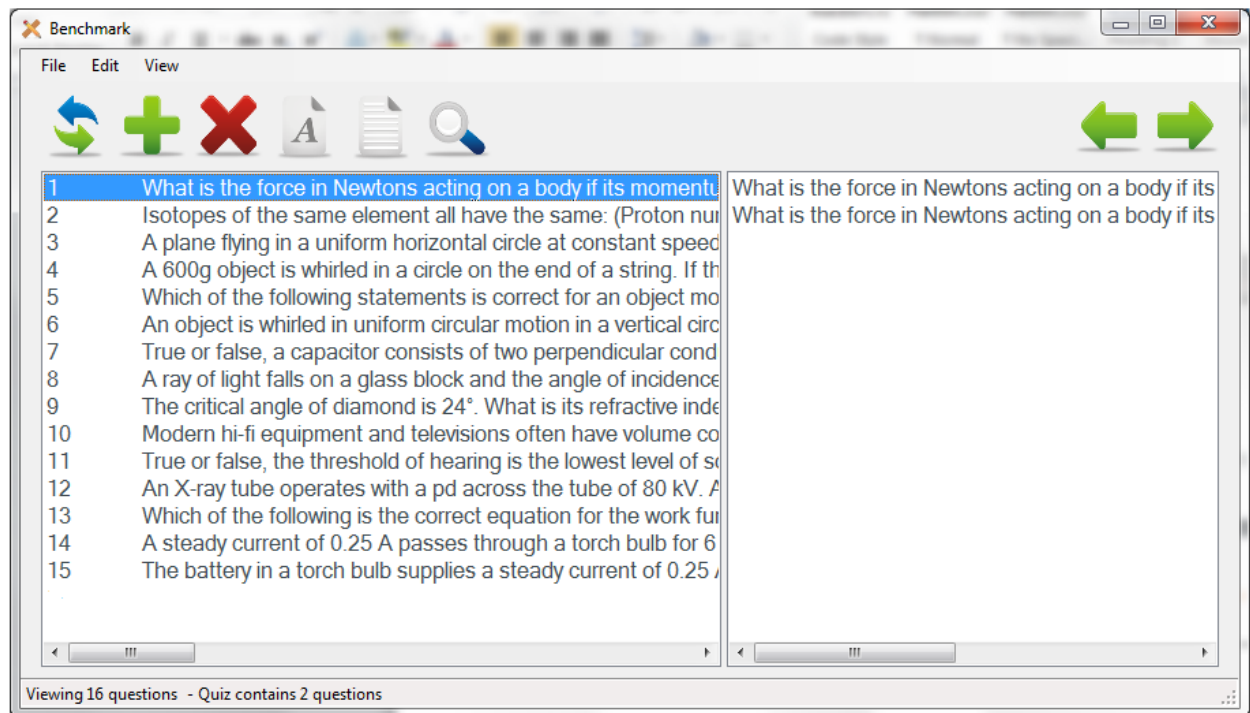
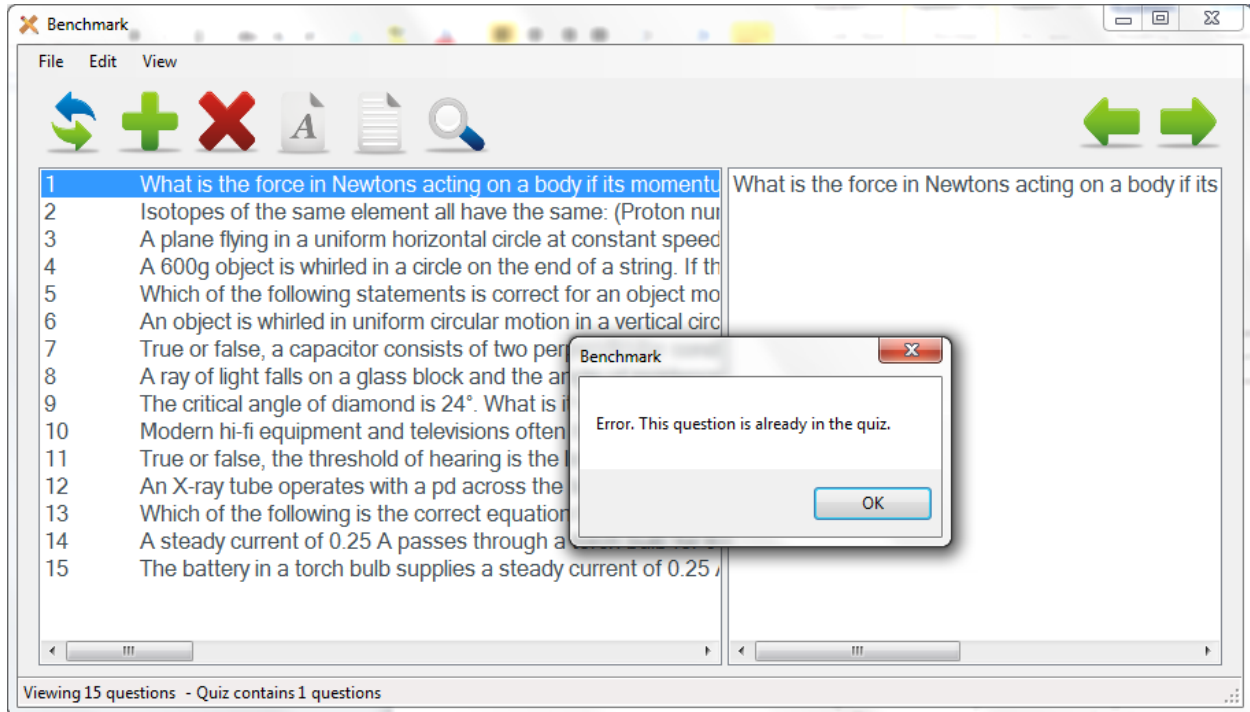
8



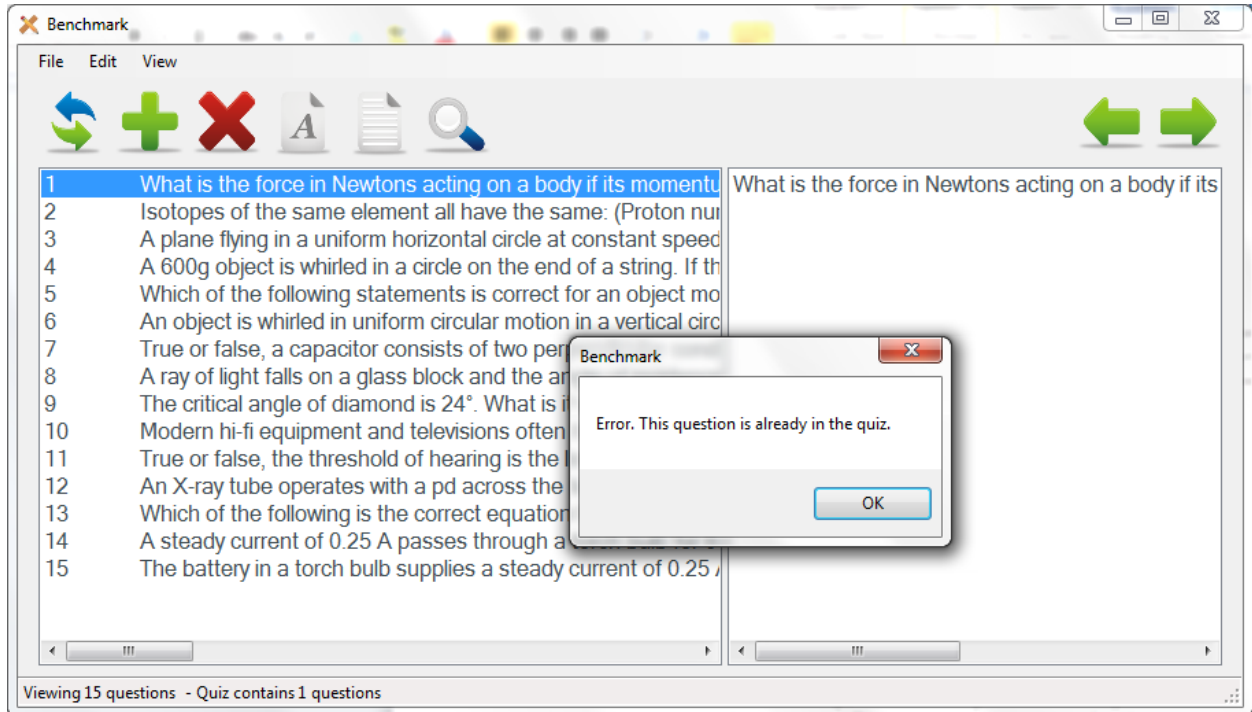
9



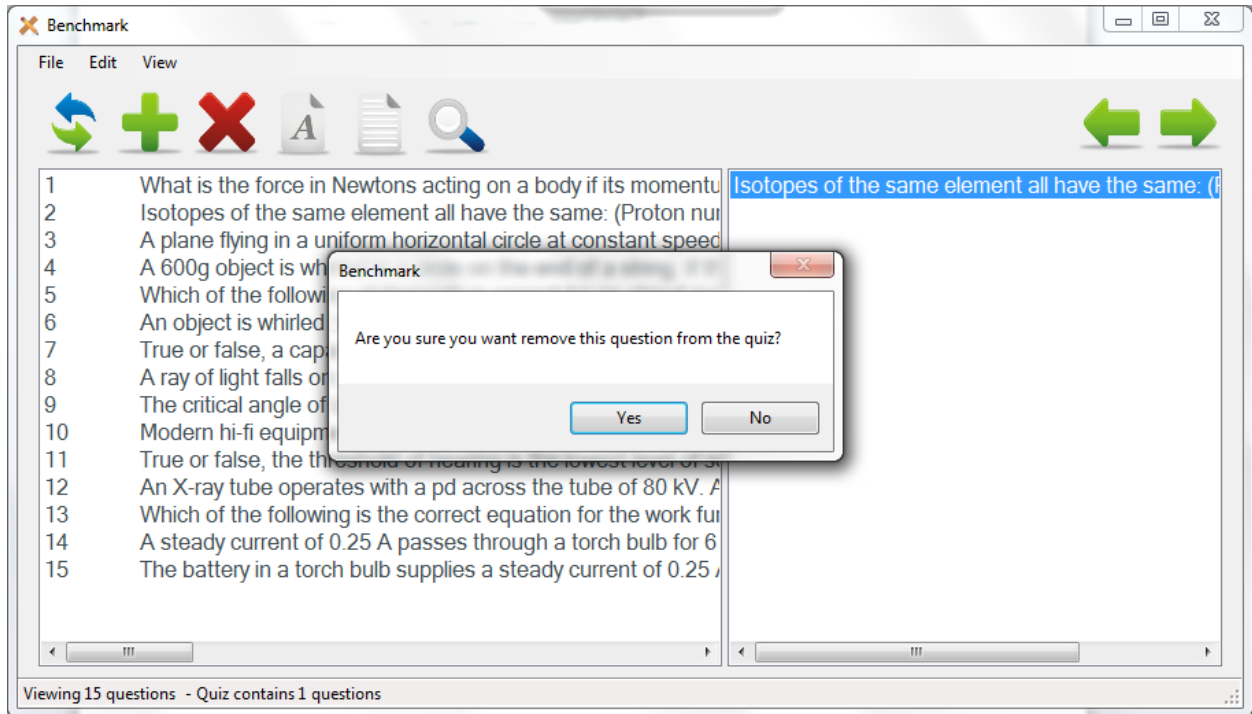
10

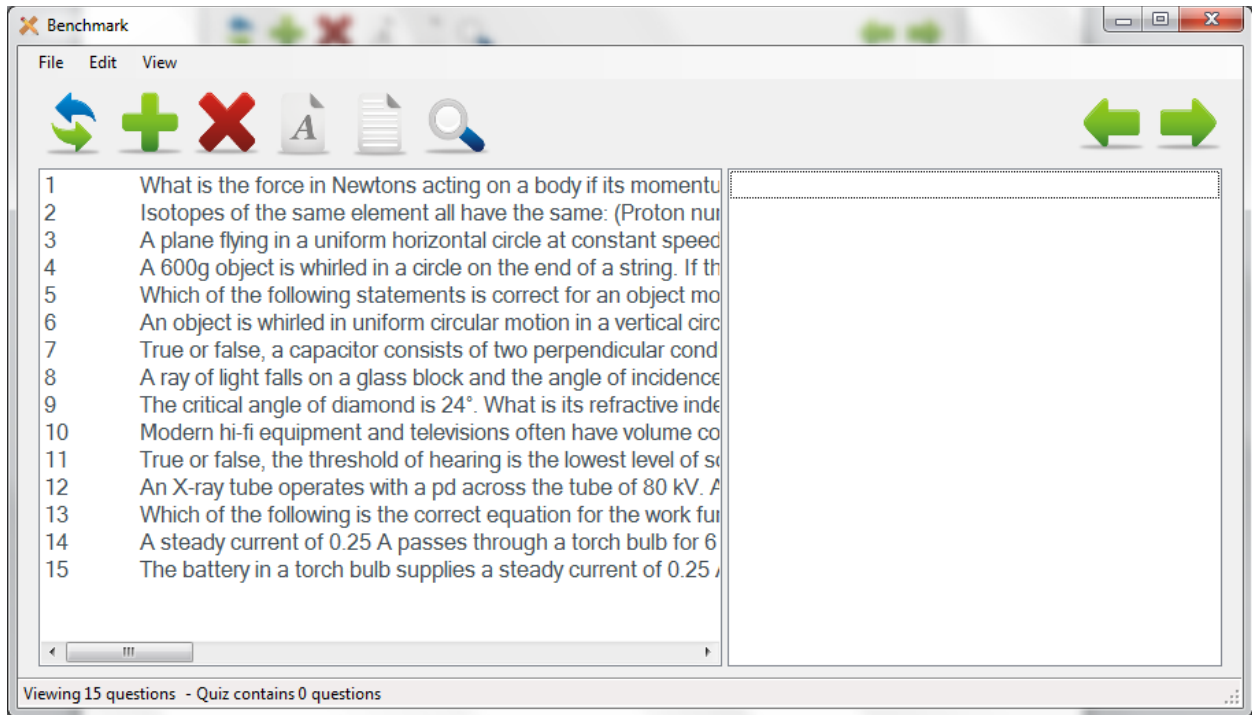


11

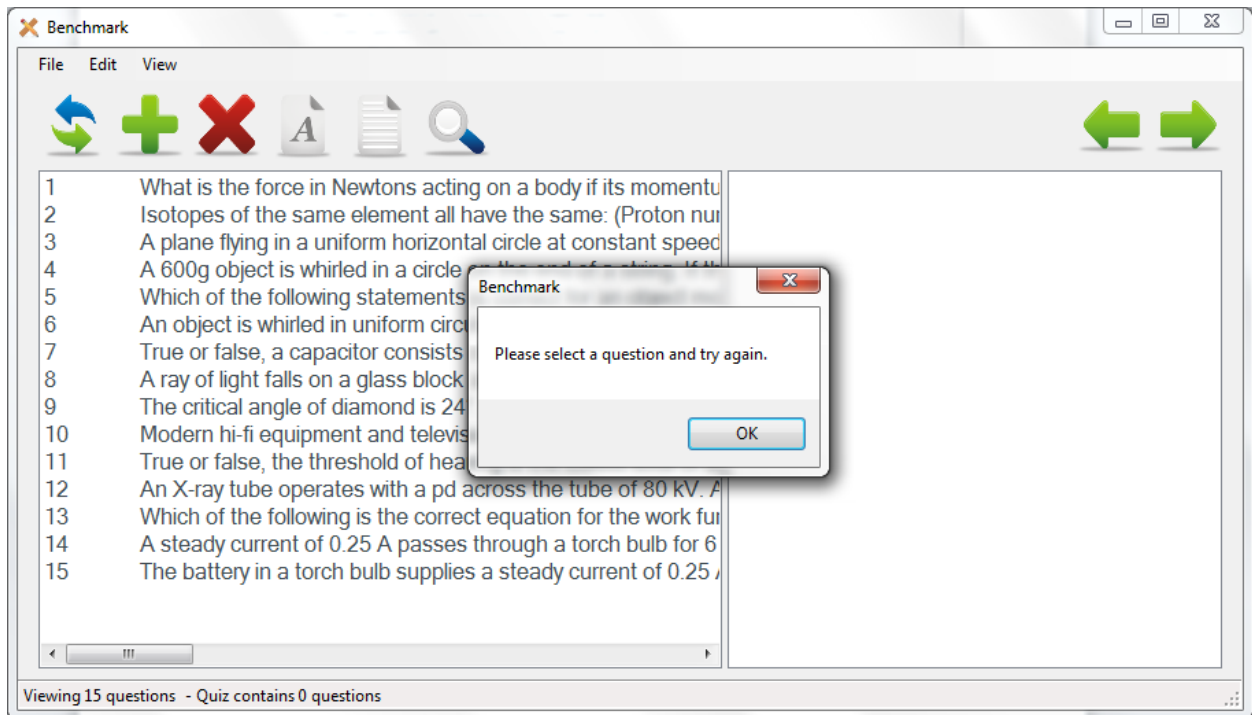


12

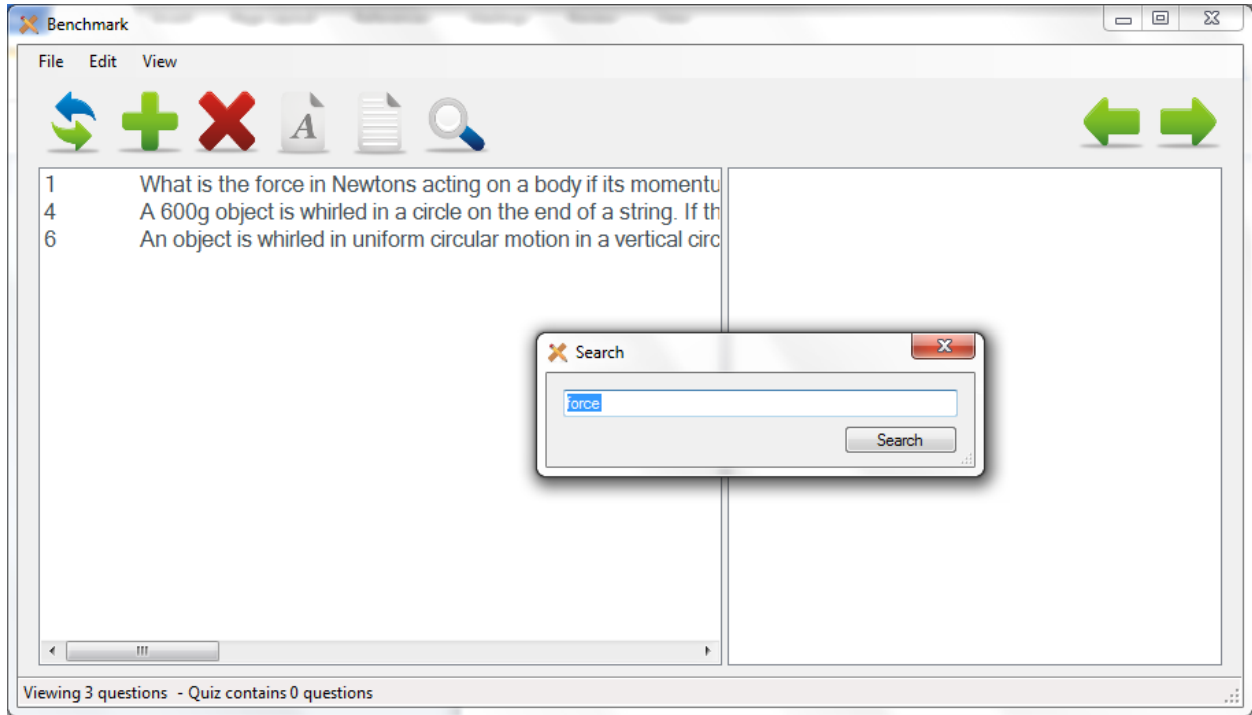




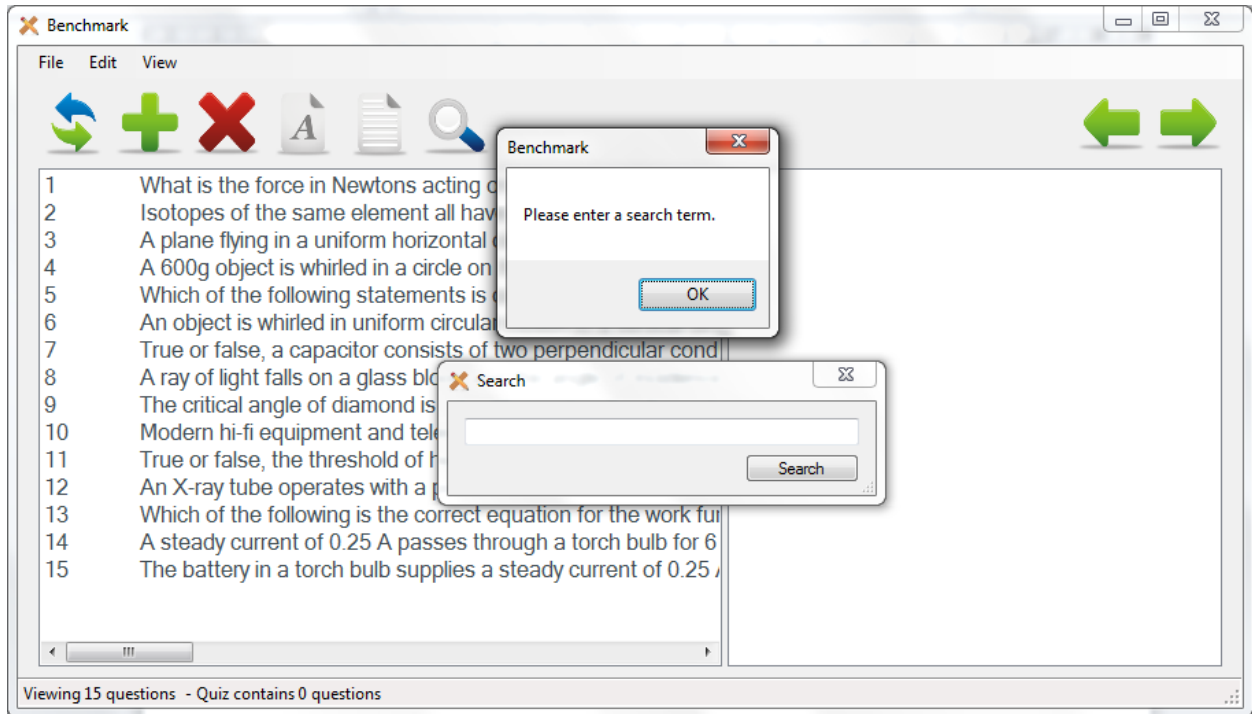
13



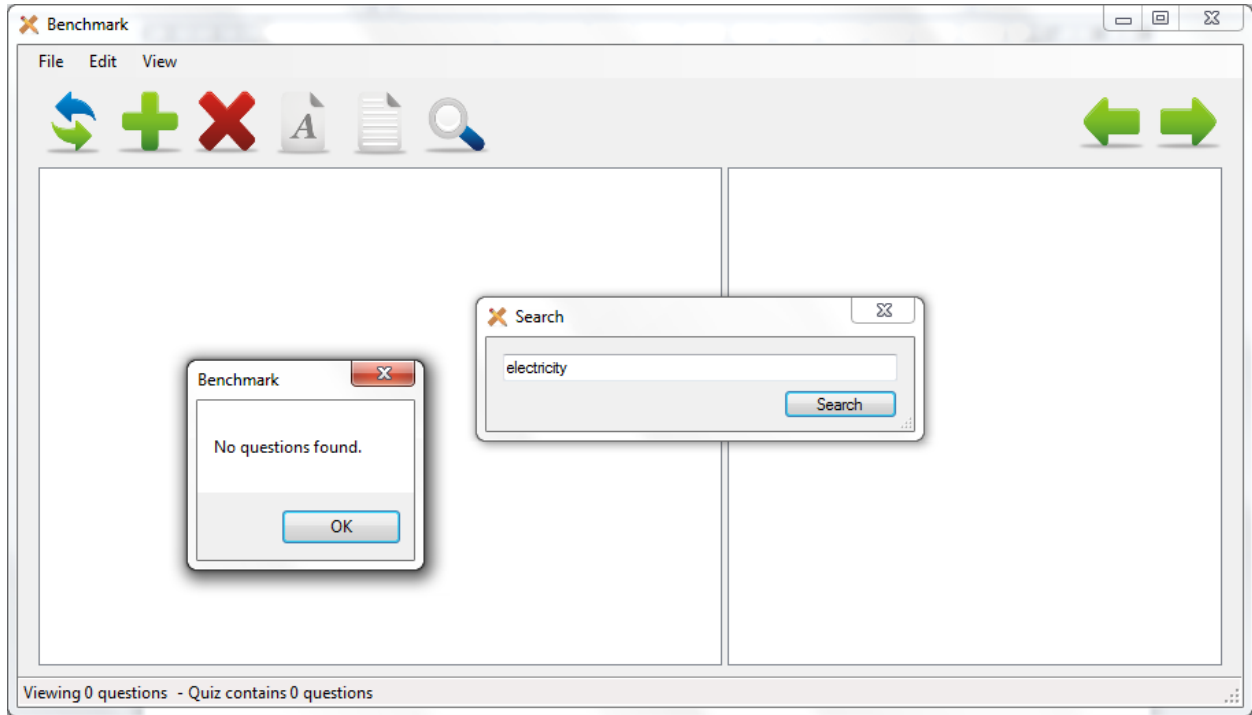
14



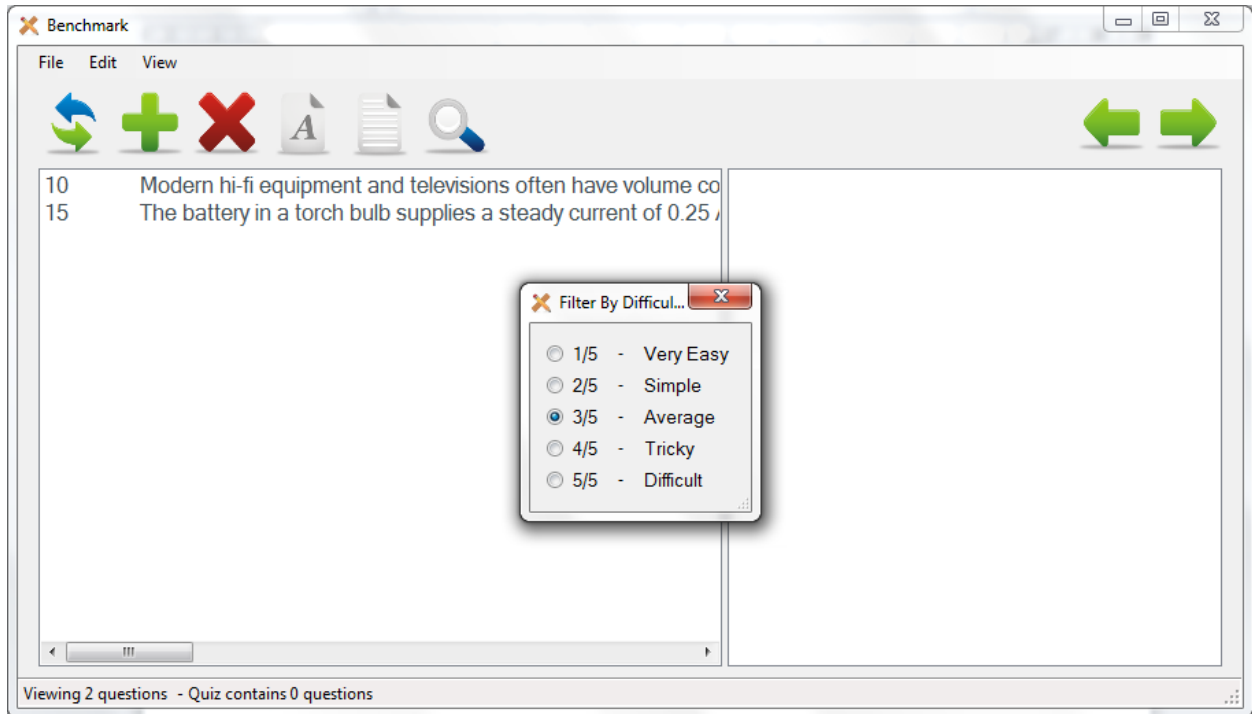
15



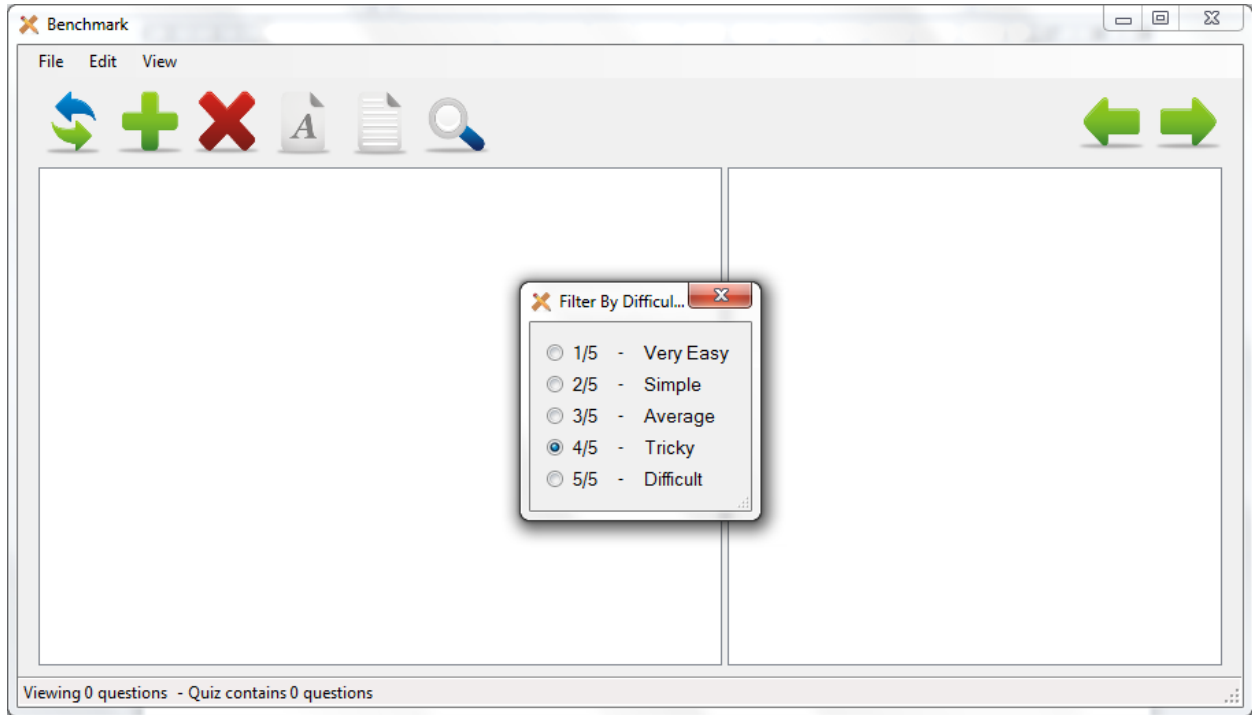
16



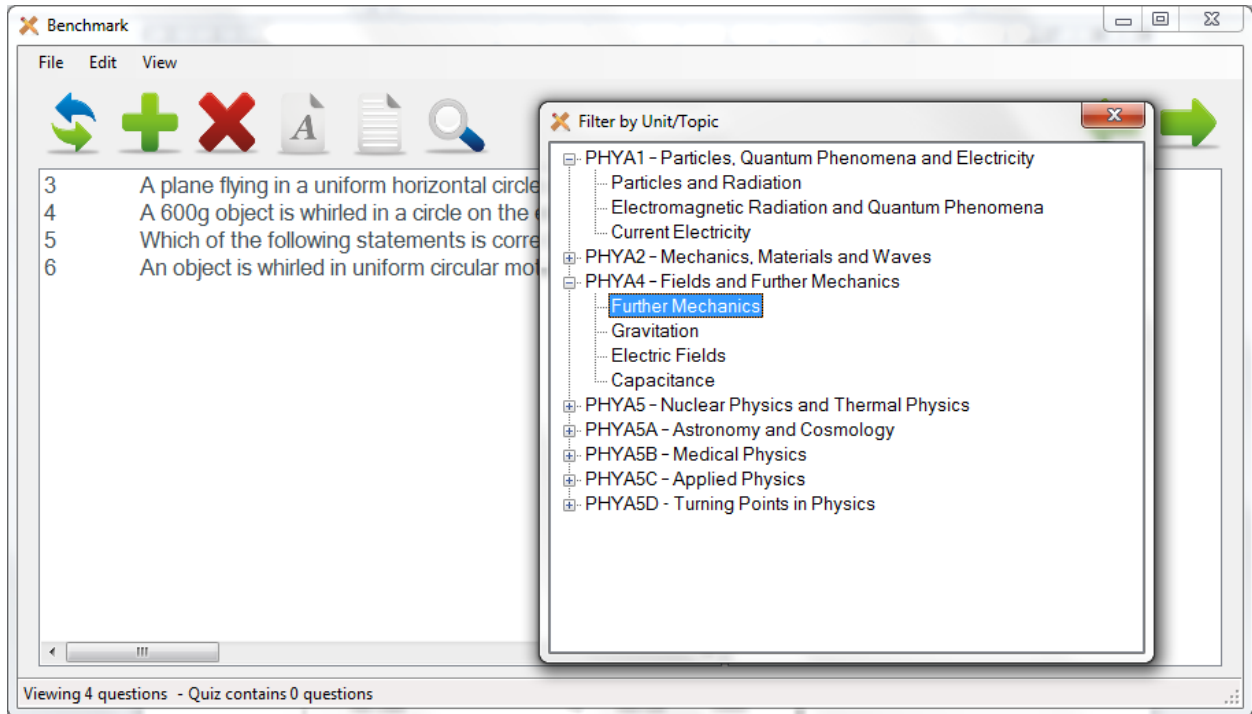
17



18

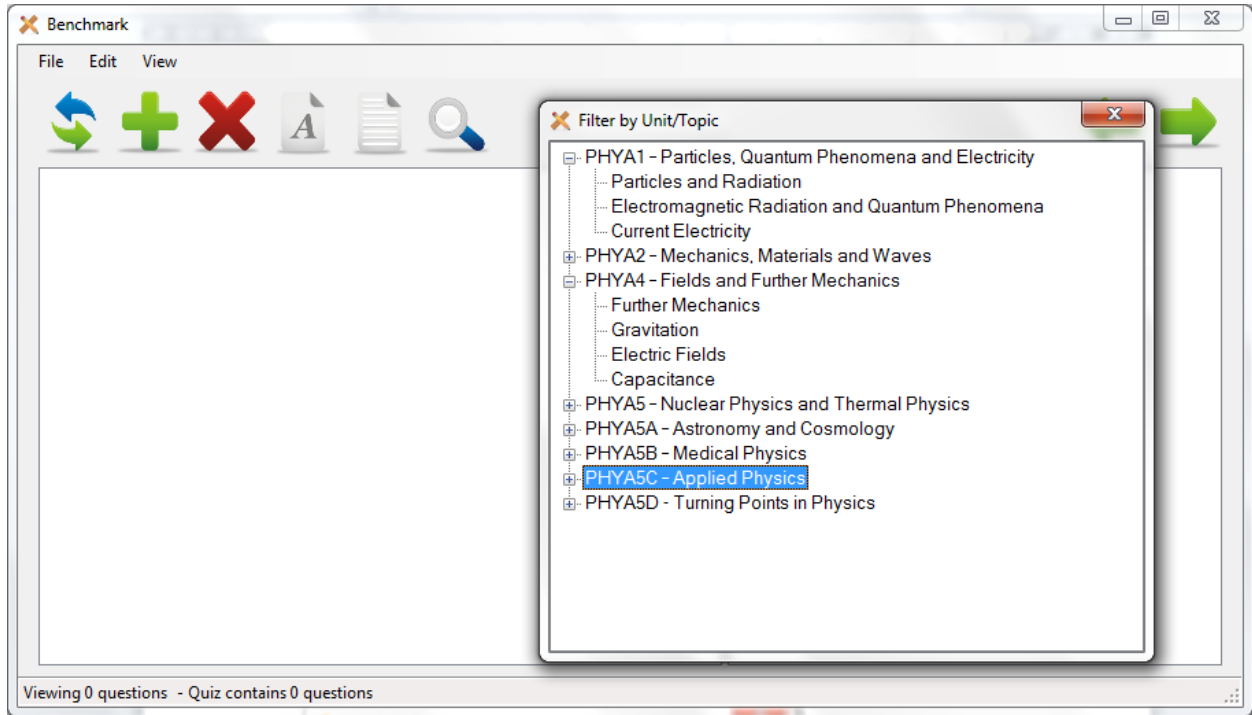


19

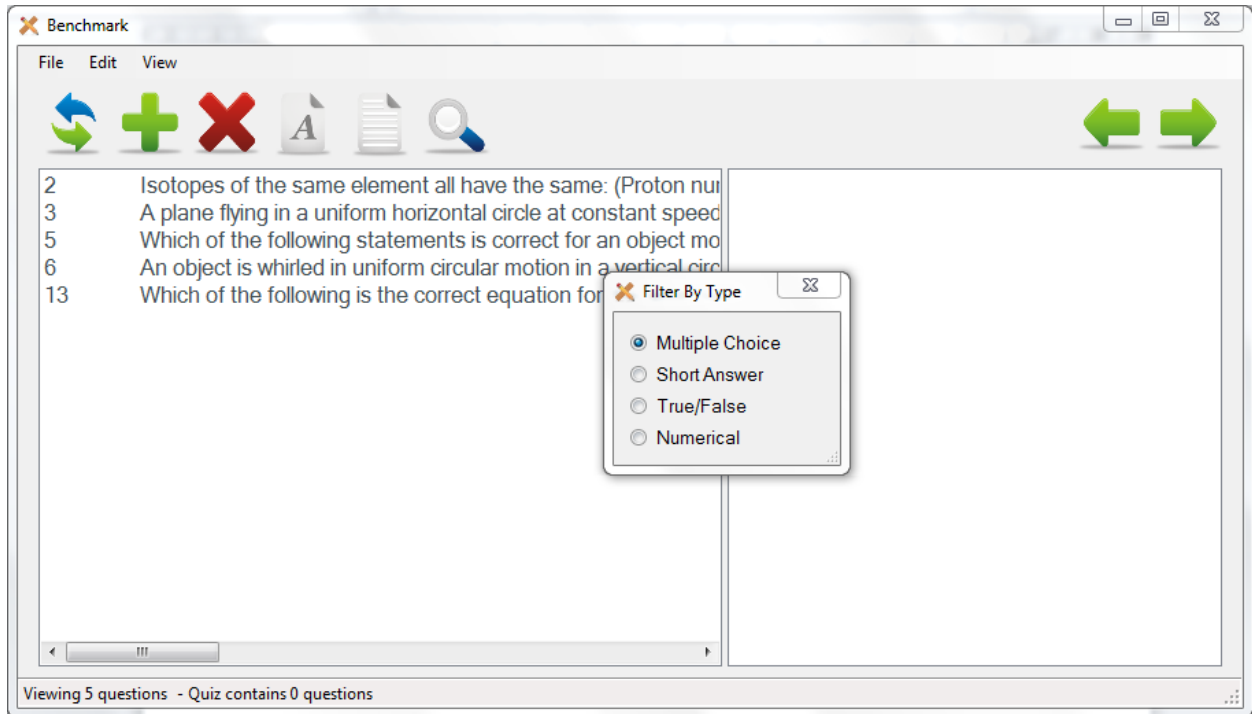




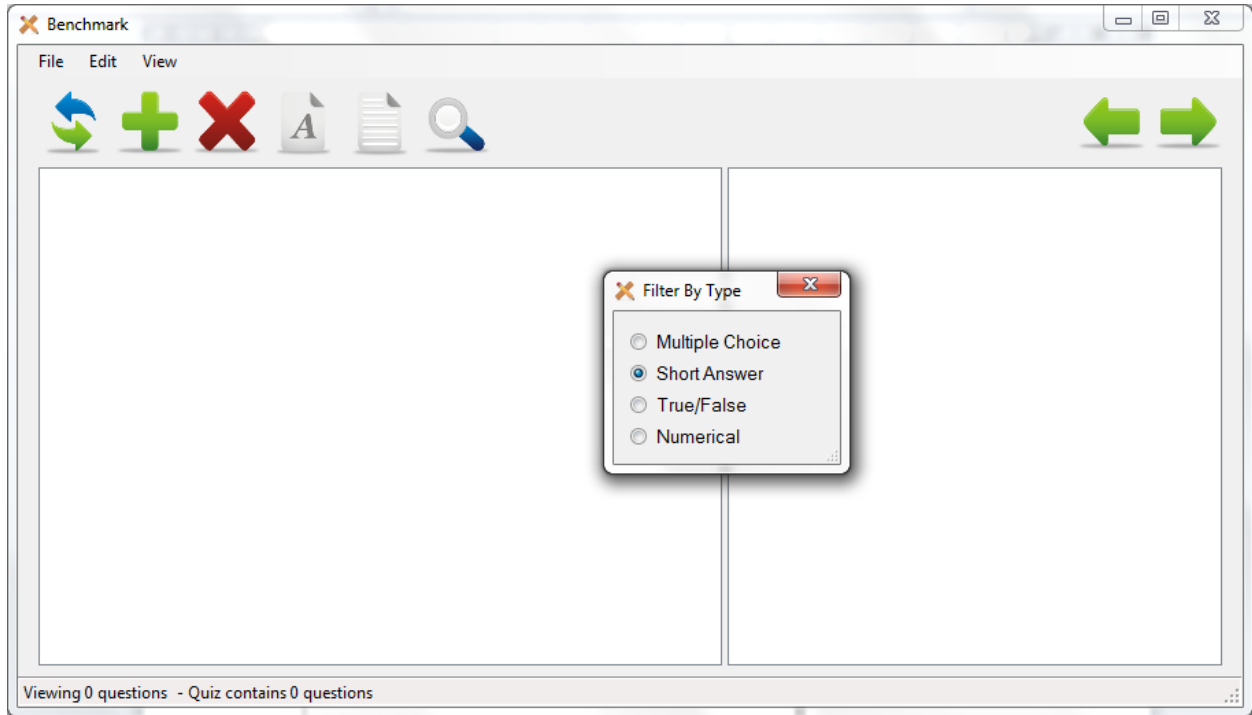
20



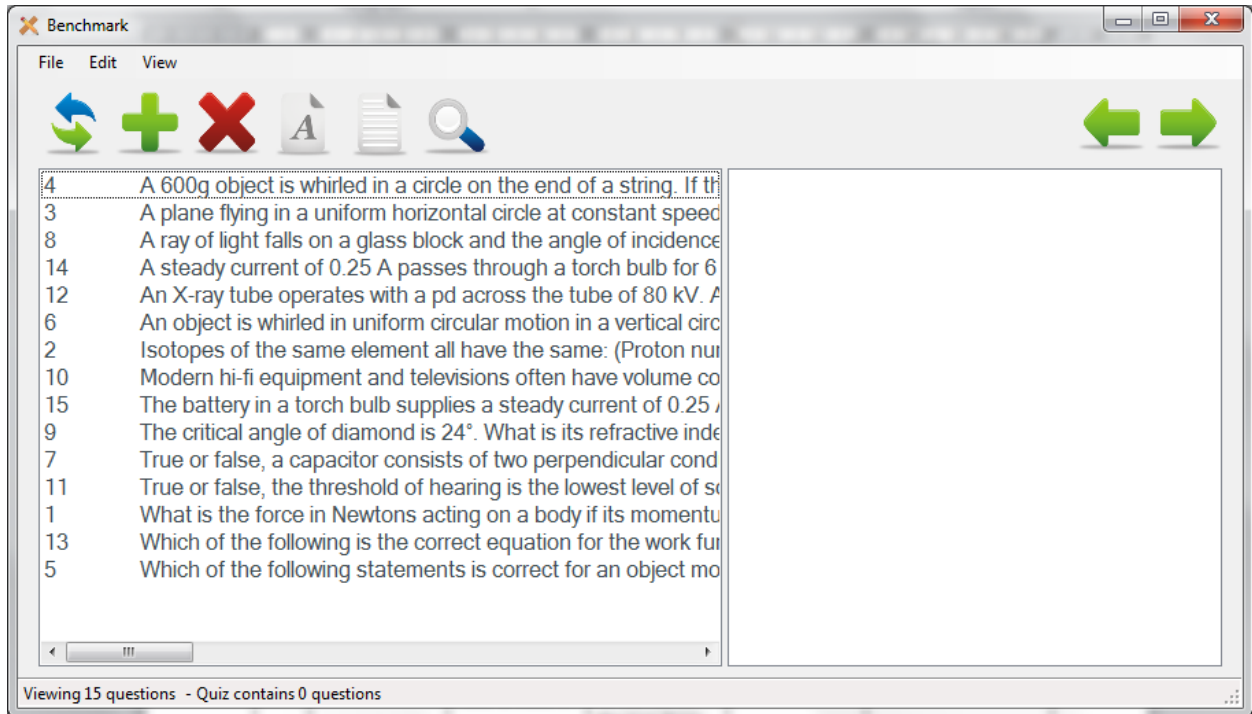
21



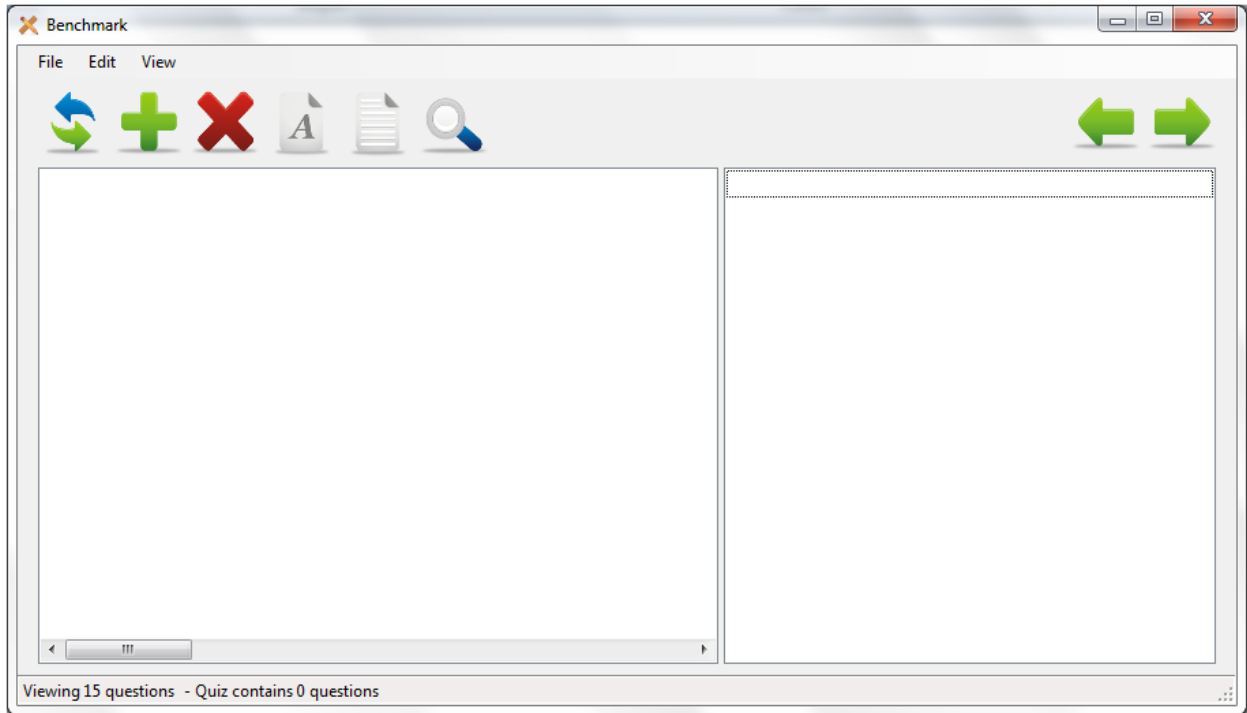
22



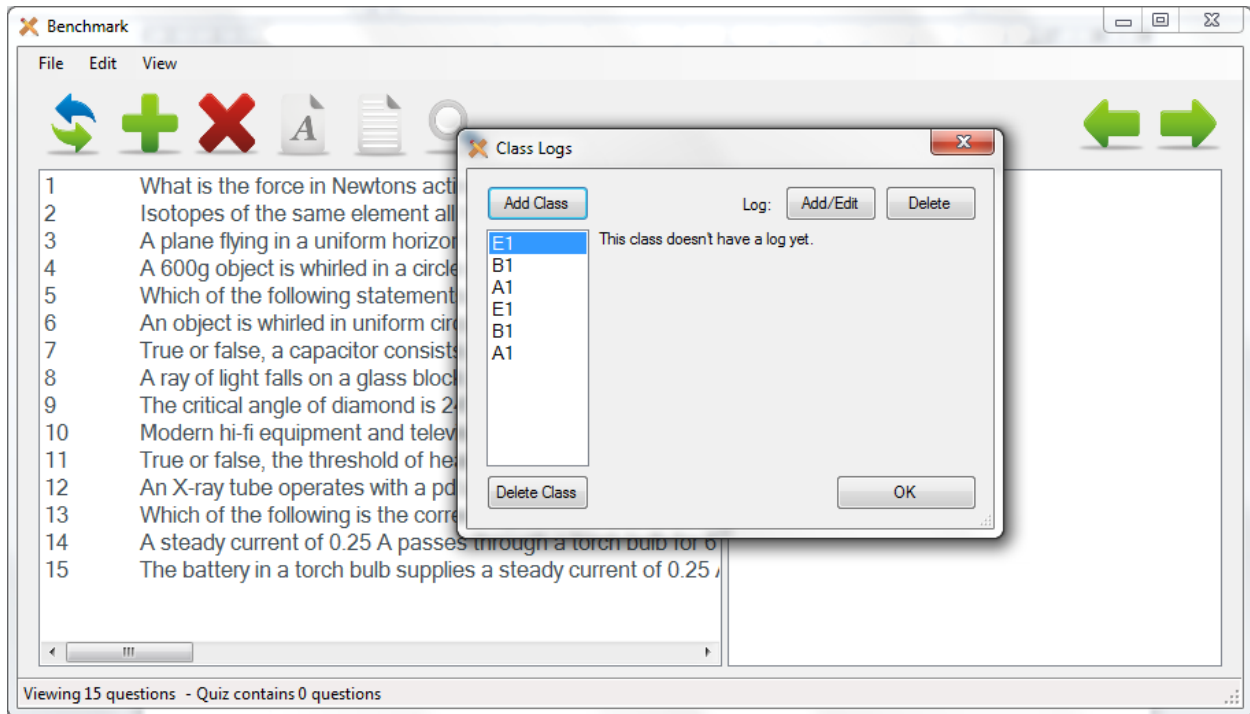
23



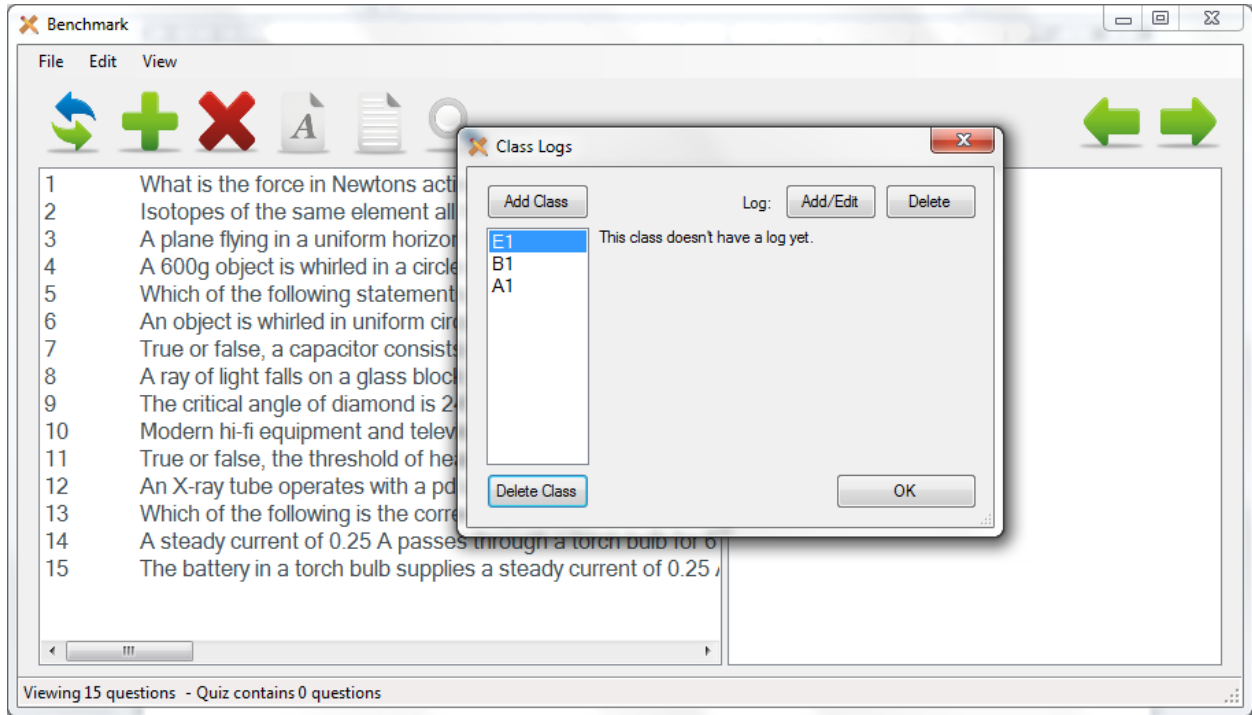
24



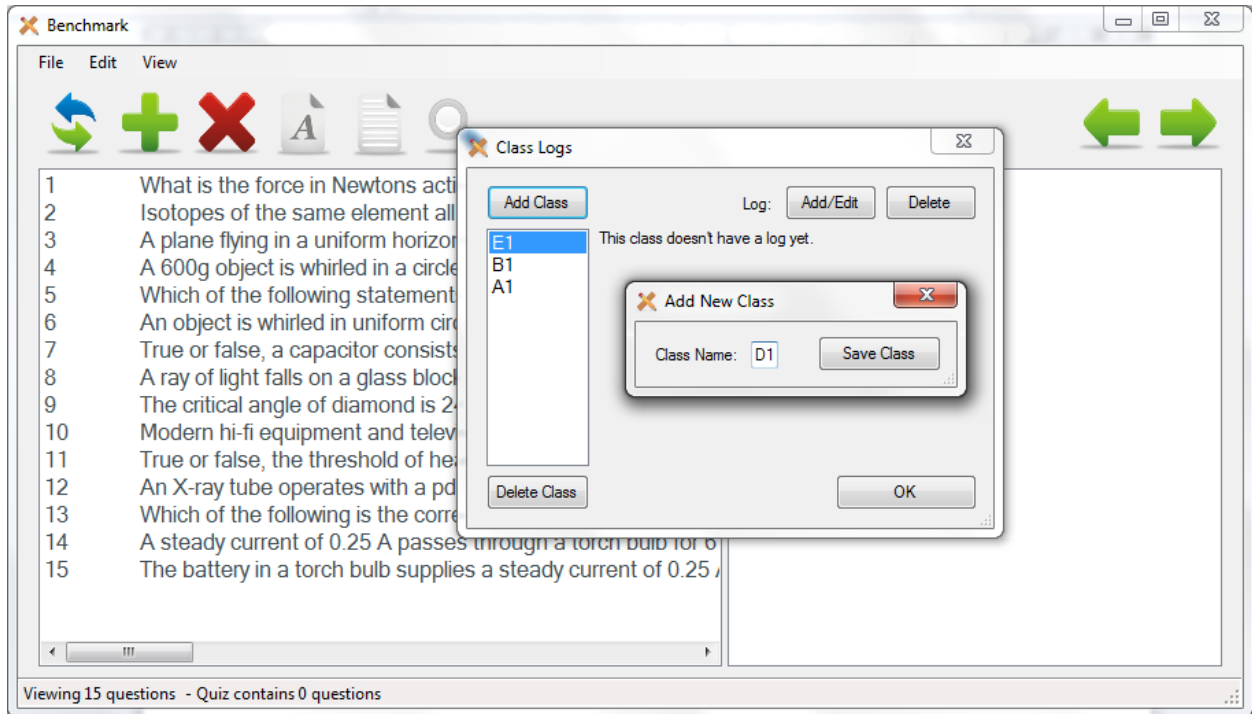
25

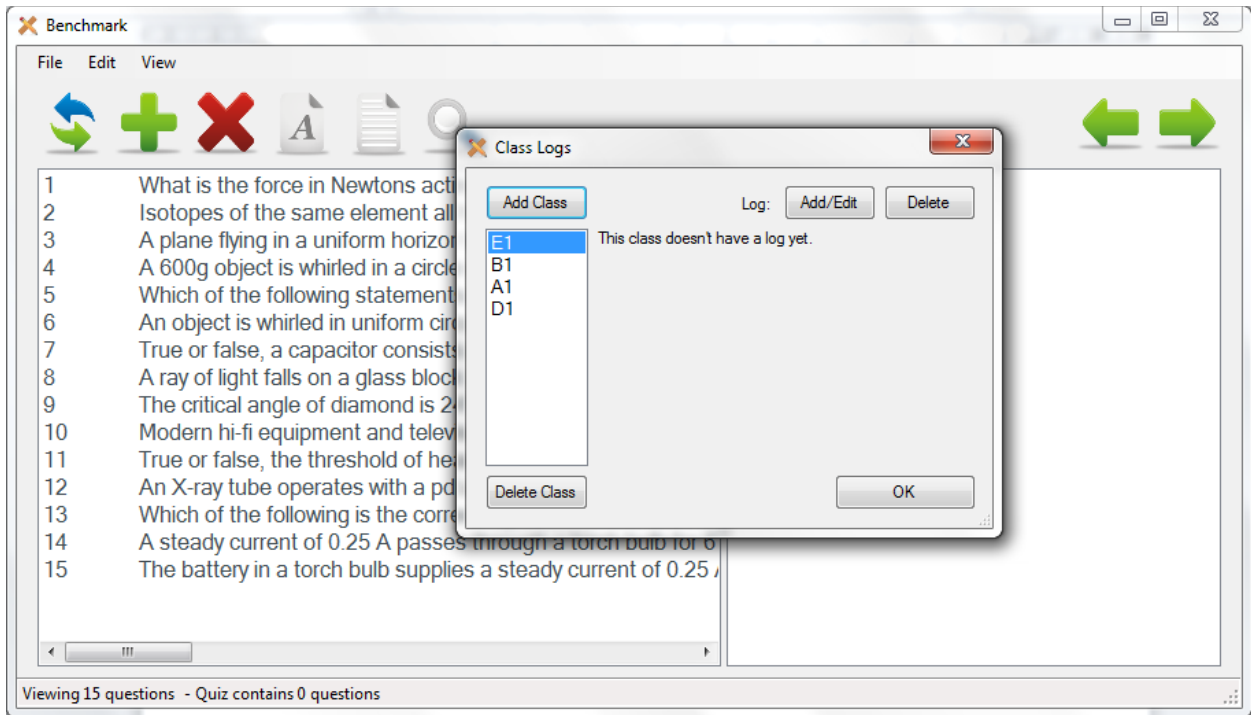


26

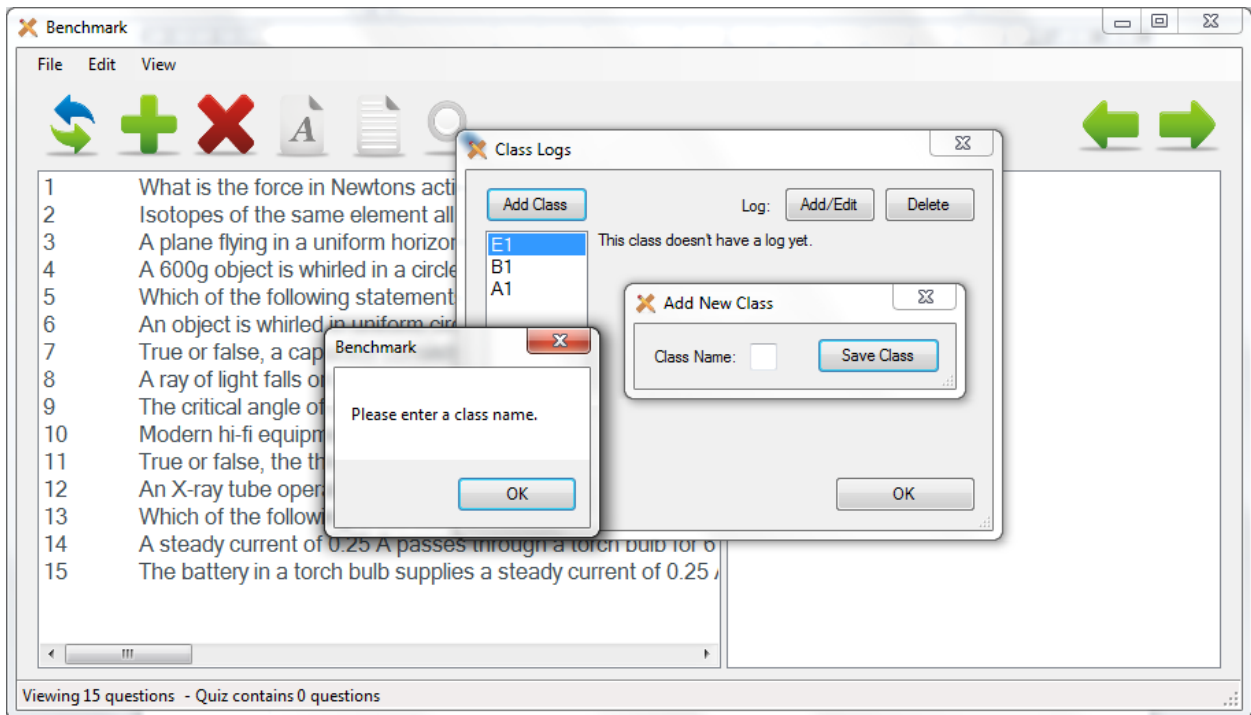


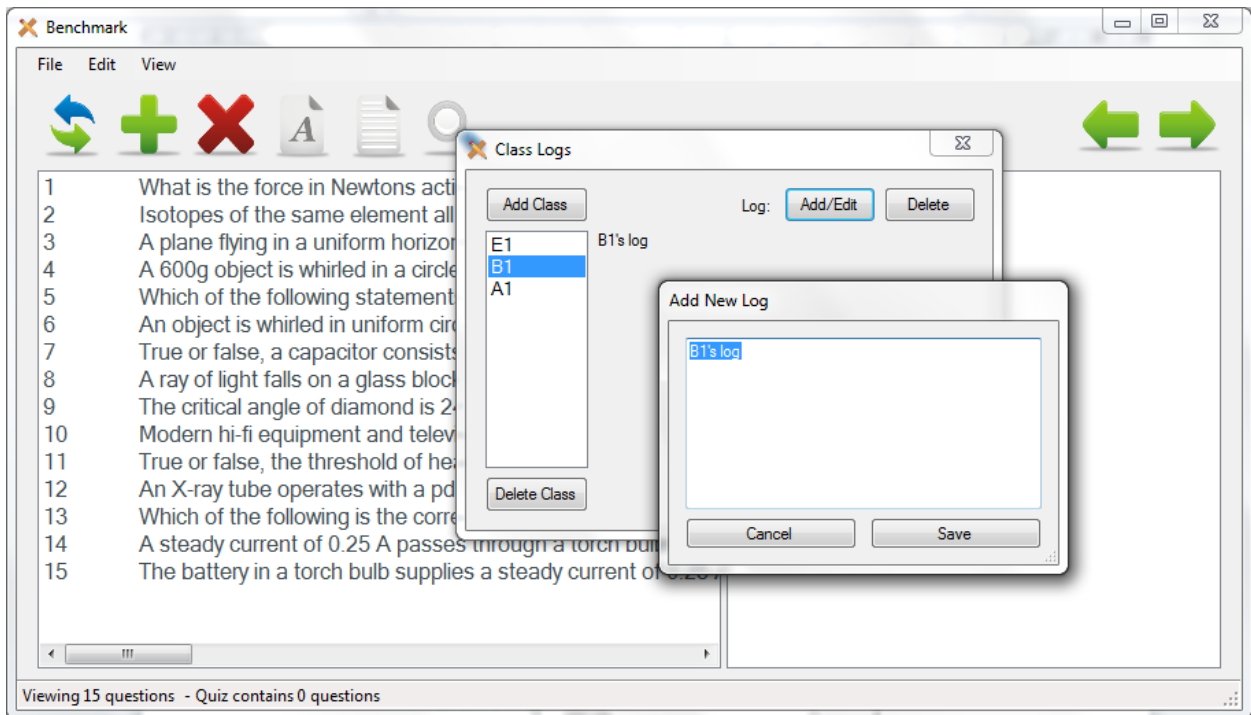
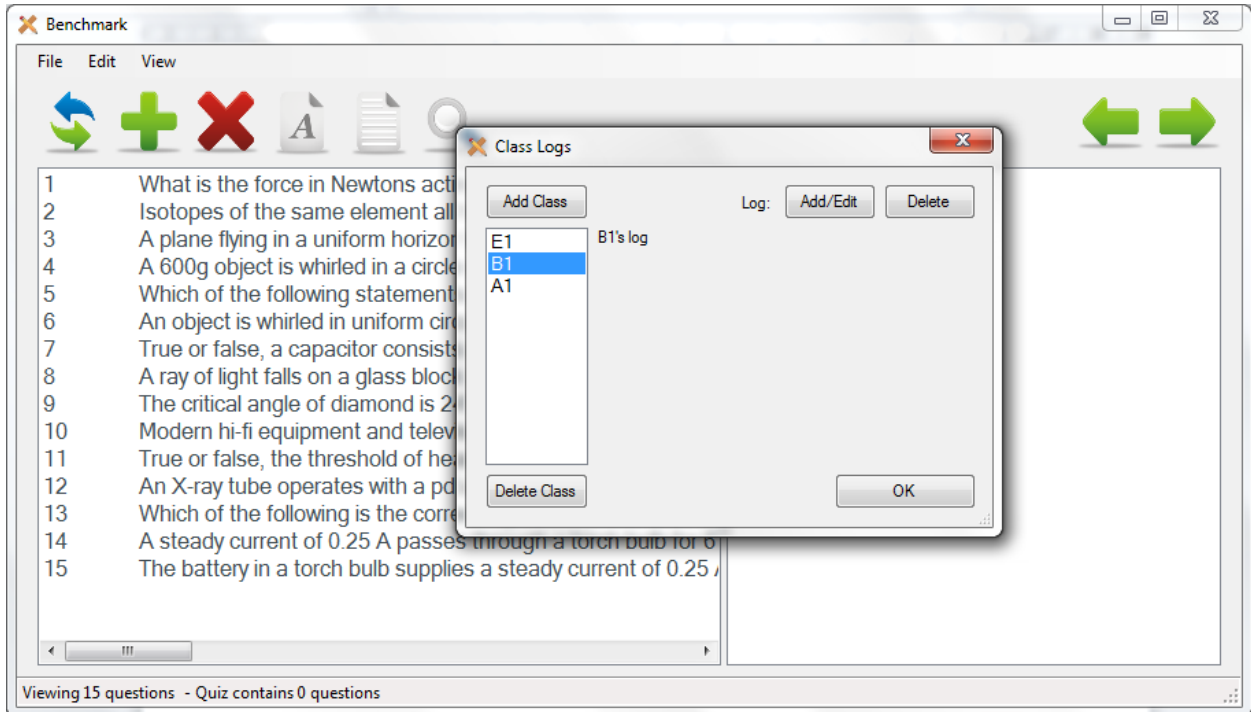
27



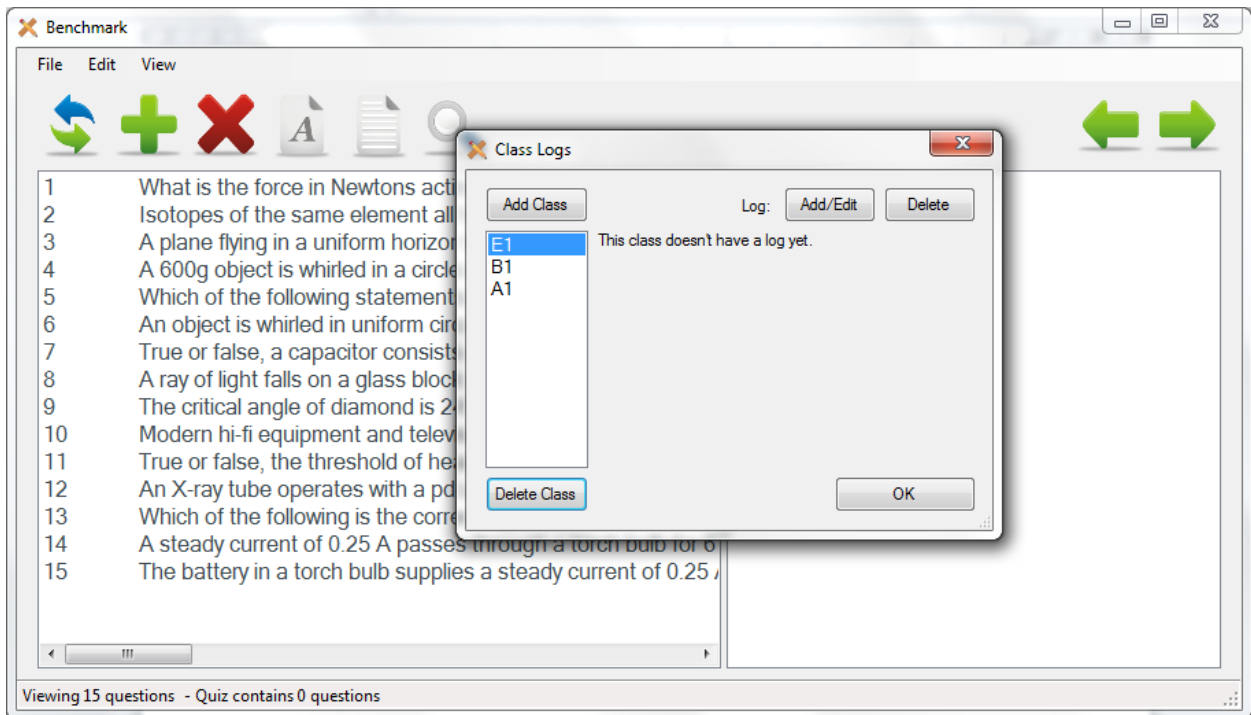
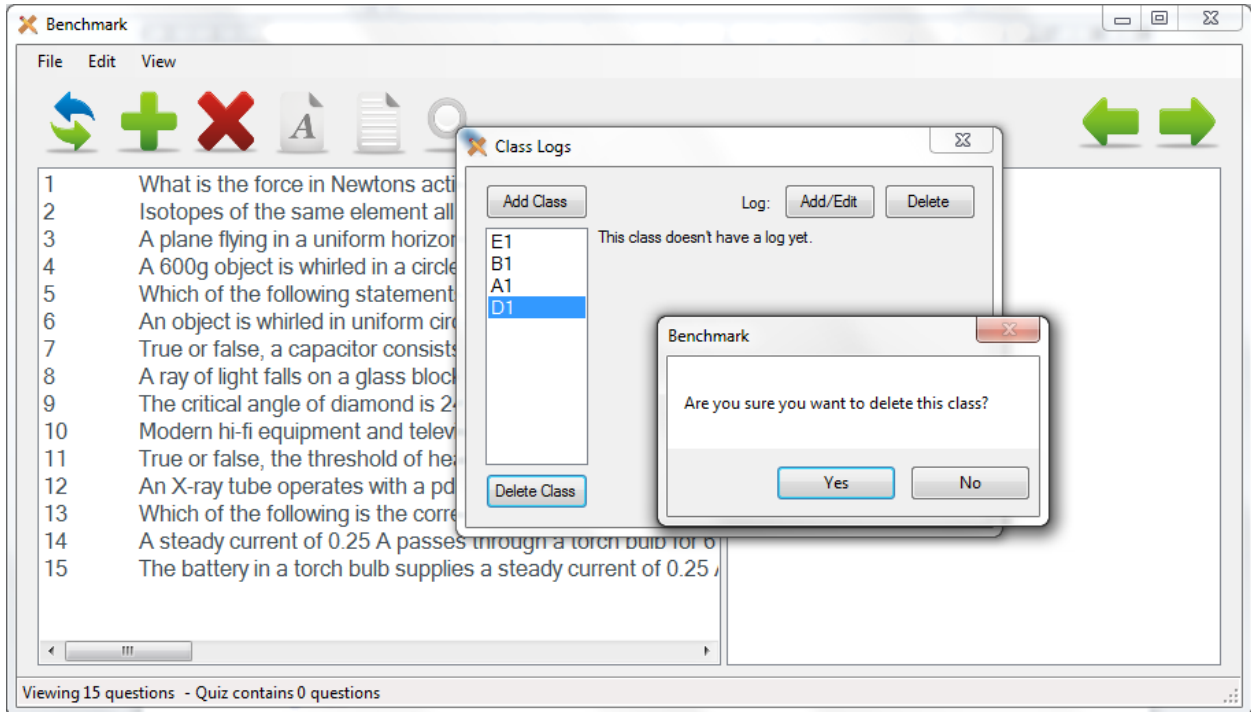


28

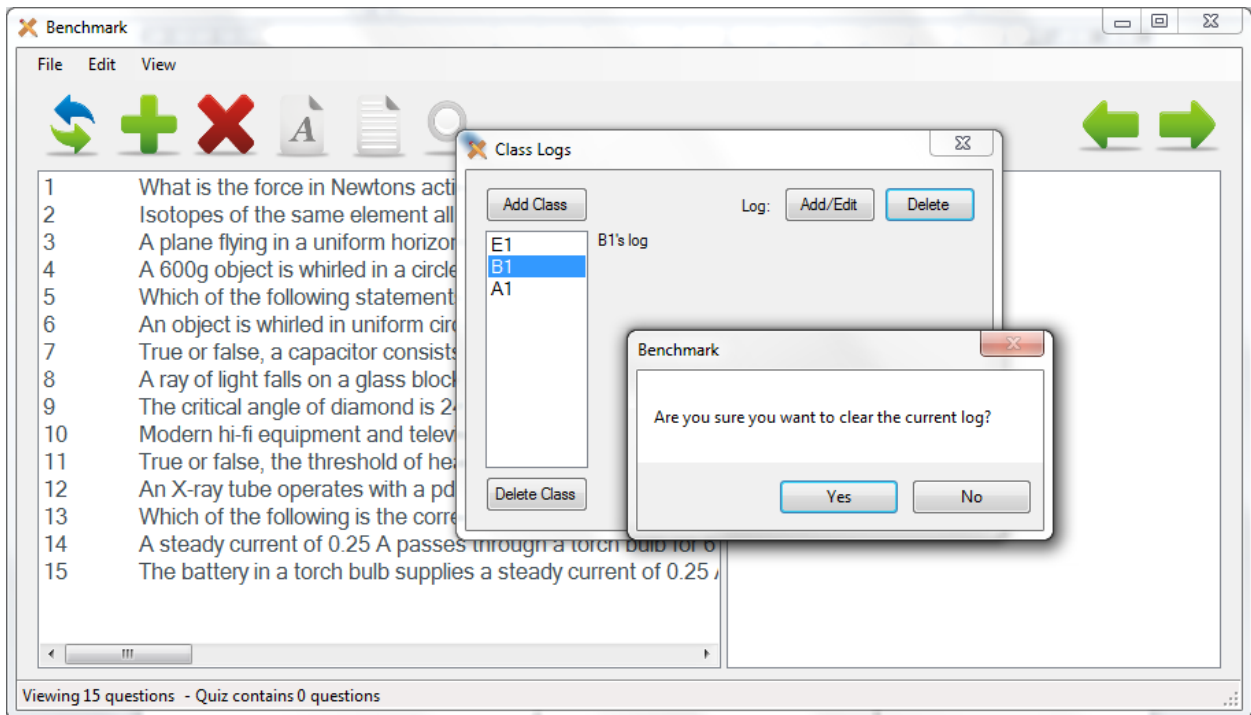
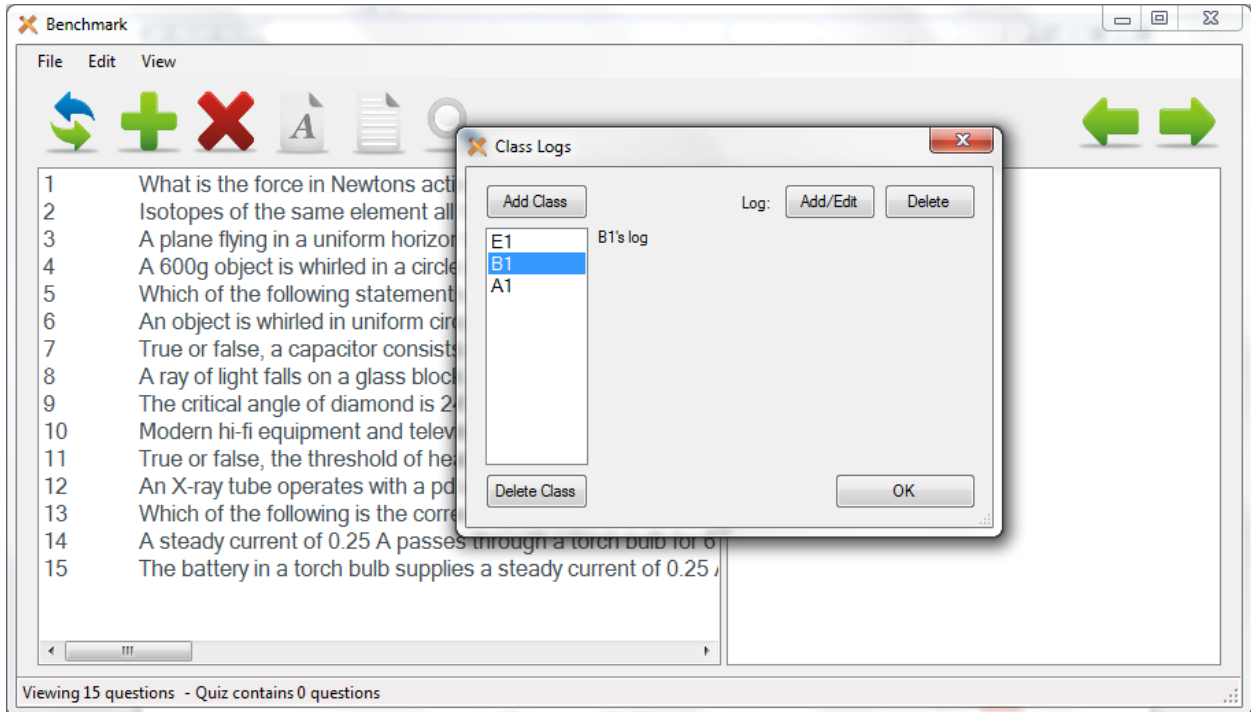




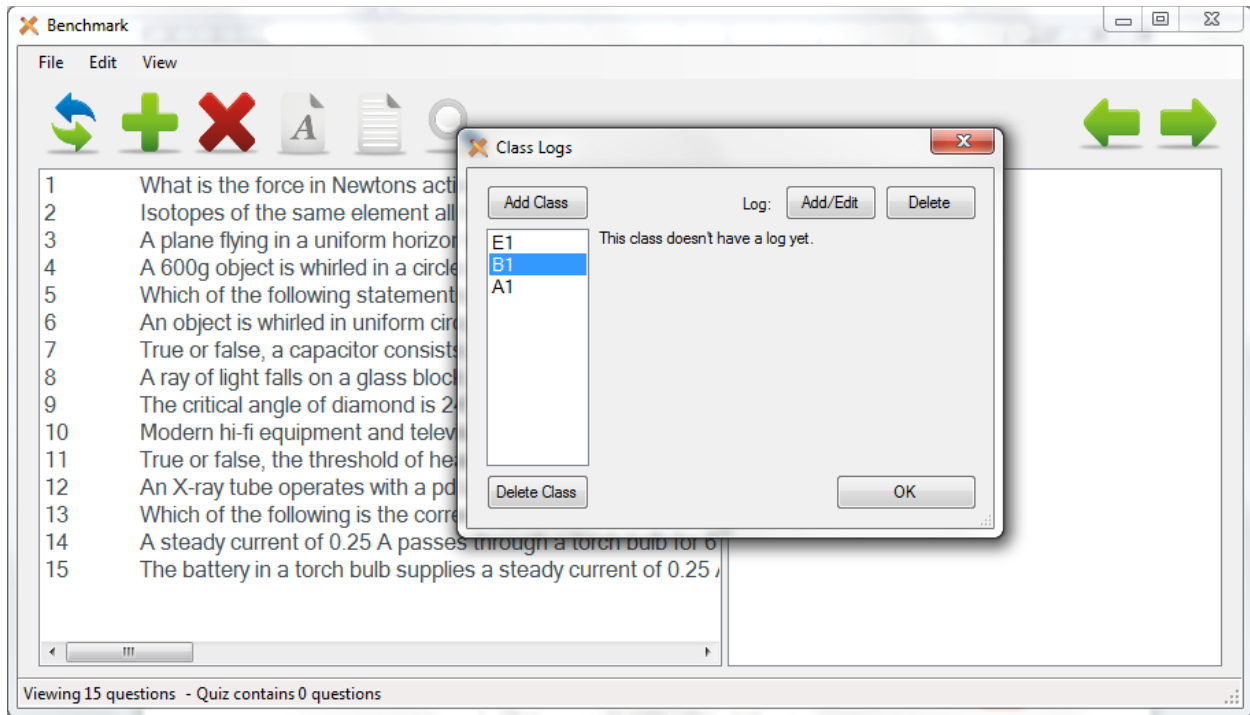
30



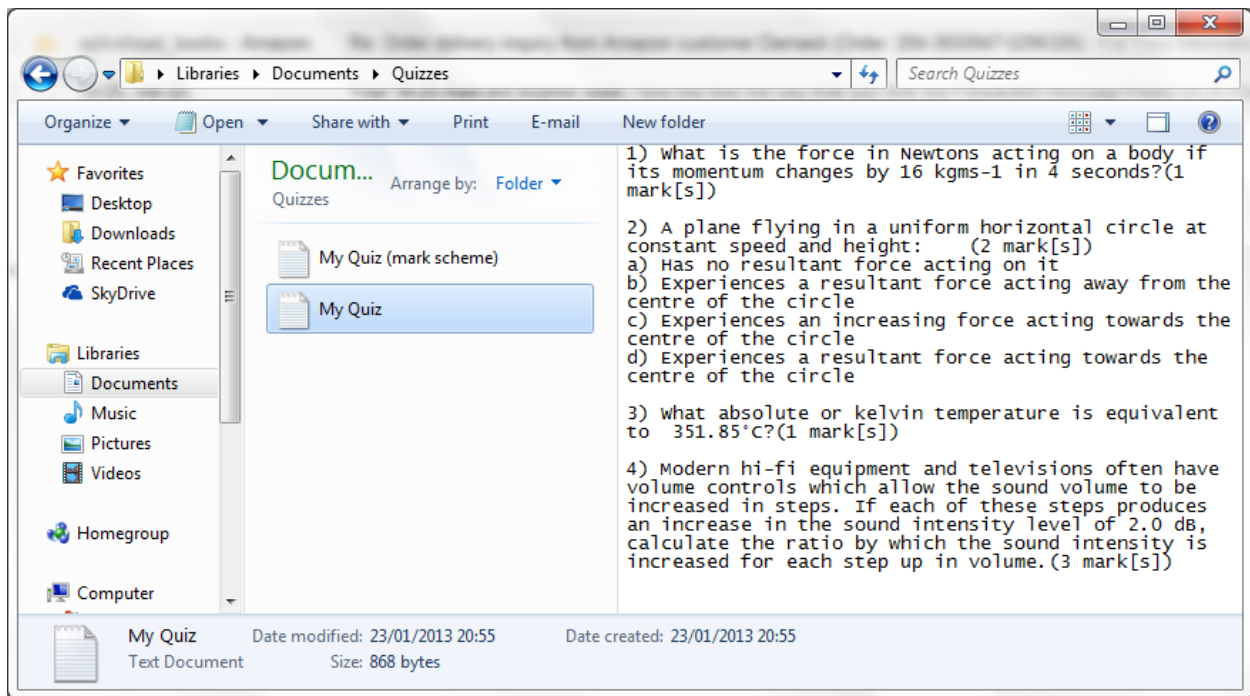
31

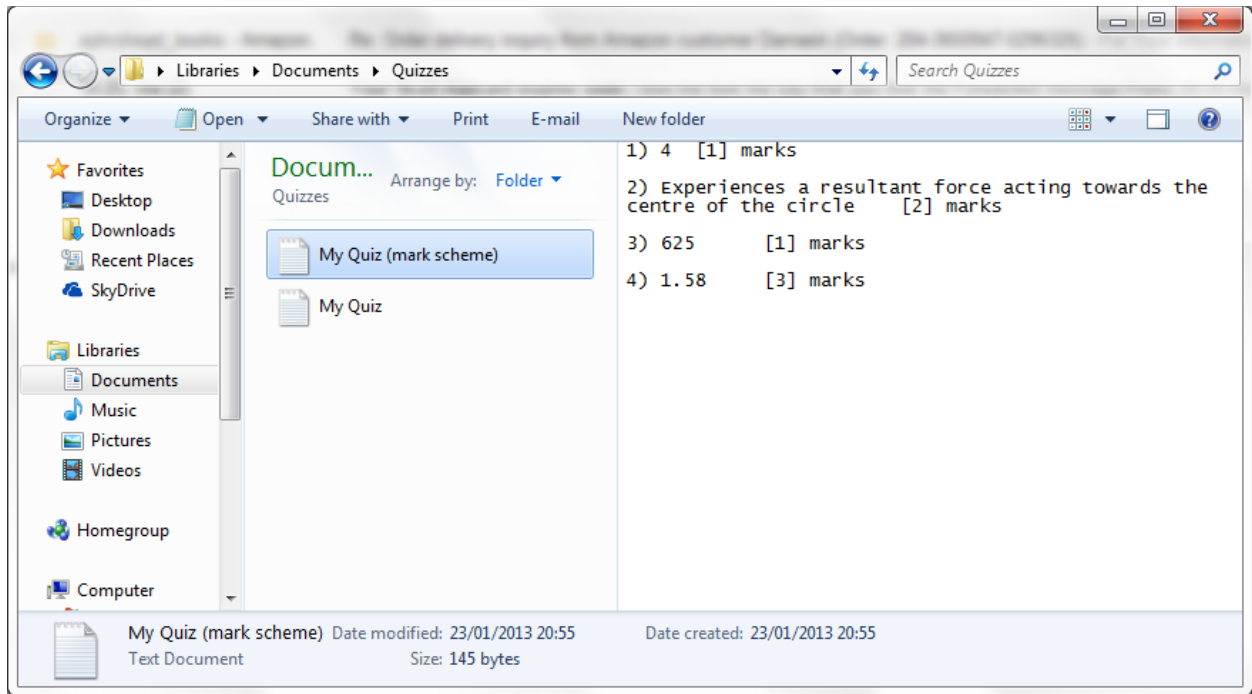




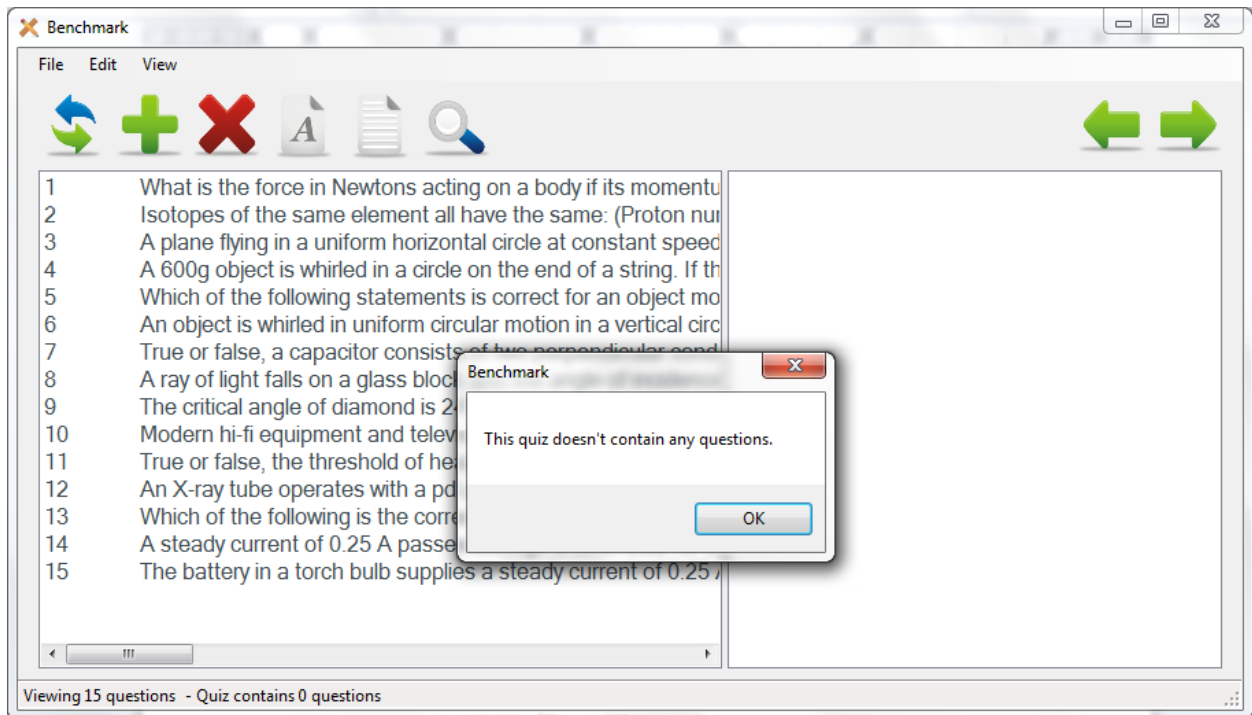


32

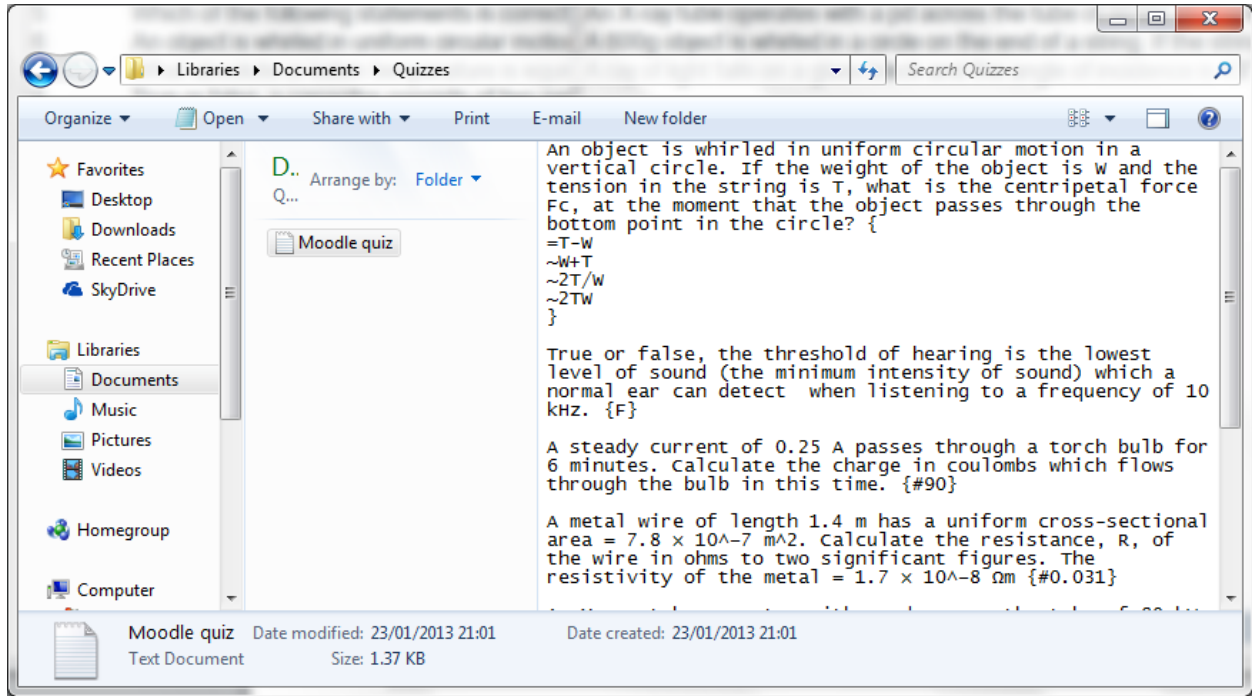




33



34



The screenshot shows a web browser window with the URL <https://moodle.godalming.ac.uk/learning/question/import.php>. The page header includes the Godalming Online logo, the text 'Damask-Computing', and the date 'Wednesday 23 January 2013'. A navigation menu contains links for Timetables, iSkills, email, eFiles, eStream, SeLF, Online Payments-WisePay, and IT FAQs. The breadcrumb trail is 'Home > DAM1 > Quizzes > My first quiz > Import questions from file', with an 'Update this Quiz' button. The main content area has tabs for 'Info', 'Results', 'Preview', and 'Edit', and sub-tabs for 'Quiz', 'Questions', 'Categories', 'Import', and 'Export'. The text 'Parsing questions from import file.' and 'Importing 7 questions from file' is displayed. A list of seven physics questions follows, each in a separate box. A 'Continue' button is at the bottom of the question list. The footer contains the Godalming College logo, a disclaimer 'Godalming College is not responsible for the content of external internet sites.', and the Moodle logo.

You are logged in as DAMASK TALARY-BROWN (Logout) **Wednesday 23 January 2013**

Timetables | iSkills | email | eFiles | eStream | SeLF | Online Payments-WisePay | IT FAQs

Home > DAM1 > Quizzes > My first quiz > Import questions from file Update this Quiz

Info Results Preview Edit

Quiz Questions Categories Import Export

Parsing questions from import file.

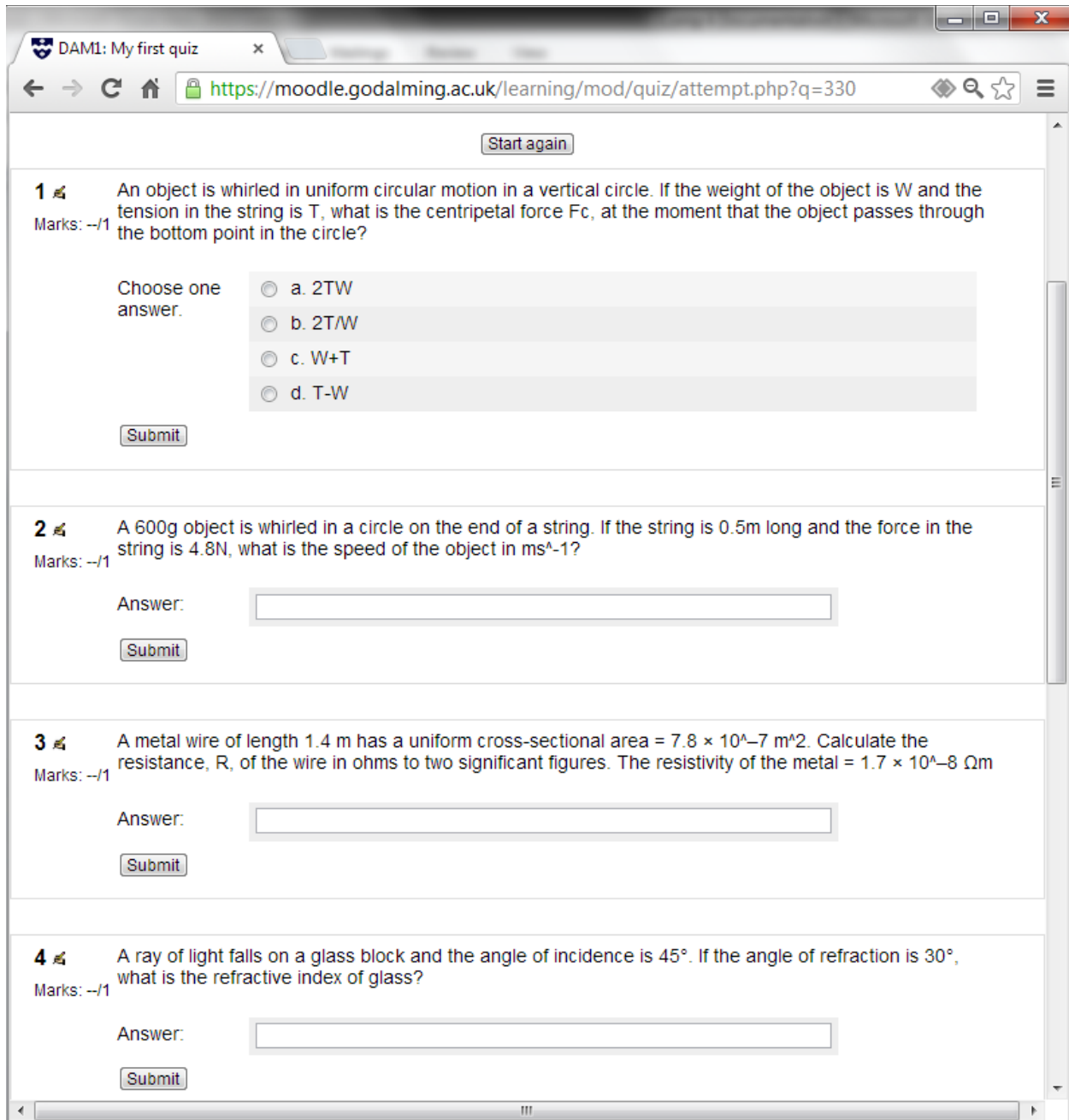
Importing 7 questions from file

1. An object is whirled in uniform circular motion in a vertical circle. If the weight of the object is  $W$  and the tension in the string is  $T$ , what is the centripetal force  $F_c$ , at the moment that the object passes through the bottom point in the circle?
2. True or false, the threshold of hearing is the lowest level of sound (the minimum intensity of sound) which a normal ear can detect when listening to a frequency of 10 kHz.
3. A steady current of 0.25 A passes through a torch bulb for 6 minutes. Calculate the charge in coulombs which flows through the bulb in this time.
4. A metal wire of length 1.4 m has a uniform cross-sectional area =  $7.8 \times 10^{-7} \text{ m}^2$ . Calculate the resistance,  $R$ , of the wire in ohms to two significant figures. The resistivity of the metal =  $1.7 \times 10^{-8} \Omega\text{m}$
5. An X-ray tube operates with a pd across the tube of 80 kV. At the working pd of 80 kV, the anode current was 120 mA. The X-ray tube has an efficiency of 0.70 %. Calculate the rate of production of heat at the anode in kW.
6. A 600g object is whirled in a circle on the end of a string. If the string is 0.5m long and the force in the string is 4.8N, what is the speed of the object in  $\text{ms}^{-1}$ ?
7. A ray of light falls on a glass block and the angle of incidence is  $45^\circ$ . If the angle of refraction is  $30^\circ$ , what is the refractive index of glass?


Continue

This site is maintained and operated by ILT Services. For more information please [contact us](#).

Godalming College Godalming College is not responsible for the content of external internet sites. Powered by:




The screenshot shows a web browser window with the URL <https://moodle.godalming.ac.uk/learning/mod/quiz/attempt.php?q=330>. The page title is "DAM1: My first quiz". There is a "Start again" button at the top. The quiz contains four questions:


**1**  An object is whirled in uniform circular motion in a vertical circle. If the weight of the object is  $W$  and the tension in the string is  $T$ , what is the centripetal force  $F_c$ , at the moment that the object passes through the bottom point in the circle?  
Marks: --/1

Choose one answer.


- a.  $2TW$
- b.  $2T/W$
- c.  $W+T$
- d.  $T-W$

**2**  A 600g object is whirled in a circle on the end of a string. If the string is 0.5m long and the force in the string is 4.8N, what is the speed of the object in  $\text{ms}^{-1}$ ?  
Marks: --/1

Answer:

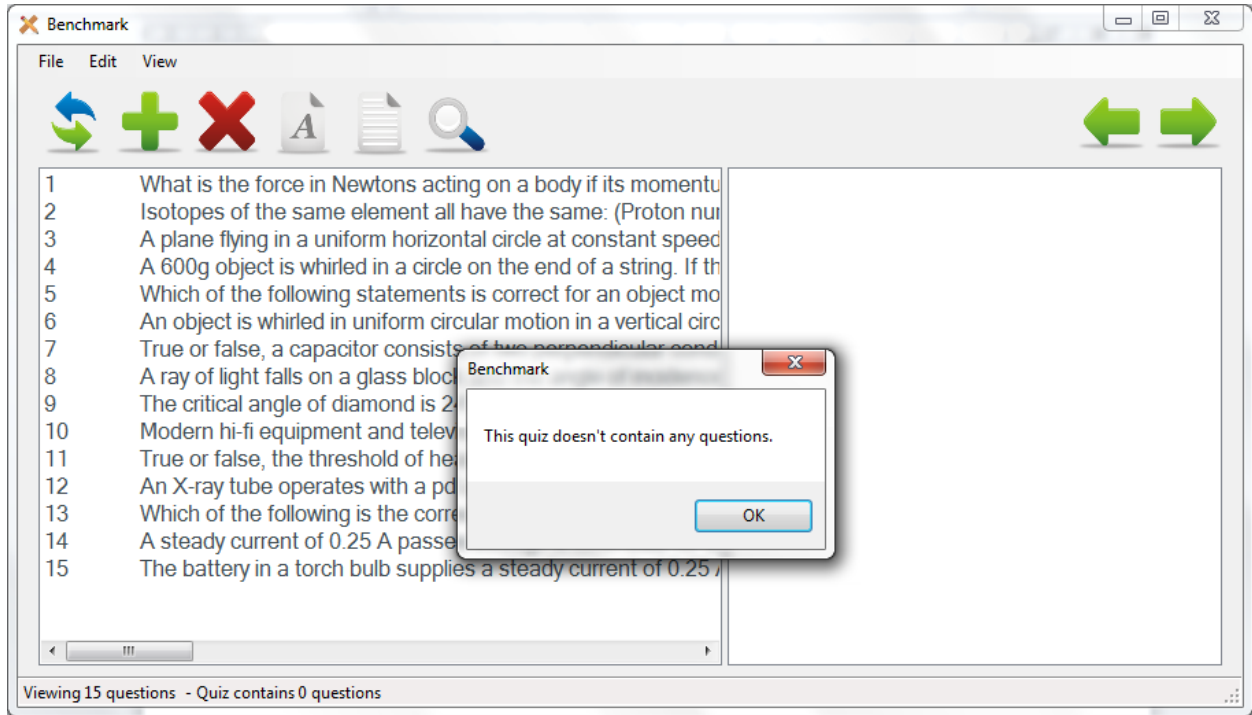
**3**  A metal wire of length 1.4 m has a uniform cross-sectional area =  $7.8 \times 10^{-7} \text{ m}^2$ . Calculate the resistance,  $R$ , of the wire in ohms to two significant figures. The resistivity of the metal =  $1.7 \times 10^{-8} \Omega \text{ m}$   
Marks: --/1

Answer:

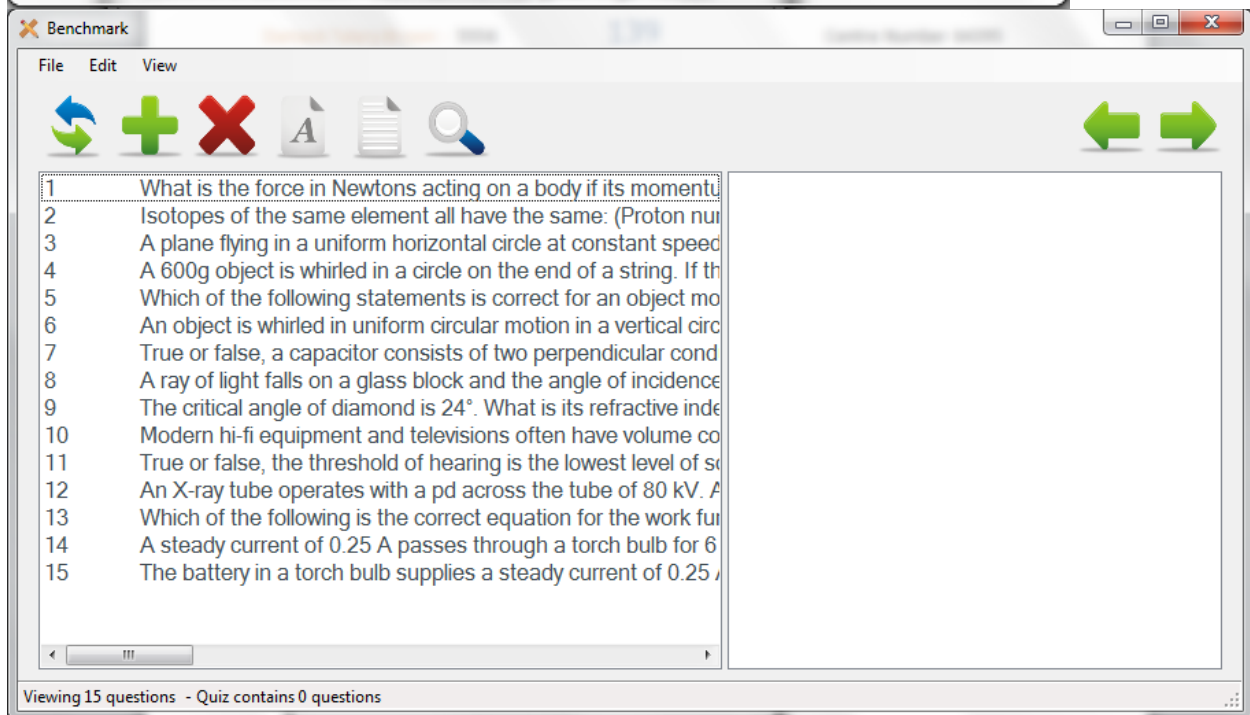
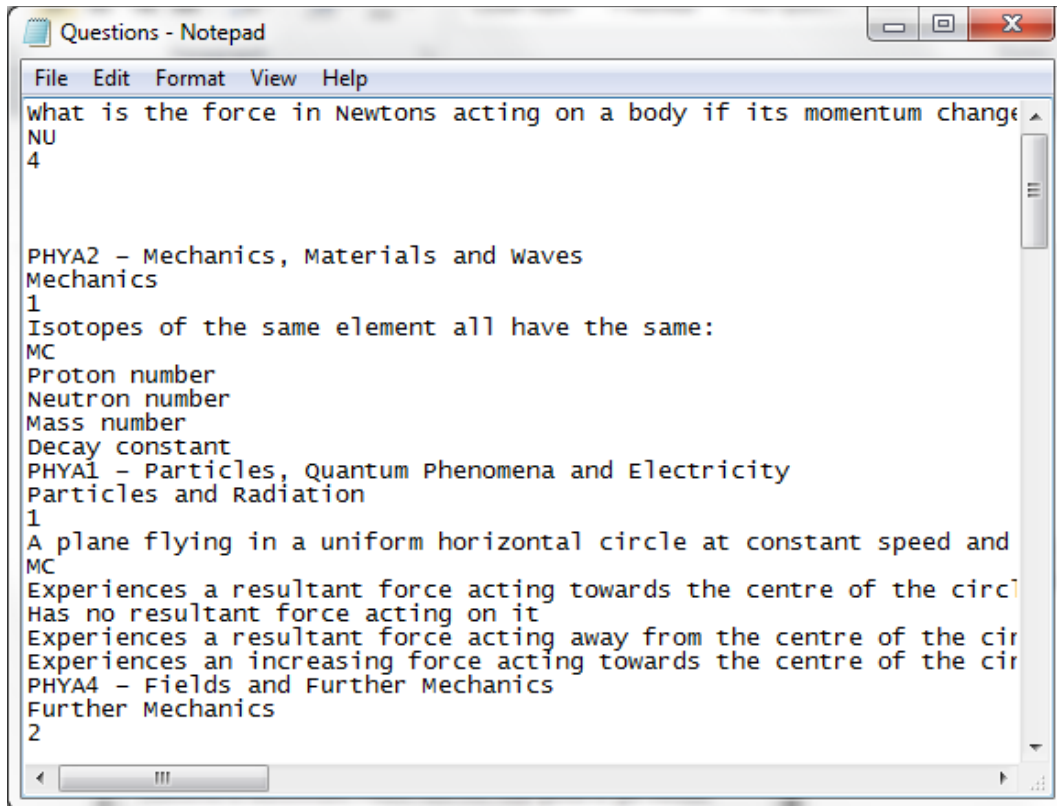
**4**  A ray of light falls on a glass block and the angle of incidence is  $45^\circ$ . If the angle of refraction is  $30^\circ$ , what is the refractive index of glass?  
Marks: --/1

Answer:

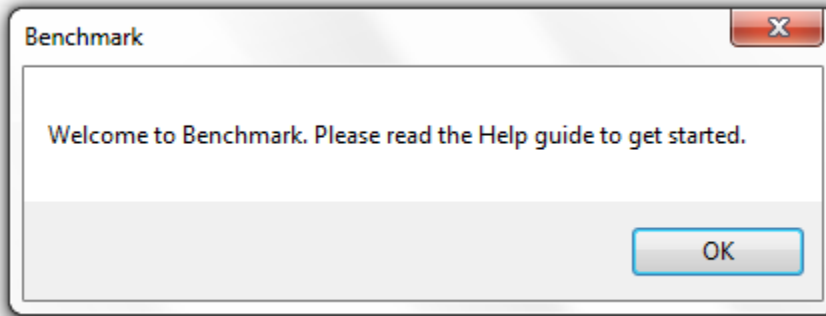
35



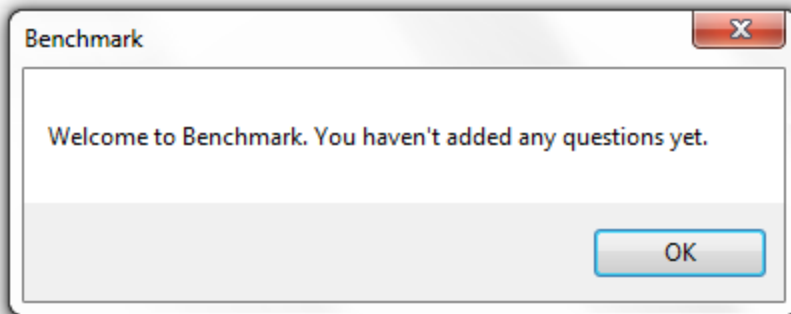
36



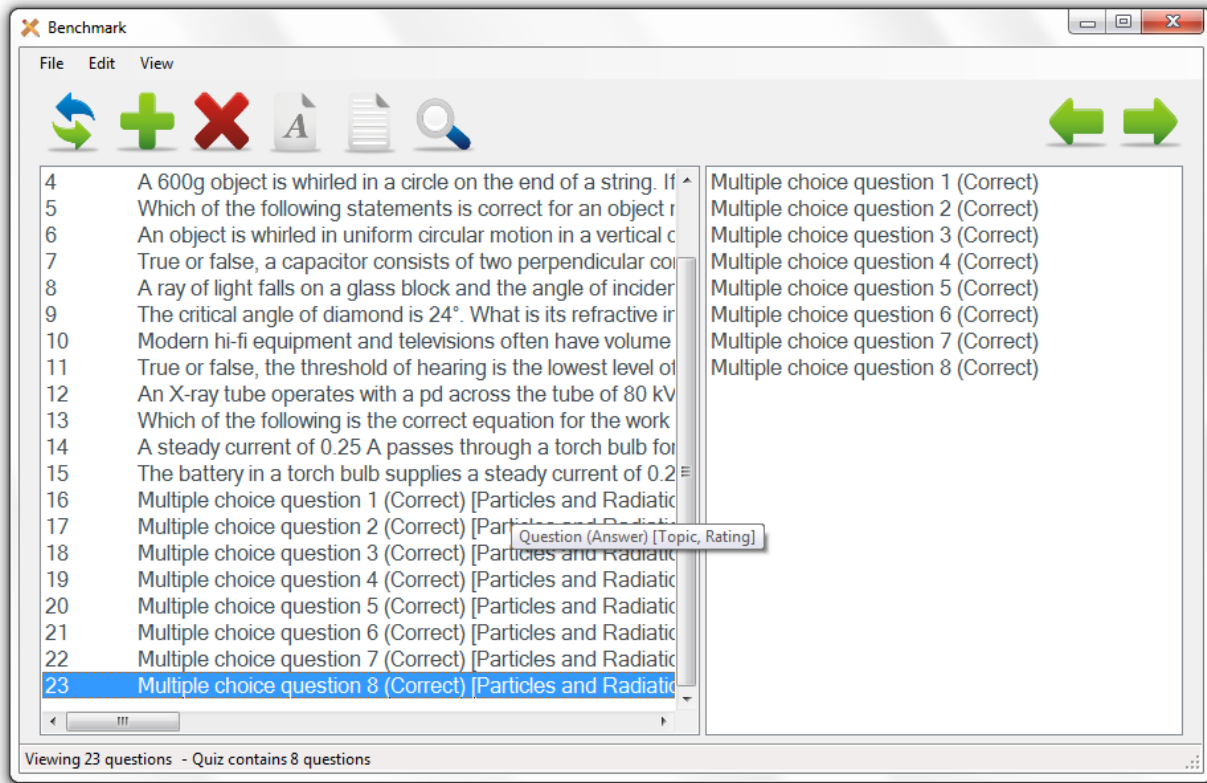
37

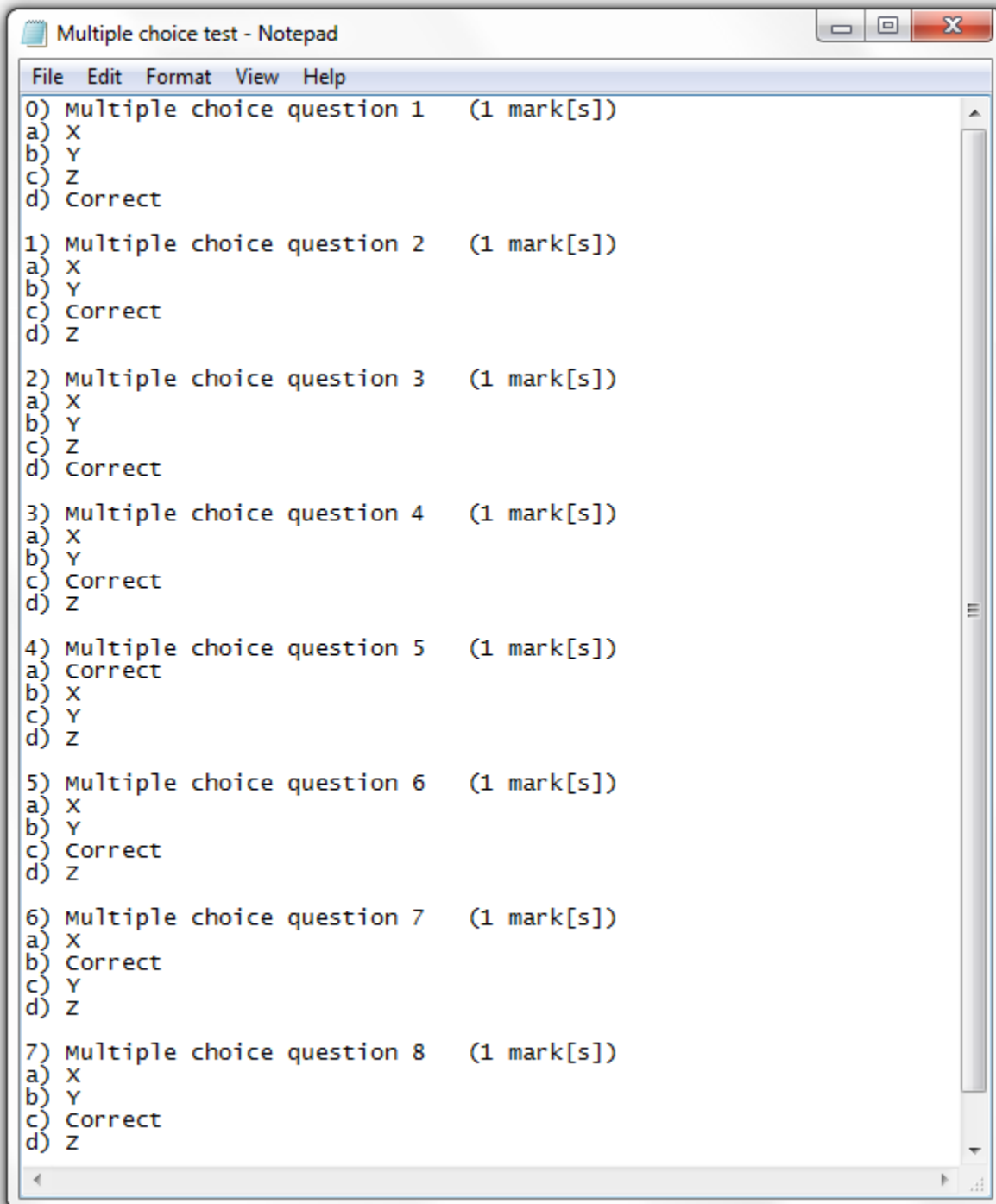


38









```
Multiple choice test - Notepad
File Edit Format View Help
0) Multiple choice question 1 (1 mark[s])
a) X
b) Y
c) Z
d) Correct

1) Multiple choice question 2 (1 mark[s])
a) X
b) Y
c) Correct
d) Z

2) Multiple choice question 3 (1 mark[s])
a) X
b) Y
c) Z
d) Correct

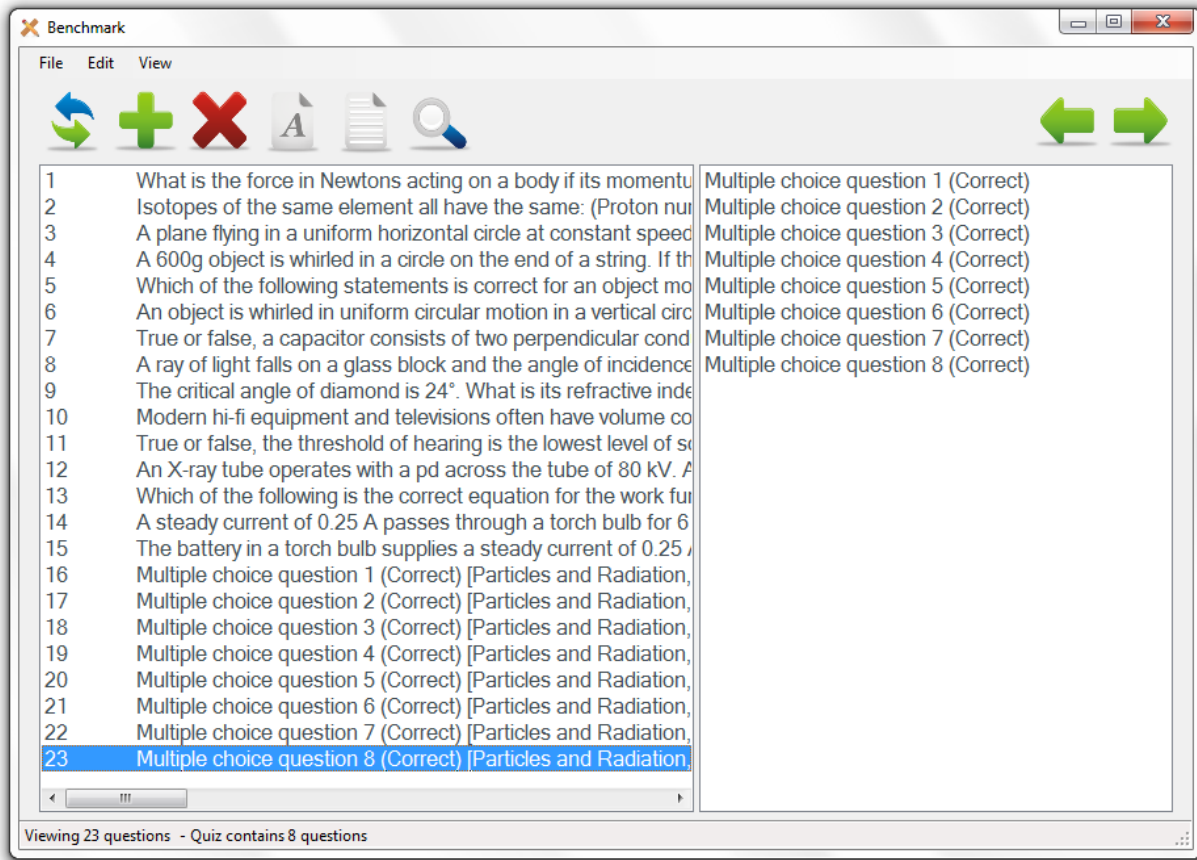
3) Multiple choice question 4 (1 mark[s])
a) X
b) Y
c) Correct
d) Z

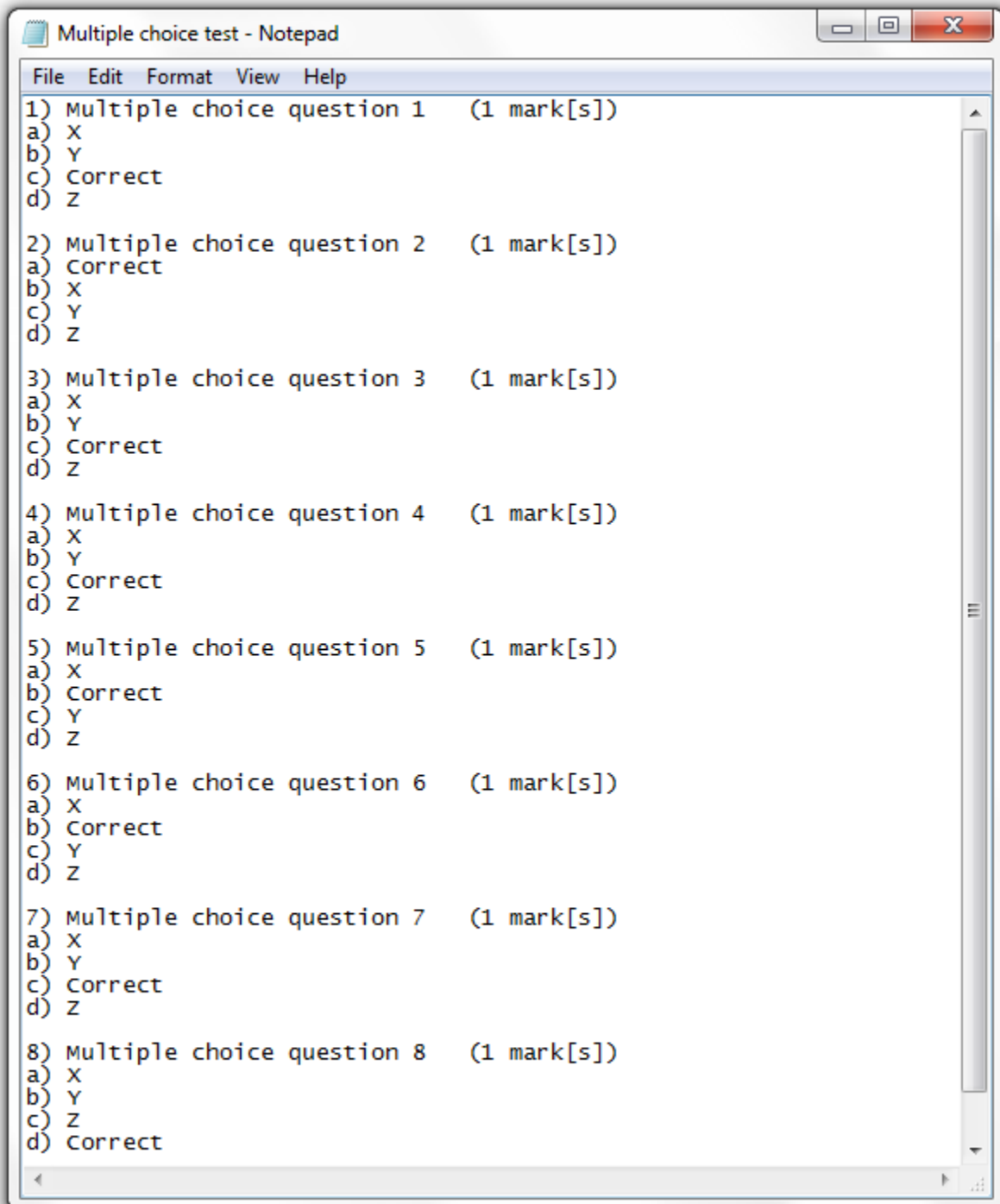
4) Multiple choice question 5 (1 mark[s])
a) Correct
b) X
c) Y
d) Z

5) Multiple choice question 6 (1 mark[s])
a) X
b) Y
c) Correct
d) Z

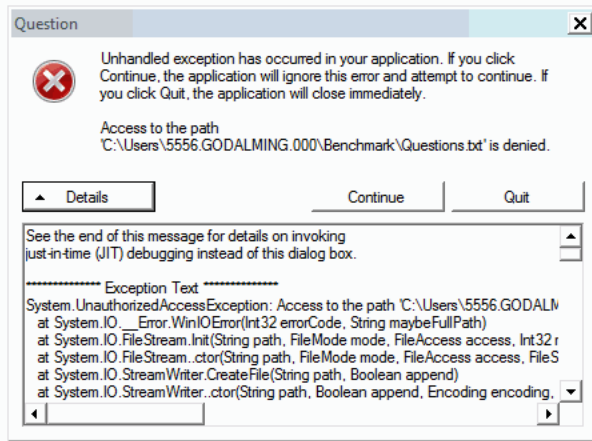
6) Multiple choice question 7 (1 mark[s])
a) X
b) Correct
c) Y
d) Z

7) Multiple choice question 8 (1 mark[s])
a) X
b) Y
c) Correct
d) Z
```

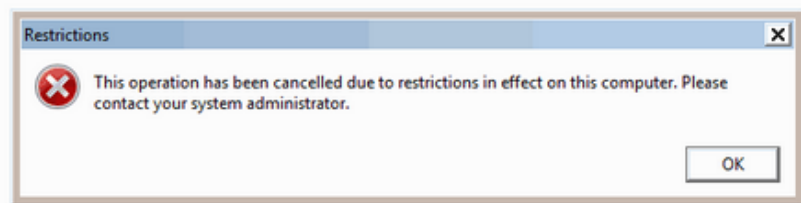




43



44



45

