GCE Computer Science (7517)

The Practical Project

|  |  |  |
| --- | --- | --- |
| Centre number |  | Centre name |
|  |  |

|  |  |  |
| --- | --- | --- |
| Candidate’s full name Bailey Mitchell Elson |  | Candidate number |
|  |  |

|  |
| --- |
| Project title Maths Revision Tool |
|  |

**Contents**

[1 Analysis *(9 marks)*](#_2et92p0)

[1.1 Identification of the problem](#_tyjcwt)

[1.2 Research carried out](#_3dy6vkm)

[1.3 Identification of the prospective user(s)](#_1t3h5sf)

[1.4 Detailed background to the problem](#_4d34og8)

[1.5 Numbered measurable, appropriate specific objectives of the project](#_2s8eyo1)

[1.6 Modelling diagrams](#_17dp8vu)

[2 Design *(12 marks)*](#_3rdcrjn)

[2.1 System design overview](#_26in1rg)

[2.2 Algorithms](#_lnxbz9)

[2.3 Data structures](#_35nkun2)

[2.4 File structure and organisation](#_1ksv4uv)

[2.5 Database design](#_44sinio)

[2.6 SQL queries](#_2jxsxqh)

[2.7 User interface design (HCI)](#_z337ya)

[2.8 Hardware selection/design](#_3j2qqm3)

[3 Technical Solution *(42 marks)*](#_1y810tw)

[3.1 Code listing](#_4i7ojhp)

[4 System testing *(8 marks)*](#_2xcytpi)

[4.1 Test plan](#_1ci93xb)

[4.2 Annotated screenshots of test results](#_3whwml4)

[5 Evaluation *(4 marks)*](#_2bn6wsx)

[5.1 Comparison of project performance against the objectives](#_qsh70q)

[5.2 Effectiveness of the solution](#_3as4poj)

[5.3 Analysis of user feedback](#_1pxezwc)

[5.4 Possible improvements](#_49x2ik5)

[6 Appendix](#_2p2csry)

[6.1 User feedback (authenticated by assessor)](#_147n2zr)

# Analysis *(9 marks)*

## Identification of the problem

The Hertfordshire and Essex’s high school teaches mathematics as part of the syllabus for all year 7 to year 11 students and has mathematics and further mathematics as options for sixth form students in year 12 and 13. Each year the school gains over one hundred new year 7’s all from different primary schools around the school district. This new year group all have ranging mathematics abilities and have been all taught to different levels. Due to this each year the mathematics department has to have a way of finding out all the students’ abilities in a range of different mathematic concepts.

Currently the students in the new year group have baseline tests so the teachers can find out the students current level of working. However these tests have to have a range of different level questions meaning the students with the higher ability has to answer questions far too easy for them and the students at the lower ability have to try and attempt questions outside of their abilities.

My program will change this by having a range of questions at different levels for the student t answer, but instead of having to sit through the questions that are far too easy for them or outside of their capabilities the students can choice a level that is inside of their capabilities but still pushes them intellectually.

## Research carried out

**Current systems:**

Firstly I looked into the programs that the students currently use. The main program that they use in maths is MyMaths. MyMaths is an online maths revision tool, the students each have personal log in credentials which when they log in gives them access to tasks set by the teacher. They can also use MyMaths to revise different topics even if the teacher hasn’t set them a task. I have personally used MyMaths in the past and at times have found the layout confusing and have also had some issues with the answer checking system.

**User interview (student):**

For the student interview I arranged a meeting with a group of year 7 students to ask them what they liked and disliked about the current system and what they would want from my program. The feedback that I got splits up into three main sections

**Log in credentials** – currently to log in to the MyMaths system you have a username consisting of three random characters and a password also consisting of three random characters. The students do not like this system as they find it difficult to remember their log in credentials and often have to write them down and check them before they log on. This means that if they don’t have their log in credentials with them they can’t access the program. In my program they would like to choose their own log in credentials as then they will be able to remember them easier and will be able to access the program without having to check the login credentials.

**Layout** – with the MyMaths system multiple questions are loaded on one page. Some of the students say that at times it’s confusing to which piece of information goes with which question. They also say it’s unclear what difficulty the question is as the program mixes easier questions on the same page with the harder higher end questions. The students would prefer a single question to a page with a clear indication of what level difficulty they are on.

**Answer checking** – the students complained about the answering system on MyMaths. They said that they would at times enter the right answer but the program would mark it as wrong and then show them the correct answer which matches the answer they inputted. It also told you to write answers to a certain number of decimal places and then the correct answer was written a different number of decimal places. I have also found this issue when using MyMaths in the past. To students wanted a clear indication of how they should answer their questions (e.g. to how many decimal places) and they also want the answer system to correctly identify when they have got the answer right.

**User interview (teacher):**

For the teacher interview I arranged a meeting with a group of mathematics teachers to talk about what they currently liked and disliked about the current system and what they want from my program. There feedback splits up into three main sections.

**Student score viewing** – currently when they look at a student’s scores they can only see a percentage score of the whole lesson. They would prefer it if my program would split the scoring up into sections based on the difficulty of the question. Therefore making it easier for the teacher to see what level the student is working at

**Accessibility** – they would like a simple GUI that is easy to navigate and understand. All they want from the GUI is to be able to log in and then access a student’s score. The current system they believe is too complicated and complex for what they want to do.

**Multiple questions** – the final piece of feedback I got was that the teachers wanted multiple questions of the same type so it’s easier to tell whether the student actually knows how to answer the question or whether it was a lucky guess.

## Identification of the prospective user(s)

My program will have two main users, the first being the year 7 mathematics students and the second being the mathematics teacher.

The year 7 students will have access to the start page, the login page, the main page, the new user page and the question page. On the start page they will have to choice to select student or teacher where they will select student. This will then lead to the login page where they can either log in using existing account details or select the new user page where they can create a new account. Once they are logged in the main page will load where the student can navigate between the five lessons and all three levels on each lesson. Each lesson they choose will load a question page.

The teachers will have access to the start page, the login page, the new user page and the teacher page. On the start page they will have to the choice between student and teacher. Once they select teacher the login page will load where they can either log in using existing account details or select the new user page where they can create a new account. Once they have successfully logged in the teacher will load.

Both the year 7 students and the mathematics teacher are competent with computers. The year 7 students have experience with computers from there other school lesson where they use other computer based revision tools. The teachers have experience with computers

## Detailed background to the problem

The mathematics department of hert’s and essex’s school need to be able to test a whole year group of students as they join the school so they can find out all the students abilities.

To do this is will be developing a program that will test the students in the basic skills they should already know. The program will focus on the topics, geometry, algebra, sequences and graph theory. To do this there will be five different lessons types. The first is algebraic equations, which will be tested by randomly generated equations and asking the user for the solutions.

**Feasibility study:**

Cost: the total cost of the project is £0. This is due to the fact that the software I intended to use is python which is free open source software. Everything else that is needed for this project is free libraries that can be imported into python.

Budget: there is no budget for this project

Time: I plan for three weeks of design, this will give me enough time to talk to the users and design a system that they want. This also gives me time to act on any feedback they have. By the end of the design phase I will have a complete plan of the system layout as well as designs for all the GUI pages. After there will be 6 weeks of development and coding. This gives me enough time to make the program as well as basic testing that the program works. Finally there will be 3 week of testing where I will thoroughly test all aspects of the program and document all tests. Included in these three weeks is any needed code alterations to fix any bugs found in the testing.

Skills: I have the required computing and coding skills to successfully develop this program to the necessary level on the chosen software. I have experience in coding on python and will be able to develop a program that meets all the users required objectives.

Hardware: the current hardware in school is capable of running python. The only change needed is to install the matplotlib library onto the python software. This is a simple process and shouldn’t cause any issues.

Training: after my program is developed the students and teachers shouldn’t require any training to use the program as it plan for it to be an easy to navigate and understand. This means that the GUI’s should be clearly laid out.

Technical feasibility: I believe that it is technically possible to create this program with these requirements. The python software that I plan to use is complex enough to be able to complete all the required objectives without putting strain on the software.

## Numbered measurable, appropriate specific objectives of the project

Objective 1 – a new user page GUI that will generate a new user account only if the username is unique and isn’t already in use

Objective 2 – a new user page that will show an error message if the password and confirm password sections do not match

Objective 3 – a new user page that will display an error message if the username is already in use

Objective 4 – the new user page will store the new user accounts details (first name, last name, username, hashed password, and score 0 for all lesson and levels for students and first name last name username and hashed password for teachers)

Objective5 – the program will hash the users password before saving it to the csv file

Objective 6 – a fully functioning login page that will log the user in if a correct username-password combination has been entered.

Objective 7 – the login page will display an error message and will not allow the user onto the rest of the program if an incorrect username-password combination has been entered

Objective 8 – a main page that the user can use to navigate to different lessons

Objective 9 – a five different lesson all with three levels that randomly generate questions

Objective 10 – all three levels on the graph lesson will produce a graph for the user to see

Objective 11 – for the lessons that require an image (triangles and area), an image will load on the GUI

Objective 12 – the user’s sores to be stored in a csv file

Objective 13 – the user’s scores to be updated inside the csv file

Objective 14 – to be able to check the user’s answer against the actual answer for all the levels inside the five lessons

Objective 15 – to be able to log back out of the program and this causing the login page to reload

Objective 16 – level 1 of algebra will generate random questions in the format ax = b (a and b representing integer values) which are solvable

Objective 17 – level 2 of algebra will generate random questions in the form ax + b = c and ax – b = c (a,b and c representing integer values) which are solvable

Objective 18 – level 3 of algebra will generate random questions in the form ax + b = cx + d,

ax – b = cx – d, ax + b = cx – d and ax – b = cx + d (a,b,c and d representing integer values) which are solvable

Objective 19 – level 1 of area will generate random questions, asking for the area of a rectangle or square with both side A and side B being randomly generated integers, which are solvable

Objective 20 – level 2 of area will generate random questions, asking for the area of a circle with the radius being a randomly generated integer, which are solvable

Objective 21 – level 3 of area will generate random questions, asking for the area of a trapezoid with both side A, side B and the height being randomly generated integers, which are solvable

Objective 22 – level 1 of triangles will generate random questions, asking for the third angle in a triangle with the other two angles being randomly generated integers which are solvable

Objective 23 – level 2 of triangles will generate random questions, asking for the value of the third side of a right angle triangle where the other two sides are randomly generated integers, which are solvable

Objective 24 – level 3 of triangles will generate questions, asking for the value of the third side of a triangle where the other two sides and two angles are randomly generated integers, which are solvable

Objective 25 – level 1 of sequences will generate random questions, in the form nth term = ax +/- b what is the c term (a,b and c representing randomly generated integers), which are solvable

Objective 26 – level 2 of sequences will generate random questions, in the form nth term = +/- b what is the c term (a,b and c representing randomly generated integers), which are solvable

Objective 27 – level 3 of sequences will generate random questions, in the form [a,b,c,d,e] is a sequence what is the nth term in terms of x (where [a,b,c,d,e] represent a random sequence), which are solvable

Objective 28 – level 1 of graphs will generate random questions, producing a graph showing a linear function and asking what does y equal when x = a (a representing a randomly generated integer), which are solvable

Objective 29 – level 2 of graph will generate random questions, producing a linear graph and asking is this graph ax +/- b (a and b represent randomly generated integers), which are solvable

Objective 30 – level 3 of graph will generate random questions, producing a linear, quadratic or cubic graph and asking what type of graph I this, which are solvable

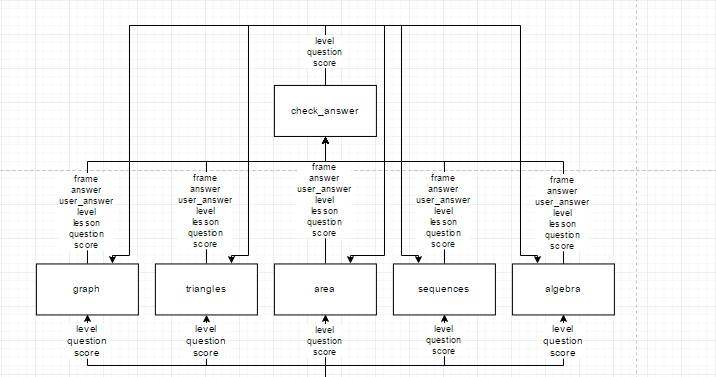
Objective 31 – a teacher’s page will load when a teacher successfully logs in

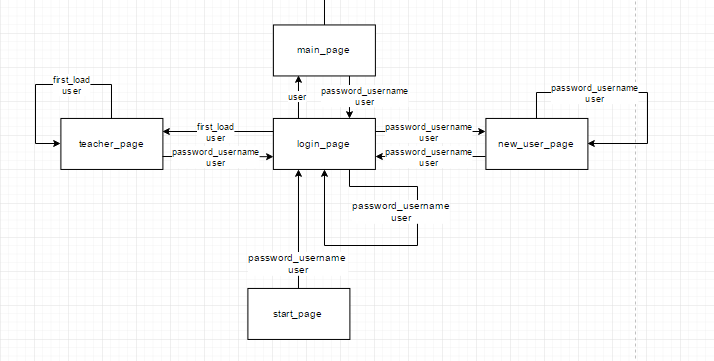
Objective 32 – if a correct students first and last name are entered into the teacher page their score will successfully be displayed

Objective 33 – if the student’s name that is entered doesn’t match any of the students in the csv file then an error message will be displayed

## Modelling diagrams

A diagram to show the variables passed between the different classes in the program.





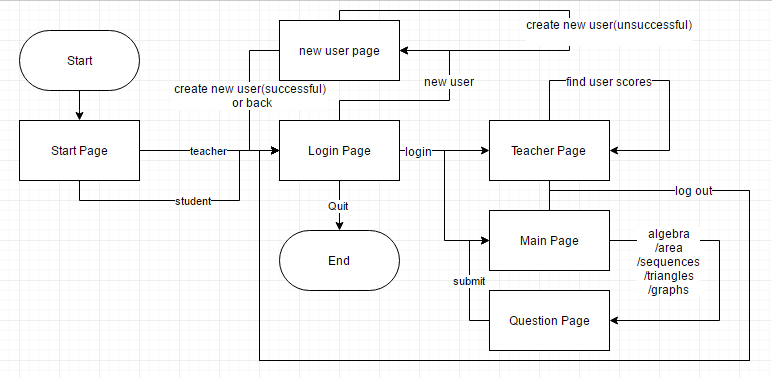
# 

# Design *(12 marks)*

## System design overview

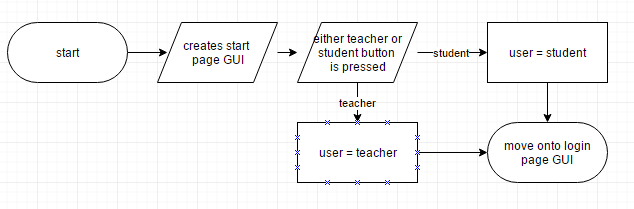
Flow chart of the interaction between GUI pages

A flowchart showing the interaction between the different GUI pages. This diagram shows which GUI page is loaded when each button is pressed.



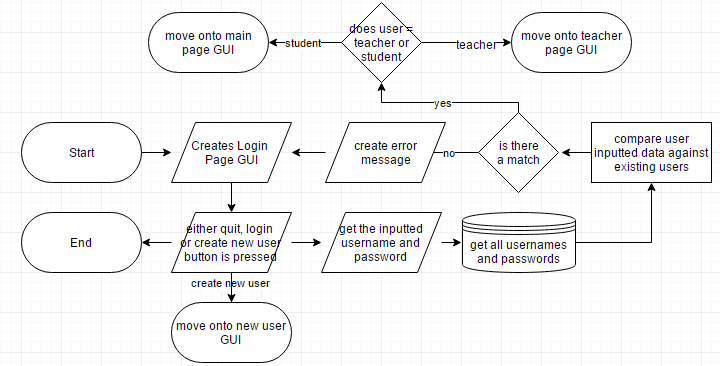
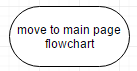
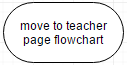
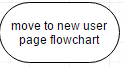
Start page flowchart

The flowchart showing the actions taken on the start page and what happens when each button is pressed.



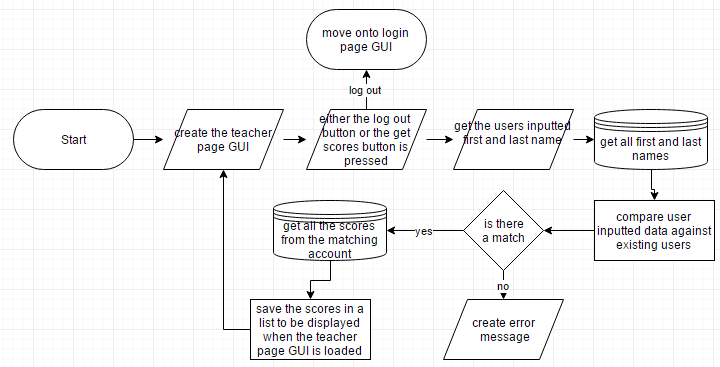
Login Page flowchart

The flowchart showing the actions taken on the login page and what happens when each button is pressed.



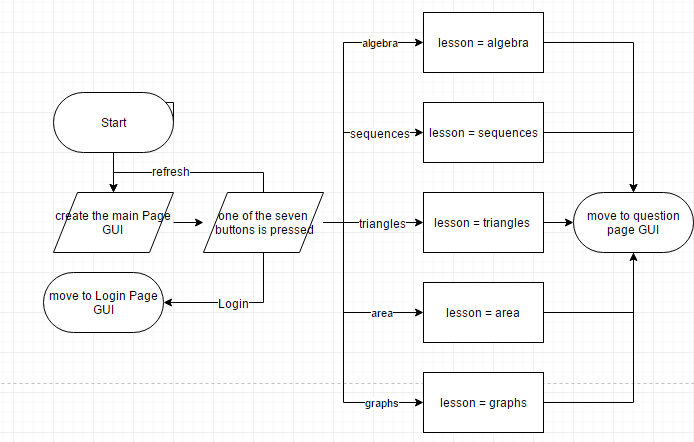
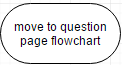
Teacher page flowchart

The flowchart showing the actions taken on the teacher page and what happens when each button is pressed.



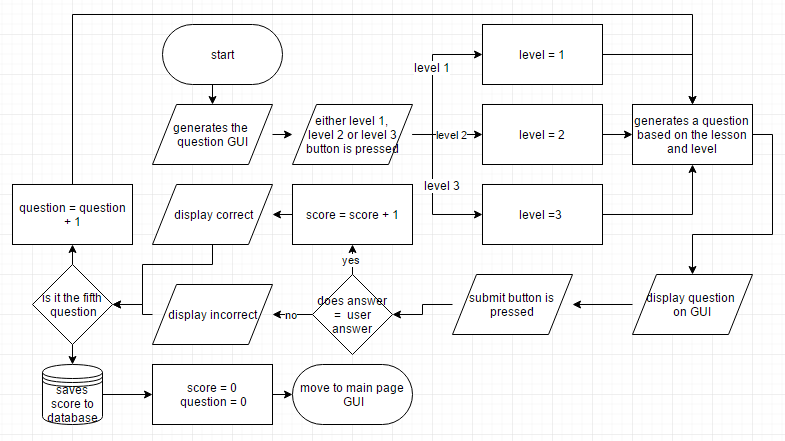
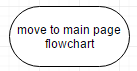
Main Page flowchart

The flowchart showing the actions taken on the main page and what happens when each button is pressed.



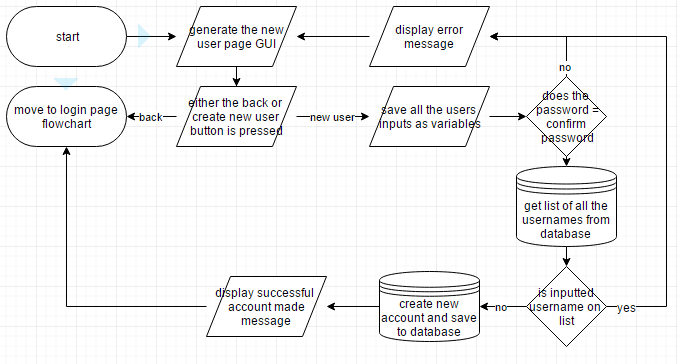
Question Page flowchart

The flowchart showing the actions taken on the question page and what happens when each button is pressed.



New user flowchart

The flowchart showing the actions taken on the new user page and what happens when each button is pressed.

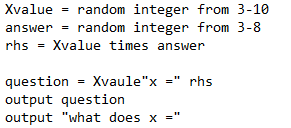


Description plus diagrams such as class diagram, flowchart, DFD, hierarchy chart (use the most appropriate)

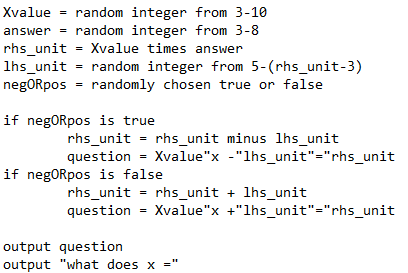
## Algorithms

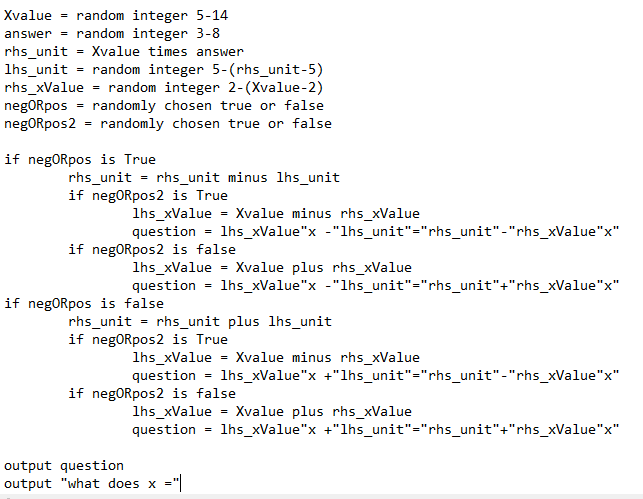
Pseudocode or similar for key algorithms essential to the success of the project

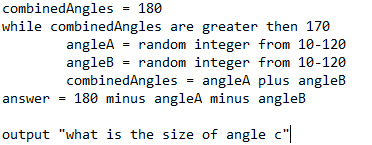
* Pseudocode for generating the algebra level 1 questions:



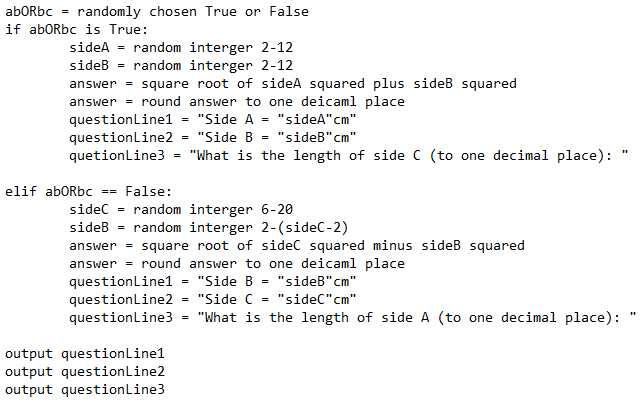
* Pseudocode for generating the algebra level 2 questions:



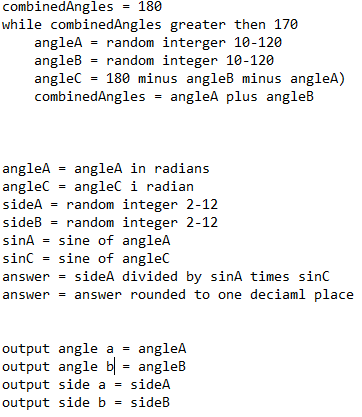
* Pseudocode for generating the algebra level 3 questions: 
* Pseudocode for generating the triangle level 1 questions:



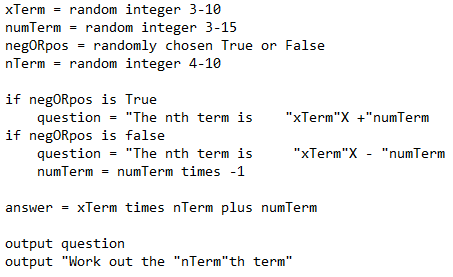
* Pseudocode for generating the triangle level 2 questions:



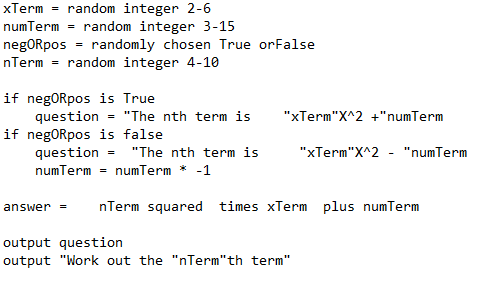
* Pseudocode for generating the triangle level 3 questions:



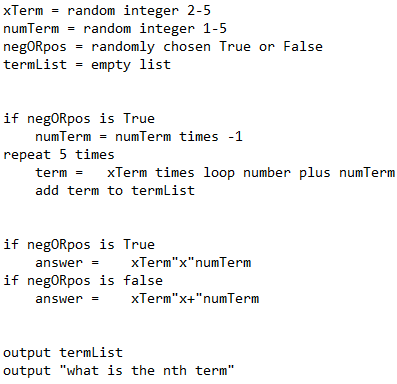
* Pseudocode for generating the sequence level 1 questions:



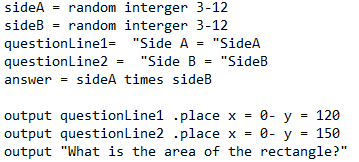
* Pseudocode for generating the sequence level 2 questions:



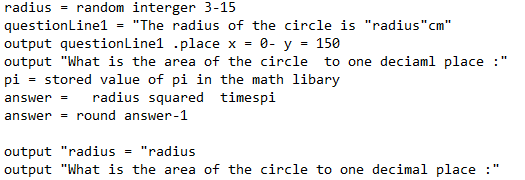
* Pseudocode for generating the sequence level 3 questions:



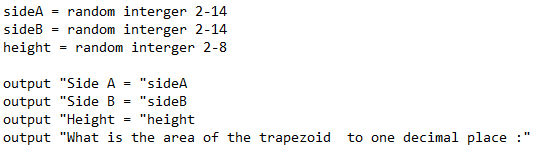
* Pseudocode for generating the area level 1 questions:



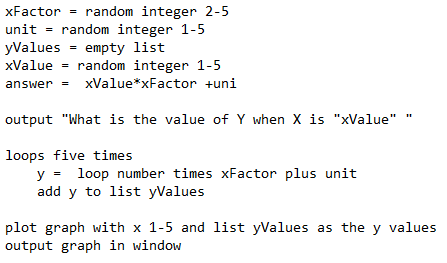
* Pseudocode for generating the area level 2 questions:



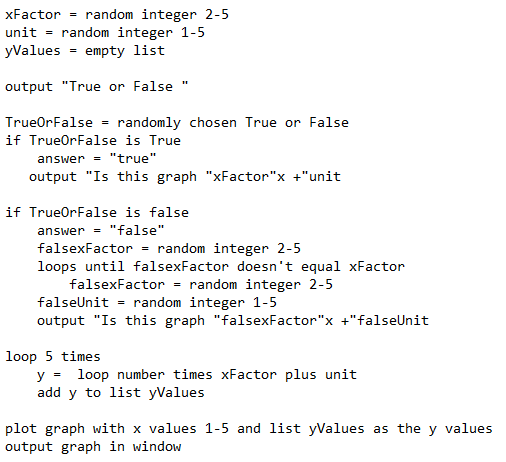
* Pseudocode for generating the area level 3 questions:



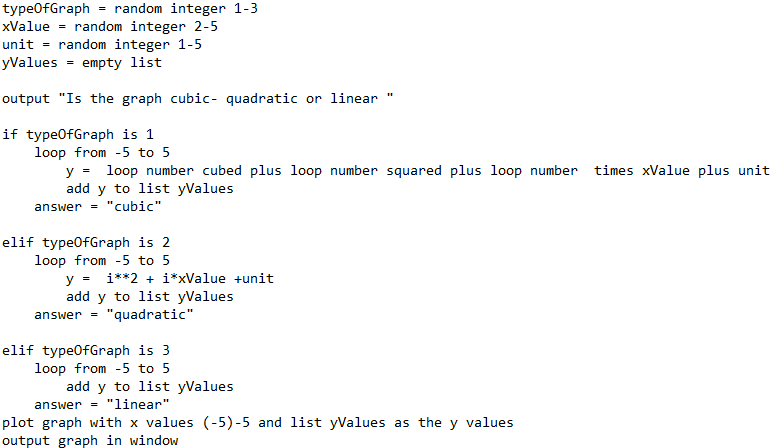
* Pseudocode for generating the graph level 1 questions:



* Pseudocode for generating the graph level 2 questions:



* Pseudocode for generating the graph level 3 questions:



## Data structures

**Description of Program Classes:**

|  |  |
| --- | --- |
| Class | Description |
| start\_page | A class that creates the start page user interface |
| login\_page | A class that creates the login page user interface |
| new\_user\_page | A class that creates the new user page user interface |
| teacher\_page | A class that creates the teacher page user interface |
| main\_page | A class that creates the main page user interface |
| algebra | A class that creates the algebra question page user interface and generates the algebra questions |
| triangles | A class that creates the triangles question page user interface and generates the triangles questions |
| sequences | A class that creates the sequences question page user interface and generates the sequences questions |
| area | A class that creates the area question page user interface and generates the area questions |
| graph | A class that creates the graph question page user interface and generates the graph questions |
| check\_answer | A class that checks the user’s answer against the actual answer an stores the scores in the csv file |

**Description of class variables:**

|  |  |  |
| --- | --- | --- |
| start\_page – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the start page is displayed on. It is created at the start of the class and destroyed at the end. |
| password\_username | String | Set to “0” and passed onto the login\_page class to stop any error messages appearing. |
| User | String | Set to “student” or “teacher” depending on the button that the user clicked. This is passed onto the login\_page class used to access the correct corresponding csv file. |

|  |  |  |
| --- | --- | --- |
| login\_page – variables | Type | Description |
| password\_username | String | Can either hold the value “0”,”1” or “2”. If password username == “0” no error message is displayed, if If password username == “1” the error message “Wrong Password” is displayed and if password username == “2” the error message “Wrong Username and Password” is displayed. The value of password\_username changes depending on the users inputs when they try to log in. |
| Root | Widget | Used to hold the tkinter window that the login page is displayed on. It is created at the start of the class and destroyed at the end. |
| User | String | Can either hold the value “teacher” or “student”. This variable is used to select the correct file name for the login details to be checked against. |
| user\_row (global variable) | Integer | Stores the value of what row the user’s details are stored in the csv file. This is later passed onto the main\_page class where it is used to display the user’s scores. |
| Username | String | Stores the user inputted username. This variable is used to compare the username against the saved usernames in the selected csv file. |
| Password | String | Stores the user inputted password. This variable is used to compare the username against the saved passwords in the selected csv file. |
| hashed\_password | String | Stores the hashed value of the user’s inputted password, the hashed\_password variable is always a 32bit hexadecimal value. |
| File | String | Stores either the value “teacher\_accounts.csv” or “student\_accounts.csv” depending on the value of user. The variable is used to access the correct file for the user’s details. |
| Correct | String | Stores either the value “0”,”1” or “2” depending on whether the user’s inputted details have been found in the csv file. The variable stops the iteration of going through all the csv files usernames when the matching username-password combination is found. |
| Exist | Boolean | Stores either the value True or False depending on whether the user’s inputted username-password combination exist in the csv file , this is used to destroy the tkinter window and call the login\_page class when the value is false |
| first\_load | Boolean | Stores either the value True or False depending on whether it’s the teachers first time accessing the teacher page since logging in. The value of this variable affects the scores loaded on the teacher page. The variable is set to true in this class before being passed onto the teacher\_page class. |
| Database | Dict | Used to store the data in the csv file. This is then used to compare against the user’s login details. |
| Username\_File | String | Used to store the username stored in the current row being compared to the user’s inputted username. The variable changes with every loop as the variable row changes |
| Password\_File | String | Used to store the password stored in the current row being compared to the user’s inputted password. The variable changes with every loop as the variable row changes |
| Row | integer | Used to store the row in the csv file currently being compared against the user’s inputted username-password combination. |

|  |  |  |
| --- | --- | --- |
| new\_user\_page – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the new user page is displayed on. It is created at the start of the class and destroyed at the end. |
| password\_username | String | Can either hold the value “0”,”1” or “2”. If password username == “0” no error message is displayed, if If password username == “1” the error message “Wrong Password” is displayed and if password username == “2” the error message “Wrong Username and Password” is displayed. The variable is set to “0” ready to be passed into the login\_page class. |
| User | String | Can either hold the value “teacher” or “student”. This variable is used to select the correct file name for the new user details to be stored in. |
| all\_usernames | List | The list is empty at the start of the class but when the inputted username is being for uniqueness all the usernames in the csv file are stored in the list. |
| Unique | String | Can either be the value “0”, “1”, or “2”. The value of the variable unique is changed depending on whether the inputted username is unique and hasn’t already been taken and stored in the csv file. |
| first\_name | String | Stores the user’s input from the first name entry box. This variable has a 30 character limit |
| Surname | String | Stores the user’s input from the last name entry box. This variable has a 30 character limit |
| Username | String | Stores the user’s input from the username entry box. This variable has a 30 character limit |
| Password | String | Stores the user’s input from the password entry box. This variable has a 30 character limit |
| confirm\_password | String | Stores the user’s input from the confirm password entry box. This variable has a 30 character limit |
| File | String | Stores either the value “teacher\_accounts.csv” or “student\_accounts.csv” depending on the value of user. The variable is used to access the correct file to store user’s details. |
| Database | Dict | Used to store the data in the csv file. This is then used to compare against the user’s new user details. |
| Row | Interger | Used to store the row in the csv file currently being compared against the user’s inputted username to check whether the username is unique |
| Username\_File | String | Used to store the username stored in the current row being compared to the user’s inputted username. The variable changes with every loop as the variable row changes |
| num\_users | Integer | Stores the value of the length of the all\_usernames list, it is then used to loop the section of code that checks the user’s inputted username against the take usernames. |
| current\_username | String | Stores the value of the stored username form the csv file that is currently being compared against the user’s inputted username. The variable changes with each iteration as the variable taken\_username is increased. |
| taken\_username | String | Stores the location in the list all\_usernames of the username currently being compared against the user’s inputted username |
| Fieldnames | List | Used to store all the column headings the csv file that is currently being updated. |
| hashed\_password | String | Stores the hashed value of the user’s inputted password, the hashed\_password variable is always a 32bit hexadecimal value. |

|  |  |  |
| --- | --- | --- |
| teacher\_page – variables | Type | Description |
| root | Widget | Used to hold the tkinter window that the teacher page is displayed on. It is created at the start of the class and destroyed at the end. |
| first\_load | Boolean | Stores either the value True or False depending on whether it’s the teachers first time accessing the teacher page since logging in. The value of this variable affects the scores loaded on the teacher page. The variable is changed to false after the user presses the get scores button. |
| user | String | Can either hold the value “teacher” or “student”. This variable is passed through the teacher\_page class and passed back into the login\_page class when the log out button is pressed. |
| blank | Integer | Used to loop fifth teen times through the code that sets the student displayed scores to “-“when the teacher page is first loaded. |
| user\_score (global variable) | List | Used to store the user’s scores that the teacher has inputted to be displayed. The list is rest to an empty list every time the get scores button is pressed ready to take a new set of fifth teen values. |
| user\_row(global variable) | Integer | Stores the value of what row the user’s details are stored in the csv file. This is then used to save the scores to the user\_score list before being displayed. |
| first\_name | String | Stores the user’s input from the first name entry box |
| last\_name | String | Stores the user’s input from the last name entry box |
| Exist | Boolean | A Boolean value that either stores the value true or false. The value of the variable changes if a student is found with a matching name to the one inputted by the user. |
| Database | Dict | Used to store the data in the csv file. This is then used to compare against the user’s new user details. |
| first\_name\_file | String | Used to store the first name stored in the current row being compared to the user’s inputted first name. The variable changes with every loop as the variable row changes |
| last\_name\_file | String | Used to store the last name stored in the current row being compared to the user’s inputted last name. The variable changes with every loop as the variable row changes |
| Row | Integer | Used to store the row in the csv file currently being compared against the user’s inputted first name-last name combination to check whether the student exists |
| Score | Integer | Used to loop through all the user’s details stored in the all\_details list and only adds the scores to the user\_score list |
| userScores | Dict | Used to store the data in the csv file. This is then used to compare against the user’s new user details. |
| Item | Integer | Used to loop through all the values in the user’s row. Each value is added to the list all\_details |
| password\_username | String | Can either hold the value “0”,”1” or “2”. If password username == “0” no error message is displayed, if If password username == “1” the error message “Wrong Password” is displayed and if password username == “2” the error message “Wrong Username and Password” is displayed. The variable is set to “0” ready to be passed into the login\_page class. |
| all\_details | List | Used to store all the students details stored in the csv file. |

|  |  |  |
| --- | --- | --- |
| main\_page – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the main page is displayed on. |
| User | String | Can either hold the value “teacher” or “student”. This variable is passed through the main\_page class and passed back into the login\_page class when the log out button is pressed. |
| Score | Integer | Used to store the score that the user is currently on. The score is set to 0 each time the main\_page class is called. The variable is passed onto all five lesson classes. |
| Question | Integer | Used to store the question that the user is currently on. The question is set to 0 each time the main\_page class is called. The variable is passed onto all five lesson classes. |
| Level | String | Used to store the level that the user is currently on. The level is set to 0 each time the main\_page class is called. The variable is passed onto all five lesson classes. |
| Database | Dict | Used to store the data in the csv file. This is then used to compare against the user’s new user details. |
| user\_details | List | Used to store all the students details stored in the csv file. |
| Item | Integer | Used to loop through all the items in the userScores |
| display\_score | Integer | Used to loop through all the user’s details stored in the all\_details list and only adds the scores to the user\_score list |
| user\_score (global variable) | List | Used to store the user’s scores that the teacher has inputted to be displayed. The list is rest to an empty list every time the get scores button is pressed ready to take a new set of fifth teen values. |
| user\_row(global variable) | Integer | Stores the value of what row the user’s details are stored in the csv file. This is then used to save the scores to the user\_score list before being displayed. |
| userScores | Dict | Used to store the data in the csv file. This is then used to compare against the user’s new user details. |
| frame1 | Widget | Used to hold the tkinter frame that main page is displayed on. The frame is raised so it’s visible on top of the other frames on the same root. |

|  |  |  |
| --- | --- | --- |
| algebra – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the main page and all the lesson question pages are displayed on. |
| Score | Integer | Used to store the score that the user is currently on. The score is passed onto the check\_answer class. |
| Question | Integer | Used to store the question that the user is currently on. The question passed onto the check\_answer class. The variable is also displayed at the top of the algebra question screen |
| Lesson | String | Used to store the lesson that the user is currently on. The question passed onto the check\_answer class. |
| Level | String | Used to store the level that the user is currently on. The level is set to either “1”, “2” or “3” depending on the which level button the user clicks. The variable passed onto the check\_answer class |
| frame2 | Widget | Used to hold the tkinter frame that algebra question page is displayed on. The frame is raised so it’s visible on top of the other frames on the same root. |
| Xvalue | Integer | A randomly generated value used to generate the question |
| Answer | Integer | A randomly generated value used to generate the question. This is passed on to the class check\_answer class. |
| rhs\_unit | Integer | A randomly generated value used to generate the question |
| lhs\_unit | Integer | A randomly generated value used to generate the question |
| rhs\_xValue | Integer | A randomly generated value used to generate the question |
| lhs\_Xvalue | Integer | A randomly generated value used to generate the question |
| negORpos | Boolean | A randomly generated value used to decide the form that the question is going to be written in. |
| negORpos2 | Boolean | A randomly generated value used to decide the form that the question is going to be written in. |
| user\_answer | String | The user’s inputted answer to the question. The value is change to the value in the answer entry box every time the submit button is pressed. This is passed on to the class check\_answer class. |
| Frame | Widget | Used to hold the tkinter frame that algebra question page is displayed on. This is passed on to the class check\_answer class. |

|  |  |  |
| --- | --- | --- |
| triangles – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the main page and all the lesson question pages are displayed on. |
| Score | Integer | Used to store the score that the user is currently on. The score is passed onto the check\_answer class. |
| Question | Integer | Used to store the question that the user is currently on. The question passed onto the check\_answer class. The variable is also displayed at the top of the triangle question screen |
| Lesson | String | Used to store the lesson that the user is currently on. The question passed onto the check\_answer class. |
| Level | String | Used to store the level that the user is currently on. The level is set to either “1”, “2” or “3” depending on the which level button the user clicks. The variable passed onto the check\_answer class |
| frame3 | Widget | Used to hold the tkinter frame that algebra triangle page is displayed on. The frame is raised so it’s visible on top of the other frames on the same root. |
| combinedAngles | Integer | A randomly generated value used to generate the question |
| angleA | Float/integer | A randomly generated value used to generate the question it is later converted from a degrees value into a radian value changing it from a integer value to a float |
| angleB | Float/integer | A randomly generated value used to generate the question it is later converted from a degrees value into a radian value changing it from a integer value to a float |
| Answer | Float/integer | A randomly generated value used to generate the question. This is passed on to the class check\_answer class. |
| Photo | String | The file name of the picture that is displayed on an label on the question page |
| sideA | Integer | A randomly generated value used to generate the question |
| sideB | Integer | A randomly generated value used to generate the question |
| sideC | Integer | A randomly generated value used to generate the question |
| questionLine1 | String | The first line of the question in saved as this variable, the variable is later displayed as text in a label on the tkinter frame |
| questionLine2 | String | The second line of the question in saved as this variable, the variable is later displayed as text in a label on the tkinter frame |
| strangle | String | The string value of the variable angleA |
| strangle | String | The string value of the variable angleB |
| sinA | Float | The sine value of the radian value of angleA |
| sinC | Float | The sine value of the radian value of angleB |
| angleC | Integer | A randomly generated value used to generate the question |
| abORbc | Boolean | A randomly generated value used to decide the form that the question is going to be written in. |
| user\_answer | String | The user’s inputted answer to the question. The value is change to the value in the answer entry box every time the submit button is pressed. This is passed on to the class check\_answer class. |
| Frame | Widget | Used to hold the tkinter frame that triangle question page is displayed on. This is passed on to the class check\_answer class. |

|  |  |  |
| --- | --- | --- |
| sequences – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the main page and all the lesson question pages are displayed on. |
| Score | Integer | Used to store the score that the user is currently on. The score is passed onto the check\_answer class. |
| Question | Integer | Used to store the question that the user is currently on. The question passed onto the check\_answer class. The variable is also displayed at the top of the sequences question screen |
| Lesson | String | Used to store the lesson that the user is currently on. The question passed onto the check\_answer class. |
| Level | String | Used to store the level that the user is currently on. The level is set to either “1”, “2” or “3” depending on the which level button the user clicks. The variable passed onto the check\_answer class. |
| frame4 | Widget | Used to hold the tkinter frame that sequences question page is displayed on. The frame is raised so it’s visible on top of the other frames on the same root. |
| xTerm | Integer | A randomly generated value used to generate the question |
| numTerm | Integer | A randomly generated value used to generate the question |
| negORpos | Boolean | A randomly generated value used to decide the form that the question is going to be written in. |
| nTerm | Integer | A randomly generated value used to generate the question |
| questionLine1 | String | The first line of the question in saved as this variable, the variable is later displayed as text in a label on the tkinter frame |
| Answer | Integer/string | A variable generated from the other variables in the class. The answer is a integer value if level == “1” or “2” but the answer is a string is level ==”3”. This is passed on to the class check\_answer class. |
| termList | List | A list storing a sequence generated using the other variables. The list is displayed in a label on the level 3 questions. |
| user\_answer | String | The user’s inputted answer to the question. The value is change to the value in the answer entry box every time the submit button is pressed. This is passed on to the class check\_answer class. |
| Frame | Widget | Used to hold the tkinter frame that sequences question page is displayed on. This is passed on to the class check\_answer class. |

|  |  |  |
| --- | --- | --- |
| area – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the main page and all the lesson question pages are displayed on. |
| Score | Integer | Used to store the score that the user is currently on. The score is passed onto the check\_answer class. |
| Question | Integer | Used to store the question that the user is currently on. The question passed onto the check\_answer class. The variable is also displayed at the top of the area question screen |
| Lesson | String | Used to store the lesson that the user is currently on. The question passed onto the check\_answer class. |
| Level | String | Used to store the level that the user is currently on. The level is set to either “1”, “2” or “3” depending on the which level button the user clicks. The variable passed onto the check\_answer class. |
| frame5 | Widget | Used to hold the tkinter frame that area question page is displayed on. The frame is raised so it’s visible on top of the other frames on the same root. |
| sideA | Integer | A randomly generated value used to generate the question |
| sideB | Integer | A randomly generated value used to generate the question |
| strSideA | String | The string value of the variable sideA |
| strSideB | String | The string value of the variable sideB |
| questionLine1 | String | The first line of the question in saved as this variable, the variable is later displayed as text in a label on the tkinter frame |
| questionLine2 | String | The second line of the question in saved as this variable, the variable is later displayed as text in a label on the tkinter frame |
| Answer | Float/integer | A variable generated from the other variables in the class. The answer is a integer value if level == “1” but the answer is a float if level ==”3” or “2”. This is passed on to the class check\_answer class. |
| Photo | String | The file name of the picture that is displayed on an label on the question page |
| Radius | Integer | A randomly generated value used to generate the question |
| Pi | Float | The mathematical concept pi from the math libary |
| Height | Integer | A randomly generated value used to generate the question |
| user\_answer | String | The user’s inputted answer to the question. The value is change to the value in the answer entry box every time the submit button is pressed. This is passed on to the class check\_answer class. |
| Frame | Widget | Used to hold the tkinter frame that area question page is displayed on. This is passed on to the class check\_answer class. |

|  |  |  |
| --- | --- | --- |
| graph – variables | Type | Description |
| Root | Widget | Used to hold the tkinter window that the main page and all the lesson question pages are displayed on. |
| Score | Integer | Used to store the score that the user is currently on. The score is passed onto the check\_answer class. |
| Question | Integer | Used to store the question that the user is currently on. The question passed onto the check\_answer class. The variable is also displayed at the top of the graph question screen |
| Lesson | String | Used to store the lesson that the user is currently on. The question passed onto the check\_answer class. |
| Level | String | Used to store the level that the user is currently on. The level is set to either “1”, “2” or “3” depending on the which level button the user clicks. The variable passed onto the check\_answer class. |
| frame6 | Widget | Used to hold the tkinter frame that area graph page is displayed on. The frame is raised so it’s visible on top of the other frames on the same root. |
| xFactor | Integer | A randomly generated value used to generate the question |
| Unit | Integer | A randomly generated value used to generate the question |
| yValues | List | An list that stores all the yValues to later be turned into a graph |
| xValue | Integer | A randomly generated value used to generate the question |
| Answer | Integer/string | A variable generated from the other variables in the class. The answer is a integer value if level == “1” but the answer is a string if level ==”3” or “2”. This is passed on to the class check\_answer class. |
| X | Integer | The variable used when looping through to generate the y values |
| Y | Integer/float | The variable used to store the generated y value of the graph before it is stored in the yValues list, the value of y changes with each iteration as x increase by 1 |
| TrueOrFalse | Boolean | A randomly generated value used to decide the form that the question is going to be written in. |
| falsexFactor | Integer | A randomly generated value used to generate the question |
| falseUnit | Integer | A randomly generated value used to generate the question |
| typeOfGraph | integer | A randomly generated value used to generate the question |
| user\_answer | String | The user’s inputted answer to the question. The value is change to the value in the answer entry box every time the submit button is pressed . This is passed on to the class check\_answer class. |
| Frame | Widget | Used to hold the tkinter frame that area graph page is displayed on. This is passed on to the class check\_answer class. |

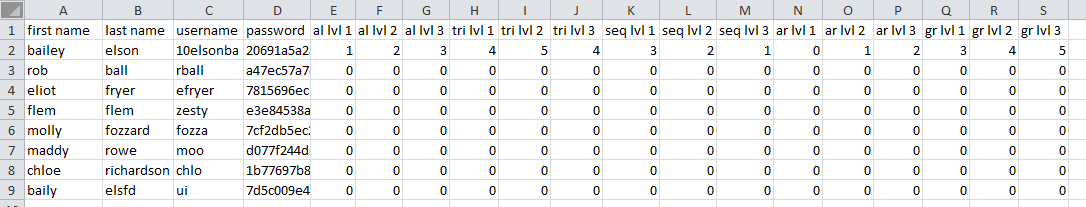
|  |  |  |
| --- | --- | --- |
| check\_answer – variables | Type | Description |
| Score | Integer | Used to store the score that the user is currently on. The score is passed onto the check\_answer class. |
| Question | Integer | Used to store the question that the user is currently on. The question passed onto the check\_answer class. The variable is also displayed at the top of the graph question screen |
| Lesson | String | Used to store the lesson that the user is currently on. The question passed onto the check\_answer class. |
| Level | String | Used to store the level that the user is currently on. The level is set to either “1”, “2” or “3” depending on the which level button the user clicks. The variable passed onto the check\_answer class. |
| Frame | Widget | Used to hold the tkinter frame that lesson question page is displayed on. The frame is destroyed in this class. |
| user\_answer | String | The user’s inputted answer to the question. The value is change to the value in the answer entry box every time the submit button is pressed |
| Answer | String | Used to store the correct answer to the question being checked. The user\_answer is compared against this value to see if the user is correct or incorrect. |
| Database | Dict | Used to store the data in the csv file. This is then used to update the students’ scores when they finish a level on any lesson |
| Lines | Integer | Used to store the line value when updating the user’s scores. |
| Location | Integer | Used to store the location in the csv file that needs to be updated, the variable is generated depending on the values of the lesson and level variables |
| user\_score (global variable) | List | Used to store the user’s scores that the teacher has inputted to be displayed. The list is rest to an empty list every time the get scores button is pressed ready to take a new set of fifth teen values. |
| user\_row(global variable) | Integer | Stores the value of what row the user’s details are stored in the csv file. This is then used to save the scores to the user\_score list before being displayed. |

## File structure and organisation

## Database design

My program will use two csv files. One is for storing the student account details and the other is for storing the teachers account details

Student\_accounts.csv (student account details)



Column A will be used to store the user’s first name. The first name will be stored as a string value.

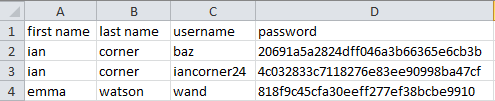
Column B will be used to store the user’s last name. The last name will be stored as a string value.

Column C will be used to store the user’s username. The username will be stored as a string value.

Column D will be used to store the user’s hashed password. The password will be stored as a string value.

Column E-S will be used to store the users scores in the different lesson and levels. They will be stored as integer values between 0-5

Teacher\_accounts.csv (teacher account details)



Column A will be used to store the user’s first name. The first name will be stored as a string value.

Column B will be used to store the user’s last name. The last name will be stored as a string value.

Column C will be used to store the user’s username. The username will be stored as a string value.

Column D will be used to store the user’s hashed password. The password will be stored as a string value.

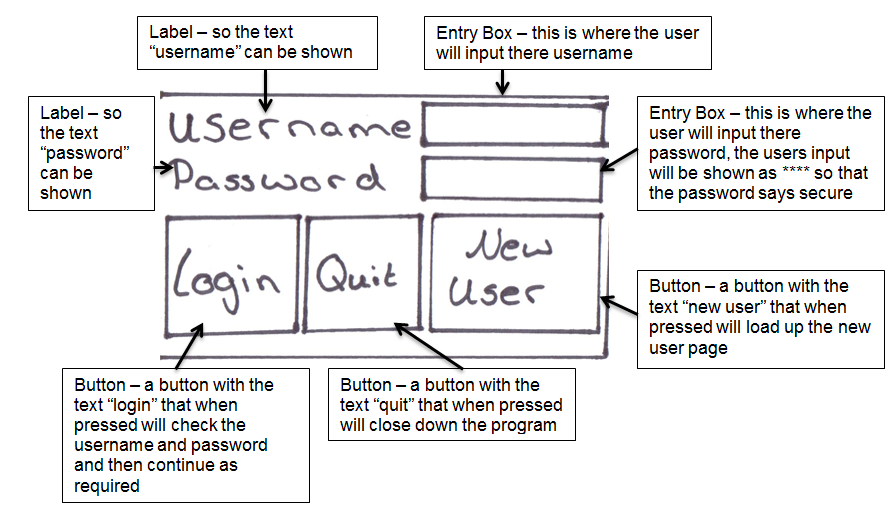
## SQL queries

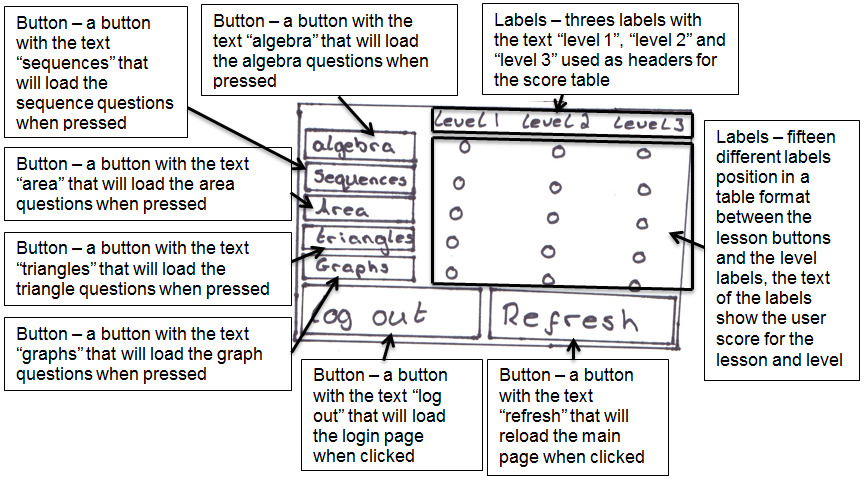
If appropriate

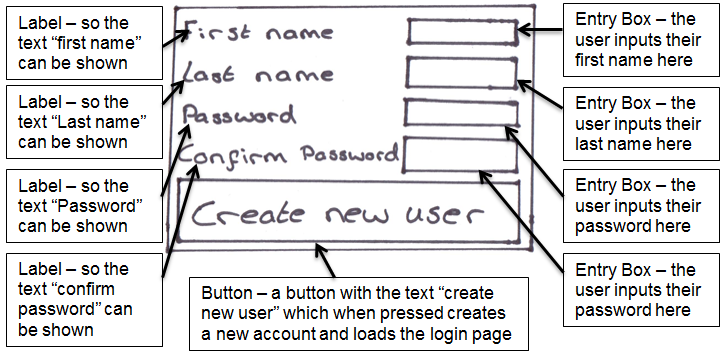
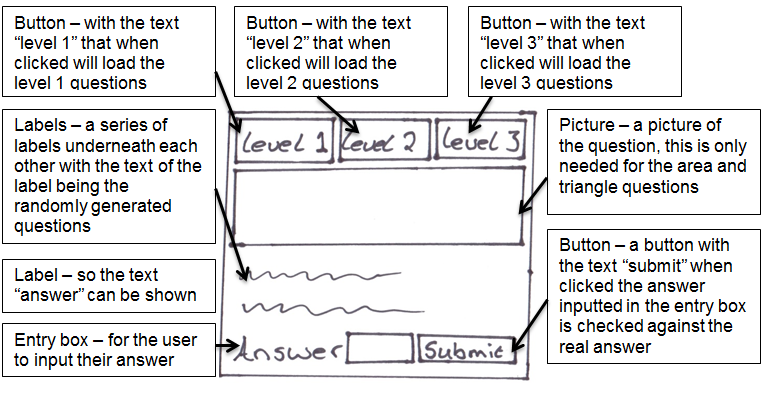
## 

## User interface design (HCI)

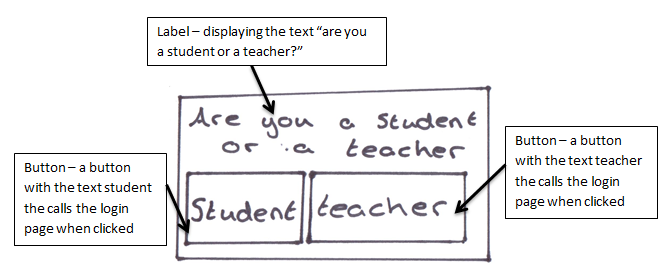
Login page:



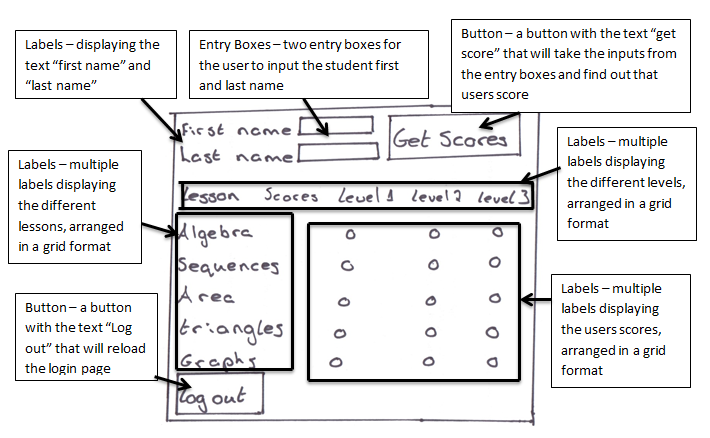
Main Page:

New User Page:Question Page:

Start Page:



Teacher Page:



# 

# Technical Solution *(42 marks)*

## Code listing

*#importing all the different libaries need for the program***import** tkinter **as** tk  
**from** tkinter **import** messagebox  
**import** random  
**import** math  
**import** csv  
**import** matplotlib.pyplot **as** plt  
**import** hashlib  
  
*#the login page class, creates the login page and the functions related to all the buttons on the login page***class** login\_page:  
  
 *#creates the initial login page, creates all the buttons, labels and entry boxes in tkinter for the login page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, password\_username and user as parameters* **def** \_\_init\_\_(self, password\_username, user):  
  
  
 *#generates the GUI window* root = tk.Tk()  
 root.geometry(**"190x95"**)  
 root.title(**"Login Page"**)  
  
 *#creating and placing all the labels on the login page* self.username\_label = tk.Label(root, text=**"Username"**).place(x = 0, y = 0)  
 self.username\_label = tk.Label(root, text=**"Password"**).place(x = 0, y = 20)  
  
 *#creating and placing all the entry boxes on the login page  
 #the password entry box stars the users input so the password stays secure* self.username\_entry = tk.Entry(root)  
 self.username\_entry.place(x = 70, y = 0)  
 self.password\_entry = tk.Entry(root, show = **"\*"**)  
 self.password\_entry.place(x = 70, y = 20)  
  
 *#creating and placing all the buttons on the login page  
 #also creates the commands for when the buttons are pressed* self.login\_button = tk.Button(root, text=**"Login"**, command = **lambda**:self.Login(root, password\_username, user))  
 self.login\_button.place(x = 0, y = 45)  
 self.quit\_button = tk.Button(root, text=**"Quit"**, command = **lambda**:self.Quit(root))  
 self.quit\_button.place(x = 45, y = 45)  
 self.new\_user\_button = tk.Button(root, text = **"Create New User"**, command = **lambda**:self.new\_user(root, user))  
 self.new\_user\_button.place(x = 90, y = 45)  
  
 *#creating the error message labels based on if one is needed and if one is needed which one is needed* **if** password\_username == **"1"**:  
 tk.messagebox.showerror(**"Error"**, **"Wrong Password"**)  
 **elif** password\_username == **"2"**:  
 tk.messagebox.showerror(**"Error"**, **"Wrong Username and Password"**)  
  
 *#the function login that is called when the login button is pressed  
 #the function uses the variables self, root, password\_username and user as parameters  
 #the purpose of the function is to check the login details and then call the needed function based on the login details  
 #objective 6 and 7 are met in this function* **def** Login(self, root, password\_username, user):  
  
 *#defines the variable user\_row as a global variable and resets the variable* **global** user\_row  
 user\_row = 0  
  
 *#takes all the users inputs from the entry boxes and saves them as sepereate varibles* Username = self.username\_entry.get()  
 Password = self.password\_entry.get()  
  
 *#hashes the pssword as saves it as a 32 bit hexidecimal variable* hashed\_password = hashlib.md5(Password.encode())  
 Password = (hashed\_password.hexdigest())  
  
 *#allocates the file name needed for the current user* **if** user == **"student"**:  
 file = **"student\_accounts.csv"  
 elif** user == **"teacher"**:  
 file = **"teacher\_accounts.csv"** *#checks if the password-username combination inputted by the user is a valid combinaton agaist the accouts stored in the csv files  
 #if the user is teacher and the input is valid then the teacher\_page function is called  
 #if the user is student and the input is valid then the main\_page function is called  
 #if the input is invalid then the login\_page class is called and an error message will appear on the login page* exist = False  
 **with** open(file) **as** csvfile:  
 database = csv.DictReader(csvfile)  
 Username = str.lower(Username)  
 correct = **"0"   
 while** correct != **"1"**:  
 **for** row **in** database:  
 user\_row = user\_row+1  
 Username\_File = row[**"username"**]  
 Password\_File = row[**"password"**]  
 **if** Username\_File == Username **and** Password\_File == Password:  
 exist = True  
 correct = **"1"** root.destroy()  
 **if** user == **"student"**:  
 main\_page(user)  
 **elif** user == **"teacher"**:  
 first\_load = True  
 teacher\_page(first\_load, user)  
 **elif** Username\_File == Username **and** Password\_File != Password:   
 password\_username = **"1"** correct = **"2"   
 elif** Username\_File != Username **and** Password\_File != Password:   
 **if** correct != **"2"**:  
 correct = **"3"** password\_username = **"2"  
 break  
 if** exist == False:  
 root.destroy()  
 login\_page(password\_username, user)  
   
 *#the function quit that is called when the quit button is pressed  
 #the function uses the variables self and root as parameters  
 #the purpose of the function is to close down the window* **def** Quit(self, root):  
 *#closes the window, ending the program* root.destroy()  
  
 *#the new\_user function is called when the create new user button is pressed  
 #the function used the variables self, root and user as parameters  
 #the purpose of the function is to create the new window for the new user page* **def** new\_user(self, root, user):  
 *#closes the current window and creates a new blank widow before calling the new\_user\_page funtion  
 #also defines the variables needed for the new\_user\_page function* root.destroy()  
 password\_username = **"0"** new\_user\_page(password\_username, user)  
  
*#the new\_user\_page class, creates the new user page and the functions related to all the buttons on the login page***class** new\_user\_page:  
   
 *#creates the new user page, creates all the buttons, labels and entry boxes in tkinter for the new user page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, password\_username and user as parameters* **def** \_\_init\_\_(self, password\_username, user):  
   
 root = tk.Tk()  
 root.geometry(**"300x160"**)  
 root.title(**"new user page"**)  
 password\_username = **"0"** *#creates and places all the labels on the new user page* self.firt\_name\_label = tk.Label(root, text = **"First Name:"**).place(x = 0, y = 0)  
 self.last\_name\_label = tk.Label(root, text = **"Surname:"**).place(x = 0, y = 20)  
 self.username\_label = tk.Label(root, text = **"Username:"**).place(x = 0, y = 40)  
 self.password\_label = tk.Label(root, text = **"Password:"**).place(x = 0, y = 60)  
 self.confirm\_password = tk.Label(root, text = **"Confirm Password:"**).place(x = 0, y = 80)  
  
 *#creates and places all the entry boxes on the new user page  
 #it also stars the input on the password and confirm password entry boxes so that the passwords are kept secure* self.first\_name\_entry = tk.Entry(root)  
 self.first\_name\_entry.place(x = 110, y = 0)  
 self.last\_name\_entry = tk.Entry(root)  
 self.last\_name\_entry.place(x = 110, y = 20)  
 self.username\_entry = tk.Entry(root)  
 self.username\_entry.place(x = 110, y = 40)  
 self.password\_entry = tk.Entry(root, show = **"\*"**)  
 self.password\_entry.place(x = 110, y = 60)  
 self.confirm\_password\_entry = tk.Entry(root, show = **"\*"**)  
 self.confirm\_password\_entry.place(x = 110, y = 80)  
  
 *#creates and places all the buttons on the new user page  
 #also creates the commands for when the buttons are pressed* self.new\_account\_button = tk.Button(root, text = **"Create New Account"**, command = **lambda**:self.create\_new\_user(root, password\_username, user))  
 self.new\_account\_button.place(x = 0, y = 110)  
 self.return\_button = tk.Button(root, text = **"Return"**, command = **lambda**:self.back(root, password\_username, user)).place(x = 150, y = 110)  
  
 *#the function create\_new\_user is called when the create new user button is pressed  
 #the function takes the variables self, root, password\_username and user as parameters  
 #the purpose of the function is to check if the username is unique and if it is then edit the csv file with the new users account details  
 #objective 1,2,3,4 and 5 are met inside this function* **def** create\_new\_user(self,root, password\_username, user):  
  
 *#defines the new variables needed for the fuction* all\_usernames = []  
 unique = **"0"** *#saves the users inputted data as variables from the entry boxes  
 #it also makes the variables first\_name and surnae lower cas* first\_name = self.first\_name\_entry.get()  
 first\_name = first\_name.lower()  
 surname = self.last\_name\_entry.get()  
 surname = surname.lower()  
 username = self.username\_entry.get()  
 password = self.password\_entry.get()  
 confirm\_password = self.confirm\_password\_entry.get()  
  
 *#allocates the file name needed for the current user* **if** user == **"student"**:  
 file = **"student\_accounts.csv"  
 elif** user == **"teacher"**:  
 file = **"teacher\_accounts.csv"** *#checks if the password and confirm password values are the same  
 #if they aren't then a error message is displayed in a message box* **if** password != confirm\_password:  
 tk.messagebox.showerror(**"Error"**, **"Passwords do not match"**)  
 *#if they are the same then then the program checks whether all the fields have been filled in* **elif** password == **"" or** username == **"" or** surname == **"" or** first\_name == **""**:  
 tk.messagebox.showerror(**"Error"**, **"Some fields have been left blank"**)  
 **elif** len(password) > 30 **or** len(username) > 30 **or** len(surname) > 30 **or** len(first\_name) > 30:  
 tk.messagebox.showerror(**"Error"**, **"All fields have a 30 character limit"**)  
 *#if they are the same then then the program checks whether the username is unique and not already in use* **else**:  
 *#checks if the user name is unique against the usernames for the accounts already made and saved in the csv files* **with** open(file) **as** csvfile:  
 database = csv.DictReader(csvfile)  
 **for** row **in** database:  
 Username\_File = row[**'username'**]  
 all\_usernames.append(Username\_File)  
 num\_users = len(all\_usernames)  
 **for** taken\_username **in** range (0,num\_users):  
 current\_username = all\_usernames[taken\_username]  
 **if** username == current\_username:  
 unique = **"1"  
 else**:   
 **if** unique != **"1"**:  
 unique = **"2"** *#if the password is unique the new account inforation is then saved into the correct csv file for the user type* **if** unique == **"2"**:  
 *#opens up the csv file in append mode* **with** open(file, **"a"**,newline=**""**) **as** database:  
 *#a message is created in a message box saying the account setup has been successful* tk.messagebox.showinfo(**"Success"**,**"Your account has been successfully created"**)  
 *#creates a new row in the student\_accounts.csv file for the new student account information  
 #this new row includes the students details like their first name, surname, username, hashed password and intials scores of 0 for all leves* **if** user == **"student"**:  
 fieldnames = [**"first name"**,**"last name"**,**"username"**,**"password"**,**"al 1"**,**"al 2"**,**"al 3"**,**"seq 1"**,**"seq 2"**,**"seq 3"**,**"tri 1"**,**"tri 2"**,**"tri 3"**,**"ar 1"**,**"ar 2"**,**"ar 3"**,**"gr 1"**,**"gr 2"**,**"gr 3"**]  
 writer = csv.DictWriter(database, fieldnames = fieldnames)  
 hashed\_password = hashlib.md5(password.encode())  
 hashed\_password = (hashed\_password.hexdigest())  
 writer.writerow({**"first name"**:first\_name,**"last name"**:surname,**"username"**:username,**"password"**:hashed\_password,**"al 1"**:**"0"**,**"al 2"**:**"0"**,**"al 3"**:**"0"**,**"seq 1"**:**"0"**,**"seq 2"**:**"0"**,**"seq 3"**:**"0"**,**"tri 1"**:**"0"**,**"tri 2"**:**"0"**,**"tri 3"**:**"0"**,**"ar 1"**:**"0"**,**"ar 2"**:**"0"**,**"ar 3"**:**"0"**,**"gr 1"**:**"0"**,**"gr 2"**:**"0"**,**"gr 3"**:**"0"**})  
 *#creates a new row in the teacher\_accounts.csv file for the new teacher account details  
 #this new row includes the teachers details like their first name, surname, username and hashed password* **elif** user == **"teacher"**:  
 fieldnames = [**"first name"**, **"last name"**,**"username"**,**"password"**]  
 writer = csv.DictWriter(database, fieldnames = fieldnames)  
 hashed\_password = hashlib.md5(password.encode())  
 hashed\_password = (hashed\_password.hexdigest())  
 writer.writerow({**"first name"**:first\_name, **"last name"**:surname,**"username"**:username,**"password"**:hashed\_password})  
 *#closes the current window and calls forward the login\_page class* root.destroy()  
 login\_page(password\_username, user)  
  
 *#if the username is already in use by another user then a error message will appear in a message box* **elif** unique == **"1"**:  
 tk.messagebox.showerror(**"Error"**,**"That username is already taken, please try another"**)  
  
 *#the function back is called when the back button is presed  
 #the function takes the variables self, root, password\_username and user as parameters  
 #the purpose of the fucntion is to reload the login page* **def** back(self, root, password\_username, user):  
 root.destroy()  
 login\_page(password\_username, user)  
   
*#the teacher\_page class, creates the teacher page and the functions related to all the buttons on the teacher page  
#objective 31 is met in this class***class** teacher\_page:  
  
 *#creates the teacher page, creates all the buttons, labels and entry boxes in tkinter for the teacher page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, user and first\_load as parameters* **def** \_\_init\_\_(self, first\_load, user):  
  
  
 *#defines the variable as a global variable not a local* **global** user\_score  
  
 *#generates and places the window for the teacher page* root = tk.Tk()  
 root.geometry(**"350x250"**)  
 root.title(**"Teacher Page"**)  
  
 *#creates and places the lables needed for the teachers page* self.first\_name\_label = tk.Label(root, text = **"Students first name:"**).place(x= 0, y = 0)  
 self.last\_name\_label = tk.Label(root, text = **"Studets last name:"**).place(x=0,y=20)  
  
 *#creates and places the entry boxes needed for the teachers page* self.first\_name\_entry = tk.Entry(root)  
 self.last\_name\_entry = tk.Entry(root)  
 self.first\_name\_entry.place(x = 120,y=0)  
 self.last\_name\_entry.place(x = 120,y=20)  
  
 *#creates and places the button needed for the teachers page  
 #also creates the commands for when the buttons are pressed* self.get\_scores\_button = tk.Button(root, text = **"Get scores"**, command = **lambda**:self.get\_scores(root, first\_load, user))  
 self.get\_scores\_button.place(x = 260, y = 0)  
 self.get\_scores\_button.config(height = 2)  
   
 *#creates and places the labels needed for the first row of the table in the teachers page* self.lessonLabel = tk.Label(root, text= **"Lesson"** ).place(x = 0, y = 45)  
 self.scoreLabel = tk.Label(root, text=**" Scores: "**).place(x = 65, y = 45)  
 self.level1Label = tk.Label(root, text=**" Level 1 "**).place(x = 120, y = 45)  
 self.level2Label = tk.Label(root, text=**" Level 2 "**).place(x = 180, y = 45)  
 self.level3Label = tk.Label(root, text=**" Level 3 "**).place(x = 240, y = 45)  
  
 *#creates and places the labeles needed for the first column of the table in the teachers page* self.algebra\_label = tk.Label(root, text =**"Algebra"**).place(x = 0, y = 65)  
 self.triangleButton = tk.Label(root, text =**"Triangles"**).place(x = 0, y = 95)  
 self.sequenceButton = tk.Label(root, text =**"Sequences"**).place(x = 0, y = 125)  
 self.areaButton = tk.Label(root, text =**"Area"**).place(x = 0, y = 155)  
 self.areaButton = tk.Label(root, text =**"Graphs"**).place(x = 0, y = 185)  
  
 *#creates and places the log out button  
 #it also sets the command for the button as well as the size of the button* self.logoutButton = tk.Button(root, text=**"Log Out"**, command = **lambda**:self.log\_out(root, user))  
 self.logoutButton.place(x = 0, y = 220)   
 self.logoutButton.config(width = 8)  
  
 *#if this is the first time the page has been loaded then it fills the list with the string value "-"  
 #it does this for 15 values in the list* **if** first\_load == True:  
 **for** blank **in** range(0,15):  
 user\_score.append(**"-"**)  
  
 *#creates and places the labels with the users scores on or if its the first time its loaded then just "-"  
 #places the labels in a table format between the level and lesson each score relates to* self.al\_l1 = tk.Label(root, text = user\_score[0]).place(x = 140, y = 65)  
 self.al\_l2 = tk.Label(root, text = user\_score[1]).place(x = 200, y = 65)  
 self.al\_l3 = tk.Label(root, text = user\_score[2]).place(x = 260, y = 65)  
 self.tri\_l1 = tk.Label(root, text = user\_score[3]).place(x = 140, y = 95)  
 self.tri\_l2 = tk.Label(root, text = user\_score[4]).place(x = 200, y = 95)  
 self.tri\_l3 = tk.Label(root, text = user\_score[5]).place(x = 260, y = 95)  
 self.seq\_l1 = tk.Label(root, text = user\_score[6]).place(x = 140, y = 125)  
 self.seq\_l2 = tk.Label(root, text = user\_score[7]).place(x = 200, y = 125)  
 self.seq\_l3 = tk.Label(root, text = user\_score[8]).place(x = 260, y = 125)  
 self.ar\_l1 = tk.Label(root, text = user\_score[9]).place(x = 140, y = 155)  
 self.ar\_l2 = tk.Label(root, text = user\_score[10]).place(x = 200, y = 155)  
 self.ar\_l3 = tk.Label(root, text = user\_score[11]).place(x = 260, y = 155)  
 self.gr\_l1 = tk.Label(root, text = user\_score[12]).place(x = 140, y = 185)  
 self.gr\_l2 = tk.Label(root, text = user\_score[13]).place(x = 200, y = 185)  
 self.gr\_l3 = tk.Label(root, text = user\_score[14]).place(x = 260, y = 185)  
  
 *#the function get\_scores is called when the get scores button is pressed  
 #the function takes the variables self, root, user and first\_load as parameters  
 #the purpose of the funtion is to get the scores of the student that the teacher has inputted into the entry boxes  
 #objective 32 and 33 are met in this function* **def** get\_scores(self,root,first\_load, user):  
   
 *#defines the variables as global and not local* **global** user\_score, user\_row  
   
 *#defines the new variables needed for the function* user\_row = int(0)  
 user\_score = []  
  
 *#saves the users inputs from the entry boxes as seperate variables  
 #it also makes both the variables loweer case* first\_name = self.first\_name\_entry.get()  
 last\_name = self.last\_name\_entry.get()  
 first\_name = first\_name.lower()  
 last\_name = last\_name.lower()  
  
 *#redifines the variable as the boolean value false* first\_load = False  
  
 *#opens the student\_accounts.csv file and searchs for a user with the same first name and last name as the one the user has inputted* **with** open(**"student\_accounts.csv"**) **as** csvfile:  
 database = csv.DictReader(csvfile)  
 user\_row = 0  
 exist = False  
 **for** row **in** database:  
 user\_row = user\_row + 1  
 first\_name\_file = row[**'first name'**]  
 last\_name\_file = row[**'last name'**]  
 *#if there is a user with a matching name then that users scores are saved into the suser\_score list  
 #the teacher page is then reloaded* **if** first\_name\_file == first\_name **and** last\_name\_file == last\_name:  
 exist = True  
 database=open(**"student\_accounts.csv"**)  
 all\_details=[]  
 userScores = csv.reader(database)  
 **for** item **in** userScores:  
 all\_details.append(item)  
 **for** score **in** range(4,19):  
 user\_score.append(all\_details[user\_row][score])  
 root.destroy()  
 teacher\_page(first\_load, user)  
 **if** exist == False:   
 *#if there isn't a user with a matching name then a error message will appear in a message box* tk.messagebox.showerror(**"Error"**,**"A student by that name doesn't exist"**)  
  
 *#the function log\_out that is called when the log out button is pressed  
 #the function uses the variables self, root and user as parameters  
 #the purpose of the function is to log out the student and load the login page  
 #objective 15 is met in this function* **def** log\_out(self,root, user):  
 *#closes the current windown, assigns a variable needed for the login\_page class before calling the function* root.destroy()  
 password\_username = **"0"** login\_page(password\_username, user)  
  
*#the main\_page class, creates the main page and the functions related to all the buttons on the main page  
#objective 8 is met in this class***class** main\_page:  
   
 *#creates the main page, creates all the buttons, labels and entry boxes in tkinter for the main page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, and user as parameters* **def** \_\_init\_\_(self, user):  
  
 *#defines the variables as global and not local* **global** user\_score, user\_row  
  
 *#defines the extra variables needed for the main\_page class* score = int(0)  
 question = int(0)  
  
  
 root = tk.Tk()  
 root.geometry(**"350x250"**)   
 root.title(**"Main Page"**)  
  
 frame1 = tk.Frame()  
 frame1.config( width=600, height = 600 )  
 frame1.grid(row = 0,column = 0)  
 frame1.tkraise()  
   
 *#resigns the variable level* level = **"0"** *#creates and places the labels needed for the main page* self.lessonLabel = tk.Label(frame1, text= **"Lesson"** ).place(x = 0, y = 0)  
 self.scoreLabel = tk.Label(frame1, text=**" Scores: "**).place(x = 65, y = 0)  
 self.level1Label = tk.Label(frame1, text=**" Level 1 "**).place(x = 120, y = 0)  
 self.level2Label = tk.Label(frame1, text=**" Level 2 "**).place(x = 180, y = 0)  
 self.level3Label = tk.Label(frame1, text=**" Level 3 "**).place(x = 240, y = 0)  
  
 *#creates and places the algebra button  
 #it also sets the command for the button as well as the size of the button* self.algebraButton = tk.Button(frame1, text =**"Algebra"**, command = **lambda**:algebra(level, question, score))  
 self.algebraButton.place(x = 0, y = 20)  
 self.algebraButton.config(width = 8)  
   
 *#creates and places the triangle button  
 #it also sets the command for the button as well as the size of the button* self.triangleButton = tk.Button(frame1, text =**"Triangles"**, command = **lambda**:triangles(level, question, score))  
 self.triangleButton.place(x = 0, y = 50)  
 self.triangleButton.config(width = 8)  
  
 *#creates and places the sequences button  
 #it also sets the command for the button as well as the size of the button* self.sequenceButton = tk.Button(frame1, text =**"Sequences"**, command = **lambda**:sequences(level, question, score))  
 self.sequenceButton.place(x = 0, y = 80)  
 self.sequenceButton.config(width = 8)  
  
 *#creates and places the area button  
 #it also sets the command for the button as well as the size of the button* self.areaButton = tk.Button(frame1, text =**"Area"**, command = **lambda**:area(level, question, score))  
 self.areaButton.place(x = 0, y = 110)  
 self.areaButton.config(width = 8)  
  
 *#creates and places the graphs button  
 #it also sets the command for the button as well as the size of the button* self.areaButton = tk.Button(frame1, text =**"Graphs"**, command = **lambda**:graph(level, question, score))  
 self.areaButton.place(x = 0, y = 140)  
 self.areaButton.config(width = 8)  
  
 *#creates and places the refresh button  
 #it also sets the command for the button as well as the size of the button* self.refreshButton = tk.Button(frame1, text=**"Refresh"**, command = **lambda**:self.refresh(root, user))  
 self.refreshButton.place(x = 0, y = 190)  
 self.refreshButton.config(width = 8)  
  
 *#creates and places the log out button  
 #it also sets the command for the button as well as the size of the button* self.logoutButton = tk.Button(frame1, text=**"Log Out"**, command = **lambda**:self.log\_out(root, user))  
 self.logoutButton.place(x = 0, y = 220)   
 self.logoutButton.config(width = 8)  
  
 *#opens the student\_accounts.csv file and takes the logged in users scores as saves them in a list* database=open(**"student\_accounts.csv"**)  
 user\_details=[]  
 userScores = csv.reader(database)  
  
 **for** item **in** userScores:  
 user\_details.append(item)  
 **for** display\_score **in** range(4,19):  
 user\_score.append(user\_details[user\_row][display\_score])  
  
 *#creates and places the labels that display the users scores  
 #places them in a grid form between the lesson buttons and the level labels* self.al\_l1 = tk.Label(frame1, text = user\_score[0]).place(x = 140, y = 20)  
 self.al\_l2 = tk.Label(frame1, text = user\_score[1]).place(x = 200, y = 20)  
 self.al\_l3 = tk.Label(frame1, text = user\_score[2]).place(x = 260, y = 20)  
 self.tri\_l1 = tk.Label(frame1, text = user\_score[3]).place(x = 140, y = 50)  
 self.tri\_l2 = tk.Label(frame1, text = user\_score[4]).place(x = 200, y = 50)  
 self.tri\_l3 = tk.Label(frame1, text = user\_score[5]).place(x = 260, y = 50)  
 self.seq\_l1 = tk.Label(frame1, text = user\_score[6]).place(x = 140, y = 80)  
 self.seq\_l2 = tk.Label(frame1, text = user\_score[7]).place(x = 200, y = 80)  
 self.seq\_l3 = tk.Label(frame1, text = user\_score[8]).place(x = 260, y = 80)  
 self.ar\_l1 = tk.Label(frame1, text = user\_score[9]).place(x = 140, y = 110)  
 self.ar\_l2 = tk.Label(frame1, text = user\_score[10]).place(x = 200, y = 110)  
 self.ar\_l3 = tk.Label(frame1, text = user\_score[11]).place(x = 260, y = 110)  
 self.gr\_l1 = tk.Label(frame1, text = user\_score[12]).place(x = 140, y = 140)  
 self.gr\_l2 = tk.Label(frame1, text = user\_score[13]).place(x = 200, y = 140)  
 self.gr\_l3 = tk.Label(frame1, text = user\_score[14]).place(x = 260, y = 140)  
  
 *#the function refresh is called when the refresh button is pressed  
 #the function uses the variables self, root and user as parameters  
 #the purpose of the function is to reload the main page therefore updating the users scores* **def** refresh(self,root, user):  
 *#closes the current window and calls the main\_page function* root.destroy()  
 main\_page(user)  
   
 *#the function log\_out that is called when the log out button is pressed  
 #the function uses the variables self, root and user as parameters  
 #the purpose of the function is to log out the student and load the login page  
 #objective 15 is met in this function* **def** log\_out(self,root, user):  
 *#closes the current windown, assigns a variable needed for the login\_page class before calling the function* root.destroy()  
 password\_username = **"0"** login\_page(password\_username, user)  
  
*#the algebra class, creates the algebra question page and the functions related to all the buttons on the algebra questions page  
#objective 9 is met in this class***class** algebra:  
  
 *#creates the teacher page, creates all the buttons, labels and entry boxes in tkinter for the teacher page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, level, question and score as parameters* **def** \_\_init\_\_(self, level, question, score):  
  
 *#assigns the lesson variable* lesson = **"algebra"** *#creates a frame inside the window and raises it so that it is the frame that is viewable* frame2 = tk.Frame()  
 frame2.config( width=600, height = 600 )  
 frame2.grid(row = 0,column = 0)  
 frame2.tkraise()  
  
 *#increase the question variable before creating and placing a label showing the user what question they are on* question = question + 1  
 question\_number = tk.Label(frame2, text = str(question)+**"/5"**).place(x = 200, y = 0)  
  
 *#if the level hasn't been selected yet then the three level buttons are created* **if** level == **"0"**:  
  
 *#creates and places the choice a level label* self.levelpick = tk.Label(frame2, text=**"Pick a level"**).place(x =90, y = 0)  
  
 *#creates and places the three level buttons  
 #it also sets the command for each of the buttons* self.level1button = tk.Button(frame2, text = **"Level 1"**, command = **lambda**:self.level1(frame2, lesson, question, score))  
 self.level1button.place(x = 0, y = 30)  
 self.level2button = tk.Button(frame2, text = **"Level 2"**,command = **lambda**:self.level2(frame2, lesson, question, score))  
 self.level2button.place(x = 100, y = 30)  
 self.level3button = tk.Button(frame2, text = **"Level 3"**, command = **lambda**:self.level3(frame2, lesson, question, score))  
 self.level3button.place(x = 200, y = 30)  
  
 *#calls forward the corresponding function to the value of the variable level  
 #the funtion level1 is called if the variable level equals 1  
 #the funtion level2 is called if the variable level equals 2  
 #the funtion level3 is called if the variable level equals 3* **if** level == **"1"**:  
 self.level1(frame2, lesson, question, score)  
 **elif** level == **"2"**:  
 self.level2(frame2, lesson, question, score)  
 **elif** level == **"3"**:  
 self.level3(frame2, lesson, question, score)  
  
 *#the function level1 one is called when the level 1 button is pressed or the variable level = 1  
 #the function level1 takes the variables self, frame2, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 1 questions of the algebra lesson  
 #objective 16 is met in this function* **def** level1(self, frame2, lesson, question, score):  
   
 *#assigns the level variables* level = **"1"** *#randomly generates the values used in the question  
 #also generates the answer* Xvalue = random.randint(3,10)  
 answer = random.randint(3,8)  
 rhs\_unit = Xvalue\*answer  
   
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame2, text = str(Xvalue)+**"x = "**+str(rhs\_unit)).place(x = 0, y = 60)  
 self.QLine2 = tk.Label(frame2, text = **"What does x equal?"**).place(x=0 , y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* self.answerLabel = tk.Label(frame2, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame2)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame2, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame2, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)  
  
 *#the function level2 one is called when the level 2 button is pressed or the variable level = 2  
 #the function level2 takes the variables self, frame2, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 2 questions of the algebra lesso  
 #objective 17 is met in this function* **def** level2(self, frame2, lesson, question, score):  
   
 *#assigns the level variable* level = **"2"** *#randomly generates the values used in the question  
 #also generates the answer* Xvalue = random.randint(3,10)  
 answer = random.randint(3,8)  
 rhs\_unit = Xvalue\*answer  
 lhs\_unit = random.randint(5,rhs\_unit-3)  
 negORpos = random.choice([True,False])  
   
 *#generates the question in one of two forms depending on the value of the varible negORpos* **if** negORpos == True:  
 rhs\_unit = rhs\_unit - lhs\_unit   
 questionLine1 = str(str(Xvalue)+**"x -"**+str(lhs\_unit)+**"="**+str(rhs\_unit))   
 **else**:  
 rhs\_unit = rhs\_unit + lhs\_unit  
 questionLine1 = str(str(Xvalue)+**"x +"**+str(lhs\_unit)+**"="**+str(rhs\_unit))  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame2, text = questionLine1).place(x = 0, y = 60)  
 self.QLine2 = tk.Label(frame2, text = **"What does x equal?"**).place(x=0 , y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* self.answerLabel = tk.Label(frame2, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame2)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame2, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame2, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)   
  
 *#the function level3 one is called when the level 3 button is pressed or the variable level = 3  
 #the function level3 takes the variables self, frame2, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 3 questions of the algebra lesson  
 #objective 18 is met in this function* **def** level3(self, frame2, lesson, question, score):  
   
 *#assigns the variable level* level = **"3"** *#randomly generates the values used in the question  
 #also generates the answer* Xvalue = random.randint(5,14)  
 answer = random.randint(3,8)  
 rhs\_unit = Xvalue\*answer  
 lhs\_unit = random.randint(5,rhs\_unit-5)  
 rhs\_xValue = random.randint(2,Xvalue-2)  
 negORpos = random.choice([True,False])  
 negORpos2 = random.choice([True,False])  
   
 *#generates the question in one of four forms depending on the value of the varible negORpos and the variable negORpos2* **if** negORpos == True:  
 rhs\_unit = rhs\_unit - lhs\_unit  
 **if** negORpos2 == True:  
 lhs\_xValue = Xvalue - rhs\_xValue  
 questionLine1 = str(str(lhs\_xValue)+ **"x -"**+str(lhs\_unit)+**"="**+str(rhs\_unit)+**"-"**+str(rhs\_xValue)+**"x"**)  
 **else**:  
 lhs\_xValue = Xvalue + rhs\_xValue  
 questionLine1 = str(str(lhs\_xValue)+ **"x -"**+str(lhs\_unit)+**"="**+str(rhs\_unit)+**"+"**+str(rhs\_xValue)+**"x"**)  
 **else**:  
 rhs\_unit = rhs\_unit + lhs\_unit  
 **if** negORpos2 == True:  
 lhs\_xValue = Xvalue - rhs\_xValue  
 questionLine1 = str(str(lhs\_xValue)+ **"x +"** +str(lhs\_unit)+**"="**+str(rhs\_unit)+**"-"**+str(rhs\_xValue)+**"x"**)  
 **else**:  
 lhs\_xValue = Xvalue + rhs\_xValue  
 questionLine1 = str(str(lhs\_xValue)+ **"x +"**+str(lhs\_unit)+**"="**+str(rhs\_unit)+**"+"**+str(rhs\_xValue)+**"x"**)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame2, text = questionLine1).place(x = 0, y = 60)  
 self.Qline2 = tk.Label(frame2, text = **"What does x equal?"**).place(x=0 , y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* self.answerLabel = tk.Label(frame2, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame2)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame2, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame2, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)   
  
 *#the function get\_user\_answer is called when the submit button in pressed  
 #the function takes the variables self, frame2, answer, level, lesson, question and score as parameters  
 #the purpose of the function is to get the users answer before calling the check\_answer function* **def** get\_user\_answer(self, frame2, answer, level, lesson, question, score):  
   
 *#gets the answer the user has inputted and saves as a variable  
 #assigns the frame variable and then calls the check\_answer function* user\_answer = self.user\_answer\_entry.get()  
 frame = frame2  
 check\_answer(frame, answer, user\_answer, level, lesson, question, score)  
  
*#the login page class, creates the login page and the functions related to all the buttons on the login page  
#objective 9 and 11 is met in this class***class** triangles():  
   
 *#creates the triangle question page, creates all the buttons, labels and entry boxes in tkinter for the triangle question page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self and first\_load as parameters* **def** \_\_init\_\_(self, level, question, score):  
  
 *#assigns the variable lesson* lesson = **"triangles"** *#creates a frame inside the window and raises it so that it is the frame that is viewable* frame3 = tk.Frame()  
 frame3.config( width=600, height = 600 )  
 frame3.grid(row = 0,column = 0)  
 frame3.tkraise()  
  
 *#increase the question variable before creating and placing a label showing the user what question they are on* question = question + 1  
 question\_number = tk.Label(frame3, text = str(question)+**"/5"**).place(x = 200, y = 0)  
  
 *#if the level hasn't been selected yet then the three level buttons are created* **if** level == **"0"**:  
  
 *#creates and places the choice a level label* self.levelpick = tk.Label(frame3, text=**"Pick a level"**).place(x =90, y = 0)  
  
 *#creates and places the three level buttons  
 #it also sets the command for each of the buttons* self.level1button = tk.Button(frame3, text = **"Level 1"**, command = **lambda**:self.level1(frame3, lesson, question, score))  
 self.level1button.place(x = 0, y = 30)  
 self.level2button = tk.Button(frame3, text = **"Level 2"**,command = **lambda**:self.level2(frame3, lesson, question, score))  
 self.level2button.place(x = 100, y = 30)  
 self.level3button = tk.Button(frame3, text = **"Level 3"**, command = **lambda**:self.level3(frame3, lesson, question, score))  
 self.level3button.place(x = 200, y = 30)  
  
 *#calls forward the corresponding function to the value of the variable level  
 #the funtion level1 is called if the variable level equals 1  
 #the funtion level2 is called if the variable level equals 2  
 #the funtion level3 is called if the variable level equals 3* **if** level == **"1"**:  
 self.level1(frame3, lesson, question, score)  
 **elif** level == **"2"**:  
 self.level2(frame3, lesson, question, score)  
 **elif** level == **"3"**:  
 self.level3(frame3, lesson, question, score)  
   
 *#the function level1 one is called when the level 1 button is pressed or the variable level = 1  
 #the function level1 takes the variables self, frame3, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 1 questions of the triangles lesson  
 #objective 22 is met in this function* **def** level1(self, frame3, lesson, question, score):  
  
 *#assigns the level variable* level = **"1"** *#randomly generates the values used in the question  
 #also generates the answer* combinedAngles = int(180)  
 **while** combinedAngles > 170:  
 angleA = random.randint(10,120)  
 angleB = random.randint(10,120)  
 combinedAngles = angleA + angleB  
 answer = (180 - angleA)-angleB  
  
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* photo = tk.PhotoImage(file = **"triangleslevel1.png"**)  
 self.pic\_label = tk.Label(frame3, image=photo)  
 self.pic\_label.image = photo  
 self.pic\_label.place(x = 135, y = 70)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame3, text = **"Angle A = "**+str(angleA)+**" degrees"**).place(x = 0, y = 120)  
 self.QLine2 = tk.Label(frame3, text = **"Angle B = "**+str(angleB)+**" degrees"**).place(x = 0, y = 150)  
 self.QLine3 = tk.Label(frame3, text = **"What is the size of angle C?"**).place(x = 0, y = 180)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* self.answerLabel = tk.Label(frame3, text = **"Answer:"**).place(x = 0, y = 210)  
 self.user\_answer\_entry = tk.Entry(frame3)  
 self.user\_answer\_entry.place(x = 75, y = 210)  
 self.enter = tk.Button(frame3, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame3, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 210)  
  
 *#the function level2 one is called when the level 2 button is pressed or the variable level = 2  
 #the function level2 takes the variables self, frame3, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 2 questions of the triangles lesson  
 #objective 23 is met in this function* **def** level2(self, frame3, lesson, question, score):  
  
 *#assigns the level variable* level = **"2"** *#randomly generates the values used in the question  
 #also generates the answer  
 #genertates the question in on of two forms depending on the value of abORbc* abORbc = random.choice([True,False])  
 **if** abORbc == True:  
 sideA = random.randint(2,12)  
 sideB = random.randint(2,12)  
 answer = float((sideA\*\*2 + sideB\*\*2)\*\*0.5)  
 answer = round(answer,1)  
 questionLine1 = str(**"Side A = "**+str(sideA)+**"cm"**)   
 questionLine2 = str(**"Side B = "**+str(sideB)+**"cm"**)  
 QLine3 = tk.Label(frame3, text = **"What is the length of side C (to one decimal place): "**).place(x = 0, y = 180)  
   
 **elif** abORbc == False:  
 sideC = random.randint(6,20)  
 sideB = random.randint(2,sideC-2)  
 answer = float((sideC\*\*2 - sideB\*\*2)\*\*0.5)  
 answer = round(answer, 1)  
 questionLine1 = str(**"Side B = "**+ str(sideB)+**"cm"**)  
 questionLine2 = str(**"Side C = "**+str(sideC)+**"cm"**)  
 QLine3 = tk.Label(frame3, text = **"What is the length of side A (to one decimal place): "**).place(x = 0, y = 180)  
  
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* photo = tk.PhotoImage(file = **"triangleslevel2.png"**)  
 self.pic\_label = tk.Label(frame3, image=photo)  
 self.pic\_label.image = photo  
 self.pic\_label.place(x = 120, y = 65)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame3, text = questionLine1).place(x = 0, y = 120)  
 self.QLine2 = tk.Label(frame3, text = questionLine2).place(x = 0, y = 150)  
  
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* tk.Label(frame3, text = **"Answer:"**).place(x = 0, y = 210)  
 self.user\_answer\_entry = tk.Entry(frame3)  
 self.user\_answer\_entry.place(x = 75, y = 210)   
 self.enter = tk.Button(frame3, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame3, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 210)  
  
 *#the function level3 one is called when the level 3 button is pressed or the variable level = 3  
 #the function level3 takes the variables self, frame3, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 3 questions of the triangles lesson  
 #objective 24 is met in this function* **def** level3(self, frame3, lesson, question, score):  
  
 *#assigns the level variable* level = **"3"** *#randomly generates the values used in the question* combinedAngles = int(180)  
 **while** combinedAngles > 170:  
 angleA = random.randint(10,120)  
 angleB = random.randint(10,120)  
 angleC = int((180 - angleB)-angleA)  
 combinedAngles = angleA + angleB  
  
 strangleA = str(angleA)  
 strangleB = str(angleB)  
   
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* photo = tk.PhotoImage(file = **"triangleslevel3.png"**)  
 self.pic\_label = tk.Label(frame3, image=photo)  
 self.pic\_label.image = photo  
 self.pic\_label.place(x = 100, y = 65)  
  
 *#generates the answer* angleA = math.radians(angleA)  
 angleC = math.radians(angleC)  
 sideA = random.randint(2,12)  
 sideB = random.randint(2,12)  
 sinA = math.sin(angleA)  
 sinC = math.sin(angleC)  
 answer = float((sideA/sinA)\*sinC)  
 answer = round(answer,1)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame3, text = str(**"Angle A = "**+ strangleA)).place(x = 0, y = 60)  
 self.QLine2 = tk.Label(frame3, text = str(**"Angle B = "**+ strangleB)).place(x = 0, y = 90)  
 self.QLine3 = tk.Label(frame3, text = str(**"Side A = "**+ str(sideA))).place(x = 0, y = 120)  
 self.QLine4 = tk.Label(frame3, text = str(**"Side B = "**+ str(sideB))).place(x = 0, y = 150)  
 self.QLine5 = tk.Label(frame3, text = **"What is the length of side C (to one deciaml place)?"**).place(x = 0, y = 180)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame3, text = **"Answer:"**).place(x = 0, y = 210)  
 self.user\_answer\_entry = tk.Entry(frame3)  
 self.user\_answer\_entry.place(x = 75, y = 210)  
 self.enter = tk.Button(frame3, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame3, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 210)  
  
 *#the function get\_user\_answer is called when the submit button in pressed  
 #the function takes the variables self, frame3, answer, level, lesson, question and score as parameters  
 #the purpose of the function is to get the users answer before calling the check\_answer function* **def** get\_user\_answer(self, frame3, answer, level, lesson, question, score):  
  
 *#gets the answer the user has inputted and saves as a variable  
 #assigns the frame variable and then calls the check\_answer function* user\_answer = self.user\_answer\_entry.get()  
 frame = frame3  
 check\_answer(frame, answer, user\_answer, level, lesson, question, score)  
  
*#the sequences class, creates the sequence question page and the functions related to all the buttons on the sequence question page  
#objective 9 is met in this class***class** sequences():  
  
 *#creates the teacher page, creates all the buttons, labels and entry boxes in tkinter for the teacher page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, level question and score as parameters* **def** \_\_init\_\_(self, level, question, score):  
  
 *#assigns the lesson variable* lesson = **"sequences"** *#creates a frame inside the window and raises it so that it is the frame that is viewable* frame4 = tk.Frame()  
 frame4.config( width=600, height = 600 )  
 frame4.grid(row = 0,column = 0)   
 frame4.tkraise()  
  
 *#increase the question variable before creating and placing a label showing the user what question they are on* question = question + 1  
 question\_number = tk.Label(frame4, text = str(question)+**"/5"**).place(x = 200, y = 0)  
  
 *#if the level hasn't been selected yet then the three level buttons are created* **if** level == **"0"**:  
   
 *#creates and places the choice a level label* self.levelpick = tk.Label(frame4, text=**"Pick a level"**).place(x =90, y = 0)  
   
 *#creates and places the three level buttons  
 #it also sets the command for each of the buttons* self.level1button = tk.Button(frame4, text = **"Level 1"**, command = **lambda**:self.level1(frame4, lesson, question, score))  
 self.level1button.place(x = 0, y = 30)  
 self.level2button = tk.Button(frame4, text = **"Level 2"**,command = **lambda**:self.level2(frame4, lesson, question, score))  
 self.level2button.place(x = 100, y = 30)  
 self.level3button = tk.Button(frame4, text = **"Level 3"**, command = **lambda**:self.level3(frame4, lesson, question, score))  
 self.level3button.place(x = 200, y = 30)  
  
 *#calls forward the corresponding function to the value of the variable level  
 #the funtion level1 is called if the variable level equals 1  
 #the funtion level2 is called if the variable level equals 2  
 #the funtion level3 is called if the variable level equals 3* **if** level == **"1"**:  
 self.level1(frame4, lesson, question, score)  
 **elif** level == **"2"**:  
 self.level2(frame4, lesson, question, score)  
 **elif** level == **"3"**:  
 self.level3(frame4, lesson, question, score)  
  
 *#the function level1 one is called when the level 1 button is pressed or the variable level = 1  
 #the function level1 takes the variables self, frame4, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 1 questions of the sequences lesson  
 #objective 25 is met in this function* **def** level1(self, frame4, lesson, question, score):  
  
 *#assigns the variable level* level = **"1"** *#randomly generates the values used in the question* xTerm = random.randint(3,10)  
 numTerm = random.randint(3,15)  
 negORpos = random.choice([True,False])  
 nTerm = random.randint(4,10)  
  
 *#genertates the question in on of two forms depending on the value of negORpos* **if** negORpos == True:  
 questionLine1 = str(**"The nth term is: "**+str(xTerm)+**"X +"**+str(numTerm))   
 **else**:  
 questionLine1 = str(**"The nth term is: "**+str(xTerm)+**"X - "**+str(numTerm))  
 numTerm = numTerm \* -1  
  
 *#generates the answer* answer = ((xTerm\*nTerm)+numTerm)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine2 = tk.Label(frame4, text = str(**"Work out the "**+str(nTerm)+**"th term"**)).place(x = 0, y = 60)  
 self.Qline1 = tk.Label(frame4, text = questionLine1).place(x = 0, y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame4, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame4)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame4, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame4, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)  
  
 *#the function level2 one is called when the level 1 button is pressed or the variable level = 2  
 #the function level2 takes the variables self, frame4, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 2 questions of the sequences lesson  
 #objective 26 is met in this function* **def** level2(self, frame4, lesson, question, score):  
  
 *#assigns the level variable* level = **"2"** *#randomly generates the values used in the question* xTerm = random.randint(2,6)  
 numTerm = random.randint(3,15)  
 negORpos = random.choice([True,False])  
 nTerm = random.randint(4,10)  
  
 *#genertates the question in on of two forms depending on the value of negORpos* **if** negORpos == True:  
 questionLine1 = str(**"The nth term is: "**+str(xTerm)+**"X^2 +"**+str(numTerm))   
 **else**:  
 questionLine1 = str(**"The nth term is: "**+str(xTerm)+**"X^2 - "**+str(numTerm))  
 numTerm = numTerm \* -1  
   
 *#generates the answer* answer = (((nTerm\*\*2)\*xTerm)+numTerm)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine2 = tk.Label(frame4, text = str(**"Work out the "**+str(nTerm)+**"th term"**)).place(x = 0, y = 60)  
 self.QLine1 = tk.Label(frame4, text = questionLine1).place(x = 0, y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame4, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame4)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame4, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame4, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)  
  
 *#the function level3 one is called when the level 3 button is pressed or the variable level = 3  
 #the function level3 takes the variables self, frame4, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 3 questions of the sequences lesson  
 #objective 27 is met in this function* **def** level3(self, frame4, lesson, question, score):  
  
 *#assigns the variable level* level = **"3"** *#randomly generates the values used in the question* xTerm = random.randint(2,5)  
 numTerm = random.randint(1,5)  
 negORpos = random.choice([True,False])  
 termList = []  
  
 *#genertates the question in on of two forms depending on the value of negORpos* **if** negORpos == True:  
 numTerm = numTerm \* -1  
 **for** i **in** range (1,6):  
 term = ((xTerm\*i)+numTerm)  
 termList.append(term)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame4, text = **"The first five terms of the sequence are:"**).place(x = 0, y = 60)  
 self.QLine2 = tk.Label(frame4, text = str(termList)).place(x = 0, y = 90)  
 self.QLine3 = tk.Label(frame4, text = **"What is the nth term (in terms of x)"**).place(x = 0, y = 120)  
  
 *#generates the answer* **if** negORpos == True:  
 answer = (str(xTerm)+**"x"**+str(numTerm))  
 **else**:  
 answer = (str(xTerm)+**"x+"**+str(numTerm))  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame4, text = **"Answer:"**).place(x = 0, y = 150)  
 self.user\_answer\_entry = tk.Entry(frame4)  
 self.user\_answer\_entry.place(x = 75, y = 150)  
 self.enter = tk.Button(frame4, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame4, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 150)  
  
 *#the function get\_user\_answer is called when the submit button in pressed  
 #the function takes the variables self, frame4, answer, level, lesson, question and score as parameters  
 #the purpose of the function is to get the users answer before calling the check\_answer function* **def** get\_user\_answer(self, frame4, answer, level, lesson, question, score):  
 user\_answer = self.user\_answer\_entry.get()  
 frame = frame4  
 check\_answer(frame, answer, user\_answer, level, lesson, question, score)  
  
*#the login page class, creates the login page and the functions related to all the buttons on the login page  
#objective 9 is met in this class***class** area():  
  
 *#creates the teacher page, creates all the buttons, labels and entry boxes in tkinter for the teacher page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self, level, question and score as parameters* **def** \_\_init\_\_(self, level, question, score):  
  
 *#assigns the lesson variable* lesson = **"area"** *#creates a frame inside the window and raises it so that it is the frame that is viewable* frame5 = tk.Frame()  
 frame5.config( width=600, height = 600 )  
 frame5.grid(row = 0,column = 0)   
 frame5.tkraise()  
  
 *#increase the question variable before creating and placing a label showing the user what question they are on* question = question + 1  
 question\_number = tk.Label(frame5, text = str(question)+**"/5"**).place(x = 200, y = 0)  
  
 *#if the level hasn't been selected yet then the three level buttons are created* **if** level == **"0"**:  
   
 *#creates and places the choice a level label* self.levelpick = tk.Label(frame5, text=**"Pick a level"**).place(x =90, y = 0)  
   
 *#creates and places the three level buttons  
 #it also sets the command for each of the buttons* self.level1button = tk.Button(frame5, text = **"Level 1"**, command = **lambda**:self.level1(frame5, lesson, question, score))  
 self.level1button.place(x = 0, y = 30)  
 self.level2button = tk.Button(frame5, text = **"Level 2"**,command = **lambda**:self.level2(frame5, lesson, question, score))  
 self.level2button.place(x = 100, y = 30)  
 self.level3button = tk.Button(frame5, text = **"Level 3"**, command = **lambda**:self.level3(frame5, lesson, question, score))  
 self.level3button.place(x = 200, y = 30)  
  
 *#calls forward the corresponding function to the value of the variable level  
 #the funtion level1 is called if the variable level equals 1  
 #the funtion level2 is called if the variable level equals 2  
 #the funtion level3 is called if the variable level equals 3* **if** level == **"1"**:  
 self.level1(frame5, lesson, question, score)  
 **elif** level == **"2"**:  
 self.level2(frame5, lesson, question, score)  
 **elif** level == **"3"**:  
 self.level3(frame5, lesson, question, score)  
  
 *#the function level1 one is called when the level 1 button is pressed or the variable level = 1  
 #the function level1 takes the variables self, frame5, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 1 questions of the triangle lesson  
 #objective 19 is met in this function* **def** level1(self, frame5, lesson, question, score):  
  
 *#assigns the level variable* level = **"1"** *#randomly generates the values used in the question  
 #also generates the answer* sideA = random.randint(3,12)  
 sideB = random.randint(3,12)  
 **while** sideB == sideA:  
 sideB = random.randint(3,12)  
 strSideA = str(sideA)  
 strSideB = str(sideB)  
 questionLine1= str(**"Side A = "** +strSideA)  
 questionLine2 = str(**"Side B = "**+strSideB)  
 answer = sideA\*sideB  
  
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* photo = tk.PhotoImage(file = **"arealevel1.png"**)  
 self.pic\_label = tk.Label(frame5, image=photo)  
 self.pic\_label.image = photo  
 self.pic\_label.place(x = 100, y = 65)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame5, text = questionLine1).place(x = 0, y = 120)  
 self.QLine2 = tk.Label(frame5, text = questionLine2).place(x = 0, y = 150)  
 self.QLine3 = tk.Label(frame5, text = **"What is the area of the rectangle?"**).place(x = 0, y = 180)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame5, text = **"Answer:"**).place(x = 0, y = 210)  
 self.user\_answer\_entry = tk.Entry(frame5)  
 self.user\_answer\_entry.place(x = 75, y = 210)  
 self.enter = tk.Button(frame5, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame5, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 210)  
  
 *#the function level2 one is called when the level 2 button is pressed or the variable level = 2  
 #the function level2 takes the variables self, frame5, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 2 questions of the triangle lesson  
 #objective 20 is met in this function* **def** level2(self, frame5, lesson, question, score):  
  
 *#assigns the level variable* level = **"2"** *#randomly generates the values used in the question  
 #also generates the answer* radius = random.randint(3,15)  
 questionLine1 = str(**"The radius of the circle is "**+str(radius)+**"cm"**)  
 self.QLine1 = tk.Label(frame5, text = questionLine1).place(x = 0, y = 150)  
 self.QLine2 = tk.Label(frame5, text = **"What is the area of the circle (to one deciaml place):"**).place(x = 0, y = 180)  
 pi = math.pi  
 answer = ((radius\*\*2)\*pi)  
 answer = round(answer,1)  
  
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* photo = tk.PhotoImage(file = **"arealevel2.png"**)  
 self.pic\_label = tk.Label(frame5, image=photo)  
 self.pic\_label.image = photo  
 self.pic\_label.place(x = 180, y = 65)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame5, text = **"Answer:"**).place(x = 0, y = 210)  
 self.user\_answer\_entry = tk.Entry(frame5)  
 self.user\_answer\_entry.place(x = 75, y = 210)  
 self.enter = tk.Button(frame5, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame5, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 210)  
  
 *#the function level3 one is called when the level 3 button is pressed or the variable level = 3  
 #the function level3 takes the variables self, frame5, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 3 questions of the triangle lesson  
 #objective 21 is met in this function* **def** level3(self, frame5, lesson, question, score):  
  
 *#assigns the variabel level* level = **"3"** *#randomly generates the values used in the question  
 #also generates the answer* sideA = random.randint(2,14)  
 sideB = random.randint(2,14)  
 height = random.randint(2,8)  
  
 *#creates a label to display the picture that goes with the question  
 #places the label at a set location* photo = tk.PhotoImage(file = **"arealevel3.png"**)  
 self.pic\_label = tk.Label(frame5, image=photo)  
 self.pic\_label.image = photo  
 self.pic\_label.place(x = 100, y = 65)  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame5, text = str(**"Side A = "**+str(sideA))).place(x = 0, y = 90)  
 self.QLine2 = tk.Label(frame5, text = str(**"Side B = "**+str(sideB))).place(x = 0, y = 120)  
 self.QLine3 = tk.Label(frame5, text = str(**"Height = "**+str(height))).place(x = 0, y = 150)  
 self.QLine4 = tk.Label(frame5, text = **"What is the area of the trapezoid (to one decimal place):"**).place(x=0, y = 180)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame5, text = **"Answer:"**).place(x = 0, y = 210)  
 self.user\_answer\_entry = tk.Entry(frame5)  
 self.user\_answer\_entry.place(x = 75, y = 210)  
 answer = float((0.5\*(sideA+sideB))\*height)  
 self.enter = tk.Button(frame5, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame5, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 210)  
  
 *#the function get\_user\_answer is called when the submit button in pressed  
 #the function takes the variables self, frame2, answer, level, lesson, question and score as parameters  
 #the purpose of the function is to get the users answer before calling the check\_answer function* **def** get\_user\_answer(self, frame5, answer, level, lesson, question, score):  
 user\_answer = self.user\_answer\_entry.get()  
 frame = frame5  
 check\_answer(frame, answer, user\_answer, level, lesson, question, score)  
  
*#the graph class, creates the graph question page and the functions related to all the buttons on the graph question page  
#objective 9, 10 and 11 are met in this class***class** graph:  
  
 *#creates the graph question page, creates all the buttons, labels and entry boxes in tkinter for the graph question page  
 #this is the function that automatically runs when the class is called  
 #the function has the variables self and level, question and score as parameters* **def** \_\_init\_\_(self, level, question, score):  
  
 *#assigns the variable lesson* lesson = **"graph"** *#creates a frame inside the window and raises it so that it is the frame that is viewable* frame6 = tk.Frame()  
 frame6.config( width=600, height = 600 )  
 frame6.grid(row = 0,column = 0)   
 frame6.tkraise()  
  
 *#increase the question variable before creating and placing a label showing the user what question they are on* question = question + 1  
 question\_number = tk.Label(frame6, text = str(question)+**"/5"**).place(x = 200, y = 0)  
  
 *#if the level hasn't been selected yet then the three level buttons are created* **if** level == **"0"**:  
  
 *#creates and places the choice a level label* self.levelpick = tk.Label(frame6, text=**"Pick a level"**).place(x =90, y = 0)  
   
 *#creates and places the three level buttons  
 #it also sets the command for each of the buttons* self.level1button = tk.Button(frame6, text = **"Level 1"**, command = **lambda**:self.level1(frame6, lesson, question, score))  
 self.level1button.place(x = 0, y = 30)  
 self.level2button = tk.Button(frame6, text = **"Level 2"**,command = **lambda**:self.level2(frame6, lesson, question, score))  
 self.level2button.place(x = 100, y = 30)  
 self.level3button = tk.Button(frame6, text = **"Level 3"**, command = **lambda**:self.level3(frame6, lesson, question, score))  
 self.level3button.place(x = 200, y = 30)  
  
 *#calls forward the corresponding function to the value of the variable level  
 #the funtion level1 is called if the variable level equals 1  
 #the funtion level2 is called if the variable level equals 2  
 #the funtion level3 is called if the variable level equals 3* **if** level == **"1"**:  
 self.level1(frame6, lesson, question, score)  
 **elif** level == **"2"**:  
 self.level2(frame6, lesson, question, score)  
 **elif** level == **"3"**:  
 self.level3(frame6, lesson, question, score)  
  
 *#the function level1 one is called when the level 1 button is pressed or the variable level = 1  
 #the function level1 takes the variables self, frame6, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 1 questions of the graph lesson  
 #objective 28 is met in this function* **def** level1(self, frame6, lesson, question, score):  
  
 *#assigns the variable level* level = **"1"** *#randomly generates the values used in the question  
 #also generates the answer* xFactor = random.randint(2,5)  
 unit = random.randint(1,5)  
 yValues = []  
 xValue = random.randint(1,5)  
 answer = (xValue\*xFactor)+unit  
  
 *#creates and places the question label which are used to show the user the question* self.QLine1 = tk.Label(frame6, text = **"What is the value of Y when X is "**+str(xValue)+**":"**).place(x = 0, y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame6, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame6)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame6, text = **"Submit"**, command = **lambda**:self.get\_user\_answer(frame6, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)  
  
 *#creates a list of the y vales of the graph* **for** x **in** range(1,6):  
 y = (x\*xFactor)+unit  
 yValues.append(y)  
  
 *#plots the graph in a seperate window and shows it* plt.plot([1,2,3,4,5],yValues)  
 plt.show()  
  
 *#the function level2 one is called when the level 2 button is pressed or the variable level = 2  
 #the function level2 takes the variables self, frame6, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 2 questions of the graph lesson  
 #objective 29 is met in this function* **def** level2(self, frame6, lesson, question, score):  
  
 *#assigns the level variable* level = **"2"** *#randomly generates the values used in the question* xFactor = random.randint(2,5)  
 unit = random.randint(1,5)  
 yValues = []  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame6, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame6)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame6, text = **"Submit"**,command = **lambda**:self.get\_user\_answer(frame6, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)  
  
 *#creates and places the question labels which are used to show the user the question* Qline2 = tk.Label(frame6, text = **"True or False:"**).place(x=0,y=90)  
  
 *#generates the answer to the question based on the value of TrueOrFalse* TrueOrFalse = random.choice([True,False])  
 **if** TrueOrFalse == True:  
 answer = **"true"** *#creates and places the question labels which are used to show the user the question* QLine1 = tk.Label(frame6, text = **"Is this graph "**+str(xFactor)+**"x +"**+str(unit)).place(x=0, y=60)  
 **else**:  
 answer = **"false"** falsexFactor = random.randint(2,5)  
 **while** falsexFactor == xFactor:  
 falsexFactor = random.randint(2,5)  
 falseUnit = random.randint(1,5)  
  
 *#creates and places the question labels which are used to show the user the question* QLine1 = tk.Label(frame6, text = **"Is this graph "**+str(falsexFactor)+**"x +"**+str(falseUnit)).place(x=0, y=60)  
  
 *#creates a list of the y vales of the graph* **for** x **in** range(1,6):  
 y = (x\*xFactor)+unit  
 yValues.append(y)  
   
 *#plots the graph in a seperate window and shows it* plt.plot([1,2,3,4,5],yValues)  
 plt.show()  
  
 *#the function level3 one is called when the level 3 button is pressed or the variable level = 3  
 #the function level3 takes the variables self, frame6, lesson, question and score as parameters  
 #the purpose of the function is to generate the level 3 questions of the graph lesson  
 #objective 30 is met in this function* **def** level3(self, frame6, lesson, question, score):  
  
 *#assigns the level variable* level = **"3"** *#randomly generates the values used in the question* typeOfGraph = random.randint(1,3)  
 xValue = random.randint(2,5)  
 unit = random.randint(1,5)  
 yValues = []  
  
 *#creates and places the question labels which are used to show the user the question* self.QLine1 = tk.Label(frame6, text = **"Is the graph cubic, quadratic or linear:"**).place(x = 0, y = 90)  
  
 *#creates and places the answer label, entry box and submit button  
 #it also assigns the comand for the submit button* tk.Label(frame6, text = **"Answer:"**).place(x = 0, y = 120)  
 self.user\_answer\_entry = tk.Entry(frame6)  
 self.user\_answer\_entry.place(x = 75, y = 120)  
 self.enter = tk.Button(frame6, text = **"Submit"**,command = **lambda**:self.get\_user\_answer(frame6, answer, level, lesson, question, score))  
 self.enter.place(x = 200, y = 120)  
  
 *#generates a list of y values based on the value of the variabe tyoeOfGraph* **if** typeOfGraph == 1:  
 **for** x **in** range(-5,6):  
 y = (x\*\*3)+(x\*\*2)+(x\*xValue)+unit  
 yValues.append(y)  
   
 *#generates the answer* answer = **"cubic"  
   
 elif** typeOfGraph == 2:  
 **for** x **in** range(-5,6):  
 y = (x\*\*2)+(x\*xValue)+unit  
 yValues.append(y)  
   
 *#generates the answer* answer = **"quadratic"  
   
 elif** typeOfGraph == 3:  
 **for** x **in** range(-5,6):  
 y = (x\*xValue)+unit  
 yValues.append(y)  
   
 *#generates the answer* answer = **"linear"** *#creates a list of the y vales of the graph  
 #plots the graph in a seperate window and shows it* plt.plot([-5,-4,-3,-2,-1,0,1,2,3,4,5],yValues)  
 plt.show()  
  
 *#the function get\_user\_answer is called when the submit button in pressed  
 #the function takes the variables self, frame6, answer, level, lesson, question and score as parameters  
 #the purpose of the function is to get the users answer before calling the check\_answer function  
 #it also closes the window with the graph in it* **def** get\_user\_answer(self, frame6, answer, level, lesson, question, score):  
 plt.close()  
 user\_answer = self.user\_answer\_entry.get()  
 user\_answer = user\_answer.lower()  
 frame = frame6  
 check\_answer(frame, answer, user\_answer, level, lesson, question, score)  
  
*#the check\_answer class, checks the user answer against the real answer  
#also creates a message box based on whether they are correct of not  
#objetives 12, 13 and 14 are met in this function***class** check\_answer:  
  
 *#this class checks the users answer against the actual answer and saves the answer to the csv file  
 #this is the function that automatically runs when the class is called  
 #the function has the variables frame, answer, user\_answer, level, lesson, question and score as parameters* **def** \_\_init\_\_(self, frame, answer, user\_answer, level, lesson, question, score):  
 *#defines the variables as global* **global** user\_score, user\_row  
  
 *#closes the current window* frame.destroy()  
  
 *#coverts the answer into a string* answer = str(answer)  
  
 *#if answer is correct, add one to score  
 #create a message box saying correct* **if** answer == user\_answer:  
 score = score + 1  
 tk.messagebox.showinfo(**"answer"**,**"Correct"**)  
  
 *#if answer is incorrect create a message box saying incorrect* **else**:  
 tk.messagebox.showinfo(**"answer"**,**"Incorrect the correct answer was "**+answer)  
  
 *#if it is the fifth question asked then the score is saved in the student\_accounts.csv file under the correct heading* **if** question == 5:  
 database = csv.reader(open(**'student\_accounts.csv'**))  
 lines = [l **for** l **in** database]  
 question = 0  
 location = 0  
 **if** lesson == **"triangles"**:  
 location = 3  
 **elif** lesson == **"sequences"**:  
 location = 6  
 **elif** lesson == **"area"**:  
 location = 9  
 **elif** lesson == **"graph"**:  
 location = 12  
 **if** level == **"2"**:  
 location = location + 1  
 **elif** level == **"3"**:  
 location = location + 2  
 user\_score[location]= score  
 lines[user\_row-2][location+4] = str(score)  
  
 writer = csv.writer(open(**'student\_accounts.csv'**, **'w'**, newline=**''**))  
 writer.writerows(lines)  
 score = 0  
  
 *#if it isn't the fifth question then the lesson they are currently on is called* **else**:  
 **if** lesson == **"algebra"**:  
 algebra(level, question, score)  
 **elif** lesson == **"triangles"**:  
 triangles(level, question, score)  
 **elif** lesson == **"sequences"**:  
 sequences(level, question, score)  
 **elif** lesson == **"area"**:  
 area(level, question, score)  
 **elif** lesson == **"graph"**:  
 graph(level, question, score)  
  
*#the start\_page class, creates the start page and the functions related to all the buttons on the start page***class** start\_page():  
  
 *#creates the teacher page, creates all the buttons in tkinter for the teacher page  
 #this is the function that automatically runs when the class is called  
 #the function has the variable self as a parameters* **def** \_\_init\_\_(self):  
  
 *#generates and places the window for the start page* root = tk.Tk()  
 root.geometry(**"190x60"**)  
 root.title(**"Start Page"**)  
  
 *#assigns the variable password\_username* password\_username = **"0"** *#creates and places the label needed for the start page* tk.Label(root, text = **"Are you a teacher or a student?"**).place(x=10,y=0)  
  
 *#creates ad places the buttons needed for the start page* self.teacher\_button = tk.Button(root, text = **"Teacher"**, command = **lambda**:self.teacher(password\_username, root))  
 self.teacher\_button.place(x = 30, y = 30)  
 self.student\_button = tk.Button(root, text = **"Student"**, command = **lambda**:self.student(password\_username, root,))  
 self.student\_button.place(x = 90, y = 30)  
  
 *#the function teacher is called when the teacher button is pressed  
 #the function uses the variables self, password\_username and root as parameters  
 #the purpose of the function is to load the login page with the variable user = teacher* **def** teacher(self, password\_username, root):  
  
 *#closes the current window* root.destroy()  
  
 *#assigns the variable user* user = **"teacher"** *#calls the login\_page class* login\_page(password\_username, user)  
  
 *#the function student is called when the student button is pressed  
 #the function uses the variables self, password\_username and root as parameters  
 #the purpose of the function is to load the login page with the variable user = student* **def** student(self, password\_username, root):  
   
 *#closes the current window* root.destroy()  
  
 *#assigns the variable user* user = **"student"** *#calls the login\_page class* login\_page(password\_username, user )   
  
*#assigns the global variables*user\_row = 0  
user\_score = []  
  
*#calls forward the start\_page class*start\_page()

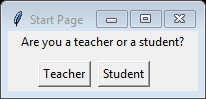
# System testing *(8 marks)*

## Test plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test num | Purpose of test | Test data | Expected outcome | Reference to test result |
| 1 | To test if the program will load the start page when the program is started. | Normal | The start page will load | Appendix 1 |
| 2 | To test if the login page will load when the teacher button is pressed | Normal | The login page will load | Appendix 2 |
| 3 | To test if the login page will load when the student button is pressed | Normal | The login page will load | Appendix 3 |
| 4 | To test if the window will close if the quit button is pressed (when the variable user == student) | Normal | The login page will close |  |
| 5 | To test if the create new user page will load when the create new user button is pressed(when the variable user == student) | Normal | The login page will close and the new user page will load | Appendix 4 |
| 6 | To test if the return button on the new user page works(when the variable user == student) | Normal | The new user page will close and the login page will load | Appendix 5 |
| 7 | To test if the program will create a new user when all the fields are filled in and a unique username is entered. (when the variable user == student) | Normal | A message box will appear with the message “account successfully made” and the new user page will close and the login page will load. Also the users details are added to student\_accounts.csv file | Appendix 6 |
| 8 | To test if the program will output an error message if the first name field is left blank (when the variable user == student) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 7 |
| 9 | To test if the program will output an error message if the last name field is left blank (when the variable user == student) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 8 |
| 10 | To test if the program will output an error message if the username field is left blank (when the variable user == student) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 9 |
| 11 | To test if the program will output an error message if the password and confirm password field is left blank (when the variable user == student) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 10 |
| 12 | To test if the program will output an error message if the password and confirm password do not match (when the variable user == student) | Erroneous | A message box will appear with the message “Passwords do not match” | Appendix 11 |
| 13 | To test if the program will output an error message if username is already being used (when the variable user == student) | Erroneous | A message box will appear with the message “That username is already taken, please try another” | Appendix 12 |
| 14 | To test if the program will create a new user when all the fields are filled in and a unique username is entered. (when the variable user == student) all the fields will contain 30 characters(the limit allowed) | Boundary | A message box will appear with the message “account successfully made” and the new user page will close and the login page will load. Also the users details are added to student\_accounts.csv file | Appendix 13 |
| 15 | To test if the program will output an error message if the first name field is larger than 30 characters (when the variable user == student) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 14 |
| 16 | To test if the program will output an error message if the last name field is larger than 30 characters (when the variable user == student) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 15 |
| 17 | To test if the program will output an error message if the username field is larger than 30 characters (when the variable user == student) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 16 |
| 18 | To test if the program will output an error message if the password and confirm password field larger than 30 characters (when the variable user == student) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 17 |
| 19 | To test if the program will log in a user when a correct username-password combination is entered (when the variable user == student) | Normal | The user will be successfully logged into the program and the main page will load displaying the users scores | Appendix 18 |
| 20 | To test if the program will log in a user when a correct username-password combination is entered (when the variable user == student) when the username-password combination are both 30 characters | Boundary | The user will be successfully logged into the program and the main page will load displaying the users scores | Appendix 19 |
| 21 | To test if the program will not log in a user when an incorrect username-password combination is entered but the username does match an account(when the variable user == student) | Erroneous | The login page will reload and an error message saying “wrong password” will appear | Appendix 20 |
| 22 | To test if the program will not log in a user when an incorrect username-password combination is entered (when the variable user == student) | Erroneous | The login page will reload and an error message saying “wrong username password” will appear” | Appendix 21 |
| 23 | To test whether the main page will refresh when the refresh button is pressed | Normal | The main page will refresh updating any new scores | Appendix 22 |
| 24 | To test whether the login page will load when the log out button is pressed | Normal | The main page will close and the login page will load | Appendix 23 |
| 25 | To test whether the question page will load when the algebra button is pressed. | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 24 |
| 26 | When the button level 1 of the algebra lesson is pressed the program will generate and display a question in the form ax=b | Normal | The question page will gain some extra widgets displaying the question | Appendix 25 |
| 27 | When the button level 2 of the algebra lesson is pressed the program will generate and display a question in the form ax + b = c or ax – b = c | Normal | The question page will gain some extra widgets displaying the question | Appendix 26 |
| 28 | When the button level 3 of the algebra lesson is pressed the program will generate and display a question in the form ax + b = cx + d, ax – b = cx – d, ax + b = cx – d or ax – b = cx + d |  | The question page will gain some extra widgets displaying the question | Appendix 27 |
| 29 | To test whether the question page will load when the triangle button is pressed. | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 28 |
| 30 | When the button level 1 of the triangle lesson is pressed the program will generate and display a question with two angles being randomly generated integers | Normal | The question page will gain some extra widgets displaying the question | Appendix 29 |
| 31 | When the button level 2 of the triangle lesson is pressed the program will generate and display a question with two sides of a triangle being randomly generated integers (sides a and b or sides b and c) | Normal | The question page will gain some extra widgets displaying the question | Appendix 30 |
| 32 | When the button level 3 of the triangle lesson is pressed the program will generate and display a question two sides and two angles being randomly generated integers | Normal | The question page will gain some extra widgets displaying the question | Appendix 31 |
| 33 | To test whether the question page will load when the sequences button is pressed. | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 32 |
| 34 | When the button level 1 of the sequences lesson is pressed the program will generate and display a question, in the form nth term = ax +/- b what is the c term | Normal | The question page will gain some extra widgets displaying the question | Appendix 33 |
| 35 | When the button level 2 of the sequences lesson is pressed the program will generate and display a question , in the form nth term = +/- b what is the c term | Normal | The question page will gain some extra widgets displaying the question | Appendix 34 |
| 36 | When the button level 3 of the sequences lesson is pressed the program will generate and display a question in the form [a,b,c,d,e] is a sequence what is the nth term in terms of x | Normal | The question page will gain some extra widgets displaying the question | Appendix 35 |
| 37 | To test whether the question page will load when the area button is pressed. | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 36 |
| 38 | When the button level 1 of the area lesson is pressed the program will generate and display a question with both side A and side B being randomly generated integers | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 37 |
| 39 | When the button level 2 of the area lesson is pressed the program will generate and display a question with radius being a randomly generated integer | Normal | The question page will gain some extra widgets displaying the question | Appendix 38 |
| 40 | When the button level 3 of the area lesson is pressed the program will generate and display a question with both side A, side B and height being randomly generated integers | Normal | The question page will gain some extra widgets displaying the question | Appendix 39 |
| 41 | To test whether the question page will load when the graph button is pressed. | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 40 |
| 42 | When the button level 1 of the graph lesson is pressed the program will generate a graph showing a linear function and asking what does y equal when x = a | Normal | The question page will load and be raised above the main page so that the main page is no longer visible | Appendix 41 |
| 43 | When the button level 2 of the graph lesson is pressed the program will generate a linear graph and asking is this graph ax +/- b | Normal | The question page will gain some extra widgets displaying the question | Appendix 42 |
| 44 | When the button level 3 of the graph lesson is pressed the program will generate a linear, quadratic or cubic graph and asking what type of graph it is | Normal | The question page will gain some extra widgets displaying the question | Appendix 43 |
| 45 | When the submit button on level 1 of the lesson algebra is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 44 |
| 46 | When the submit button on level 1 of the lesson algebra is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 45 |
| 47 | When the submit button is pressed on level 1 of the lesson algebra and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 46 |
| 48 | When the submit button is pressed on level 1 of the lesson algebra and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 47 |
| 49 | When the submit button on level 2 of the lesson algebra is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 48 |
| 50 | When the submit button on level 2 of the lesson algebra is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 49 |
| 51 | When the submit button is pressed on level 2 of the lesson algebra and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 50 |
| 52 | When the submit button is pressed on level 2 of the lesson algebra and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 51 |
| 53 | When the submit button on level 3 of the lesson algebra is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 52 |
| 54 | When the submit button on level 3 of the lesson algebra is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 53 |
| 55 | When the submit button is pressed on level 3 of the lesson algebra and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 54 |
| 56 | When the submit button is pressed on level 3 of the lesson algebra and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 55 |
| 57 | When the submit button on level 1 of the lesson triangles is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 56 |
| 58 | When the submit button on level 1 of the lesson triangles is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 57 |
| 59 | When the submit button is pressed on level 1 of the lesson triangles and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 58 |
| 60 | When the submit button is pressed on level 1 of the lesson triangles and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 59 |
| 61 | When the submit button on level 2 of the lesson triangles is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 60 |
| 62 | When the submit button on level 2 of the lesson triangles is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 61 |
| 63 | When the submit button is pressed on level 2 of the lesson triangles and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 62 |
| 64 | When the submit button is pressed on level 2 of the lesson triangles and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 63 |
| 65 | When the submit button on level 3 of the lesson triangles is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 64 |
| 66 | When the submit button on level 3 of the lesson triangles is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 65 |
| 67 | When the submit button is pressed on level 3 of the lesson triangles and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 66 |
| 68 | When the submit button is pressed on level 3 of the lesson triangles and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 67 |
| 69 | When the submit button on level 1 of the lesson sequences is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 68 |
| 70 | When the submit button on level 1 of the lesson sequences is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 69 |
| 71 | When the submit button is pressed on level 1 of the lesson sequences and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 70 |
| 72 | When the submit button is pressed on level 1 of the lesson sequences and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 71 |
| 73 | When the submit button on level 2 of the lesson sequences is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 72 |
| 74 | When the submit button on level 2 of the lesson sequences is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 73 |
| 75 | When the submit button is pressed on level 2 of the lesson sequences and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 74 |
| 76 | When the submit button is pressed on level 2 of the lesson sequences and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 75 |
| 77 | When the submit button on level 3 of the lesson sequences is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 76 |
| 78 | When the submit button on level 3 of the lesson sequences is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 77 |
| 79 | When the submit button is pressed on level 3 of the lesson sequences and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 78 |
| 80 | When the submit button is pressed on level 3 of the lesson sequences and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 79 |
| 81 | When the submit button on level 1 of the lesson area is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 80 |
| 82 | When the submit button on level 1 of the lesson area is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 81 |
| 83 | When the submit button is pressed on level 1 of the lesson area and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 82 |
| 84 | When the submit button is pressed on level 1 of the lesson area and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 83 |
| 85 | When the submit button on level 2 of the lesson area is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 84 |
| 86 | When the submit button on level 2 of the lesson area is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 85 |
| 87 | When the submit button is pressed on level 2 of the lesson area and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 86 |
| 88 | When the submit button is pressed on level 2 of the lesson area and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 87 |
| 89 | When the submit button on level 3 of the lesson area is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 88 |
| 90 | When the submit button on level 3 of the lesson area is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 89 |
| 91 | When the submit button is pressed on level 3 of the lesson area and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 90 |
| 92 | When the submit button is pressed on level 3 of the lesson area and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 91 |
| 93 | When the submit button on level 1 of the lesson graph is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 92 |
| 94 | When the submit button on level 1 of the lesson graph is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 93 |
| 95 | When the submit button is pressed on level 1 of the lesson graph and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 94 |
| 96 | When the submit button is pressed on level 1 of the lesson graph and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 95 |
| 97 | When the submit button on level 2 of the lesson graph is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 96 |
| 98 | When the submit button on level 2 of the lesson graph is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 97 |
| 99 | When the submit button is pressed on level 2 of the lesson graph and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 98 |
| 100 | When the submit button is pressed on level 2 of the lesson graph and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 99 |
| 101 | When the submit button on level 3 of the lesson graph is pressed and the answer is correct a message box will appear saying correct | Normal | A new question will load and a message box saying “correct’’ will appear | Appendix 100 |
| 102 | When the submit button on level 3 of the lesson graph is pressed and the answer is incorrect a message box will appear saying incorrect | Normal | A new question will load and a message box saying “incorrect the correct answer was (the variable answer)” will appear | Appendix 101 |
| 103 | When the submit button is pressed on level 3 of the lesson graph and it isn’t the fifth question a new question will load | Normal | A new question will load replacing the previous question | Appendix 102 |
| 104 | When the submit button is pressed on level 3 of the lesson graph and it is the fifth question a new question won’t load and the score will be updated in the student\_accounts.csv file | Normal | The main page will appear | Appendix 103 |
| 105 | When level 1 of the lesson triangle is chosen the correct image will load | Normal | The question page will appear along with an image | Appendix 104 |
| 106 | When level 2 of the lesson triangle is chosen the correct image will load | Normal | The question page will appear along with an image | Appendix 105 |
| 107 | When level 3 of the lesson triangle is chosen the correct image will load | Normal | The question page will appear along with an image | Appendix 106 |
| 108 | When level 1 of the lesson area is chosen the correct image will load | Normal | The question page will appear along with an image | Appendix 107 |
| 109 | When level 2 of the lesson area is chosen the correct image will load | Normal | The question page will appear along with an image | Appendix 108 |
| 110 | When level 3 of the lesson area is chosen the correct image will load | Normal | The question page will appear along with an image | Appendix 109 |
| 111 | To test whether the window containing the graph for level 1 of the lesson graph is destroyed when the submit button is pressed. | Normal | The window containing the graph related to the question is closed |  |
| 112 | To test whether the window containing the graph for level 2 of the lesson graph is destroyed when the submit button is pressed. | Normal | The window containing the graph related to the question is closed |  |
| 113 | To test whether the window containing the graph for level 3 of the lesson graph is destroyed when the submit button is pressed. | Normal | The window containing the graph related to the question is closed |  |
| 114 | To test if the window will close if the quit button is pressed (when the variable user == teacher) | Normal | The login page will close |  |
| 115 | To test if the create new user page will load when the create new user button is pressed(when the variable user == teacher) | Normal | The login page will close and the new user page will load | Appendix 110 |
| 116 | To test if the return button on the new user page works(when the variable user == teacher) | Normal | The new user page will close and the login page will load | Appendix 111 |
| 117 | To test if the program will create a new user when all the fields are filled in and a unique username is entered. (when the variable user == teacher) | Normal | A message box will appear with the message “account successfully made” and the new user page will close and the login page will load. Also the users details are added to student\_accounts.csv file | Appendix 112 |
| 118 | To test if the program will output an error message if the first name field is left blank (when the variable user == teacher) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 113 |
| 119 | To test if the program will output an error message if the last name field is left blank (when the variable user == teacher) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 114 |
| 120 | To test if the program will output an error message if the username field is left blank (when the variable user == teacher) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 115 |
| 121 | To test if the program will output an error message if the password and confirm password field is left blank (when the variable user == teacher) | Erroneous | A message box will appear with the message “some of the fields have been left blank” | Appendix 116 |
| 122 | To test if the program will output an error message if the password and confirm password do not match (when the variable user == teacher) | Erroneous | A message box will appear with the message “Passwords do not match” | Appendix 117 |
| 123 | To test if the program will output an error message if username is already being used (when the variable user == teacher) | Erroneous | A message box will appear with the message “That username is already taken, please try another” | Appendix 118 |
| 124 | To test if the program will create a new user when all the fields are filled in and a unique username is entered. (when the variable user == teacher) all the fields will contain 30 characters(the limit allowed) | Boundary | A message box will appear with the message “account successfully made” and the new user page will close and the login page will load. Also the users details are added to student\_accounts.csv file | Appendix 119 |
| 125 | To test if the program will output an error message if the first name field is larger than 30 characters (when the variable user == teacher) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 120 |
| 126 | To test if the program will output an error message if the last name field is larger than 30 characters (when the variable user == teacher) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 121 |
| 127 | To test if the program will output an error message if the username field is larger than 30 characters (when the variable user == teacher) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 122 |
| 128 | To test if the program will output an error message if the password and confirm password field larger than 30 characters (when the variable user == teacher) | Erroneous | A message box will appear with the message “all fields have a 30 character limit” | Appendix 123 |
| 129 | To test if the program will log in a user when a correct username-password combination is entered (when the variable user == teacher) | Normal | The user will be successfully logged into the program and the teacher page will load | Appendix 124 |
| 130 | To test if the program will log in a user when a correct username-password combination is entered (when the variable user == teacher) when the username-password combination are both 30 characters | Boundary | The user will be successfully logged into the program and the teacher page will load | Appendix 125 |
| 131 | To test if the program will not log in a user when an incorrect username-password combination is entered but the username does match an account(when the variable user == teacher) | Erroneous | The login page will reload and an error message saying “wrong password” will appear | Appendix 126 |
| 132 | To test if the program will not log in a user when an incorrect username-password combination is entered (when the variable user == teacher) | Erroneous | The login page will reload and an error message saying “wrong username password” will appear” | Appendix 127 |
| 133 | To test if the program will load a student’s score if the correct first and last name are entered | Normal | The students score’s will be displayed on the teacher page | Appendix 128 |
| 134 | To test if the program will display an error message if a the inputted first and last name are not store in the student\_accounts.csv file | Erroneous | An error message saying “a student by that name doesn’t exist” will appear” | Appendix 129 |
| 134 | To test if the program will load a student’s score if the correct first and last name are entered (but both are of 30 characters) | Boundary | The students score’s will be displayed on the teacher page | Appendix 130 |
| 135 | To test if the program will log in a user when a correct username-password combination is entered (when the variable user == student) when there is over 150 rows in the csv file and the row required is at the end | Boundary | The user will be successfully logged into the program and the teacher page will load | Appendix 131 |
| 136 | To test if the program will log in a user when a correct username-password combination is entered (when the variable user == teacher) when there is over 150 rows in the csv file and the row required is at the end | Boundary | The user will be successfully logged into the program and the main page will load | Appendix 132 |
| 137 | To test if the program will load a student’s score if the correct first and last name are entered when there is over 150 rows in the csv file and the row required is at the end | Boundary | The students score’s will be displayed on the teacher page | Appendix 133 |

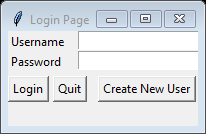
## Annotated screenshots of test results

Appendix 1:



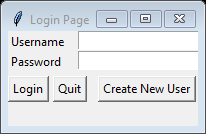
A screenshot of the start page that loads when the program starts

Appendix 2:



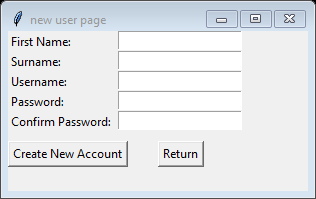
A screenshot of the login page that loads when the teacher button is pressed

Appendix 3:



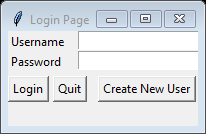
A screenshot of the login page that loads when the student button is pressed

Appendix 4:



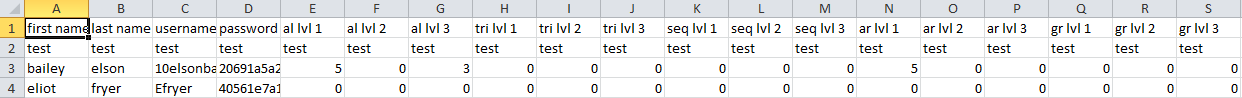
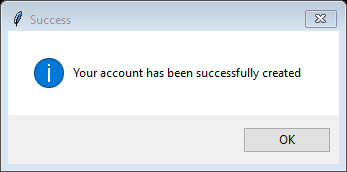
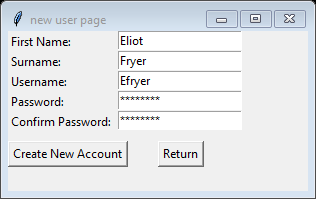
A screenshot of the new user page that is loaded when the create new user button is pressed

Appendix 5:



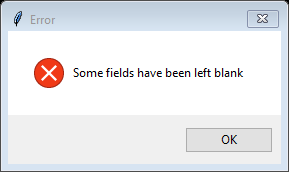
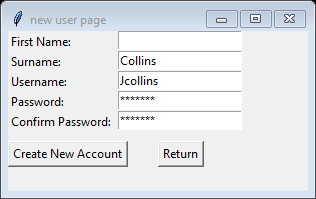
A screenshot of the login page that is loaded when the return button is pressed

Appendix 6:



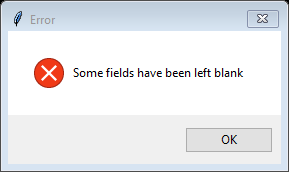
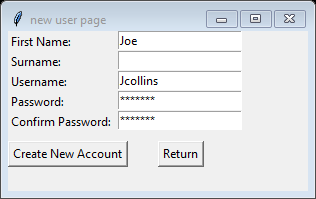
A screenshot of the filled out new user page, the pop up message and the csv file where the details are stored when the create new user button is pressed

Appendix 7:



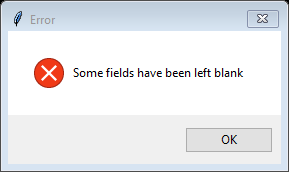
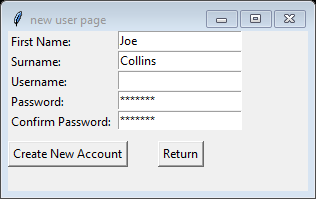
A screenshot of the new user page with the empty field and the pop up error message

Appendix 8:



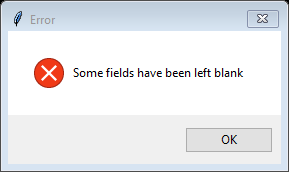
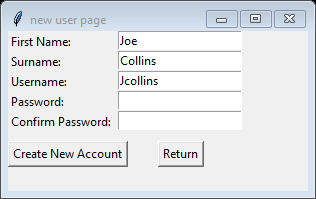
A screenshot of the new user page with the empty field and the pop up error message

Appendix 9:



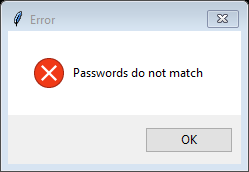
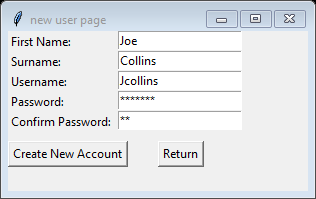
A screenshot of the new user page with the empty field and the pop up error message

Appendix 10:



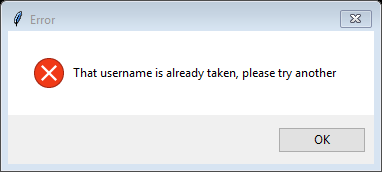
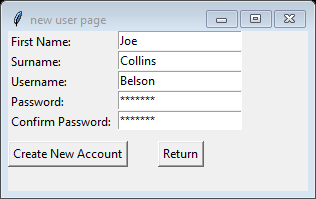
A screenshot of the new user page with the empty field and the pop up error message

Appendix 11:



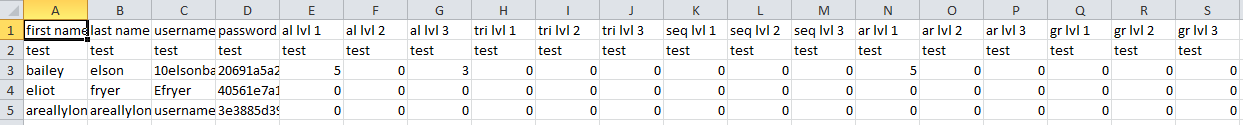
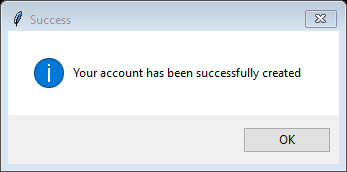
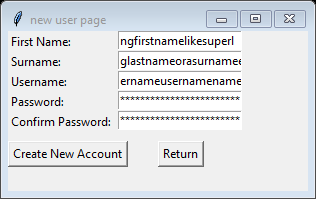
A screenshot of the new user page with the fields password and confirm passwords not matching and the pop up error message

Appendix 12:



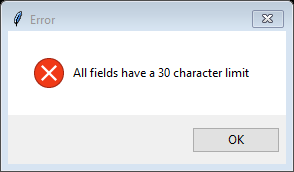
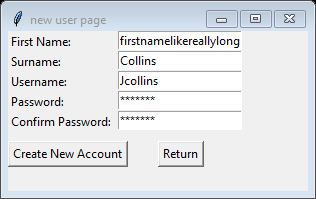
A screenshot of the new user page with a non-unique username and the pop up error message

Appendix 13:



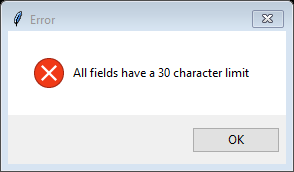
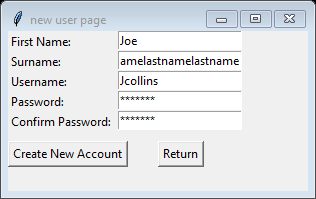
A screenshot of the filled out new user page with all fields containing 30 characters, the pop up message and the csv file where the details are stored when the create new user button is pressed

Appendix 14:



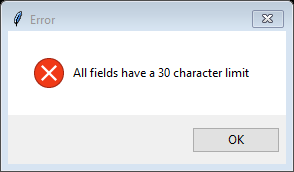
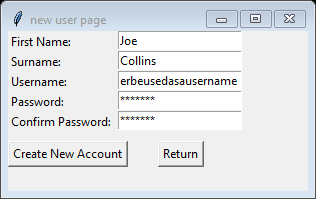
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 15:



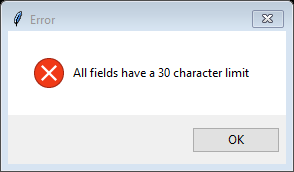
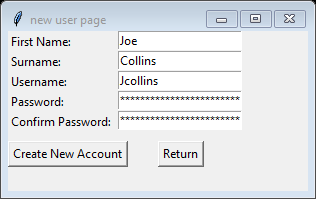
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 16:



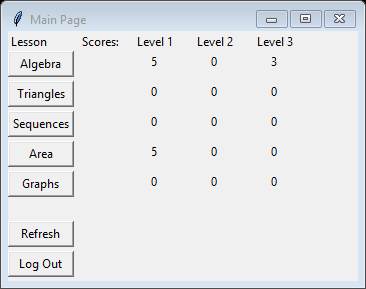
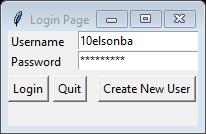
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 17:



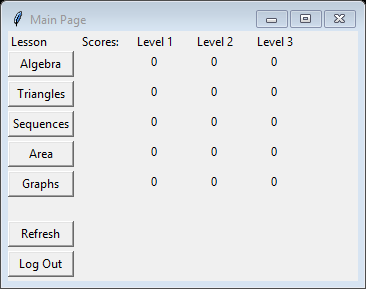
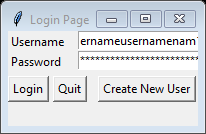
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 18:



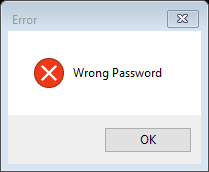
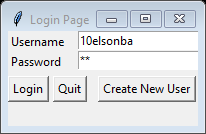
A screenshot a login page with a correct username-password combination and the main page that loads when the user logs in

Appendix 19:



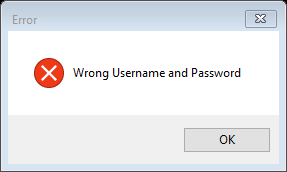
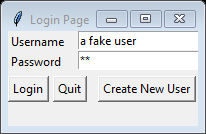
A screenshot a login page with a correct username-password combination (both of 30 characters) and the main page that loads when the user logs in

Appendix 20:



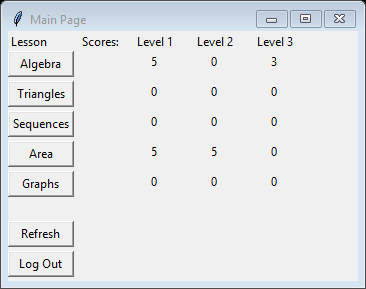
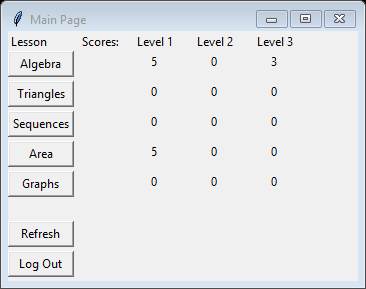
A screenshot a login page with an incorrect username-password combination and pop up error message

Appendix 21:



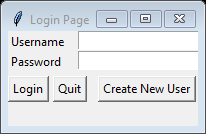
A screenshot a login page with an incorrect username-password combination and pop up error message

Appendix 22:



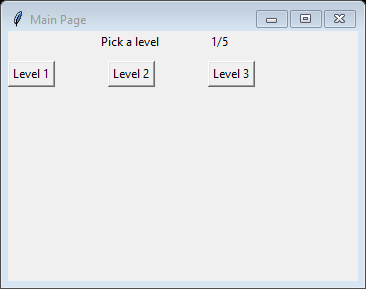
A screenshot of the main page before and after the refresh button has been pressed. (The area level 2 score has been updated)

Appendix 23:



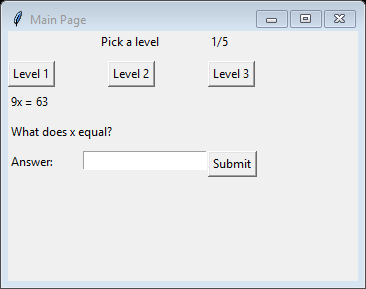
A screenshot of the login page that is loaded when the log out button is pressed

Appendix 24:



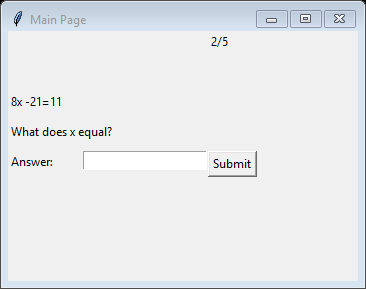
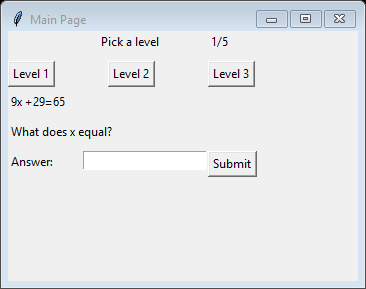
A screenshot of the question page that is loaded after the algebra button has been pressed

Appendix 25:



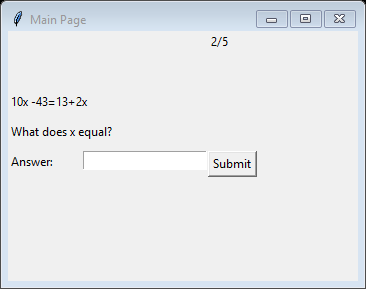
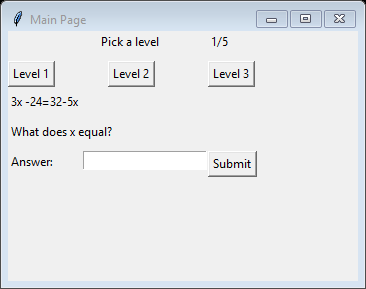
A screenshot of the question page that is loaded after the level 1 algebra button has been pressed

Appendix 26:

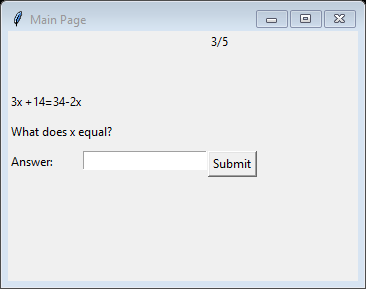
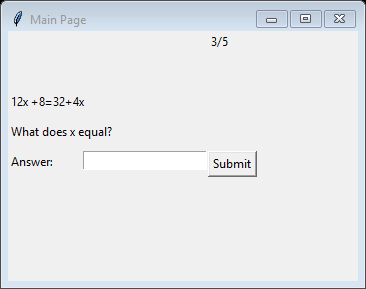


A screenshot of the question page that is loaded after the level 2 algebra button has been pressed

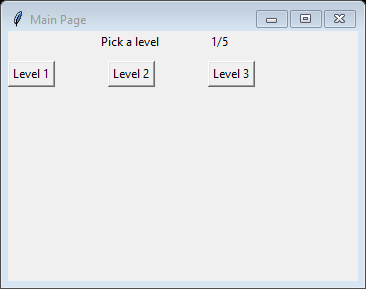
Appendix 27:



A screenshot of the question page that is loaded after the level 3 algebra button has been pressed

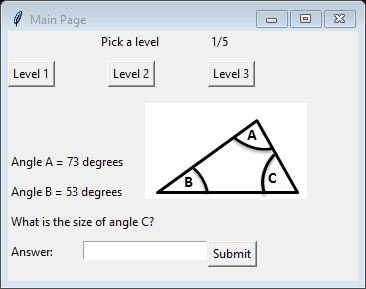


Appendix 28:



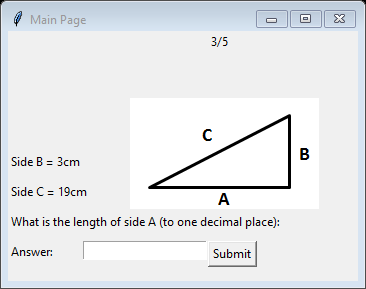
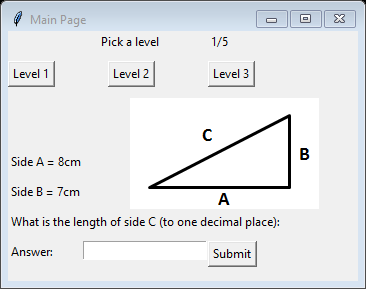
A screenshot of the question page that is loaded after the triangle button has been pressed

Appendix 29:



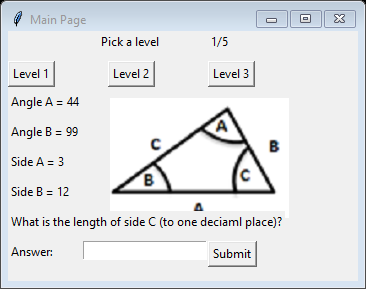
A screenshot of the question page that is loaded after the triangle level 1 button has been pressed

Appendix 30:



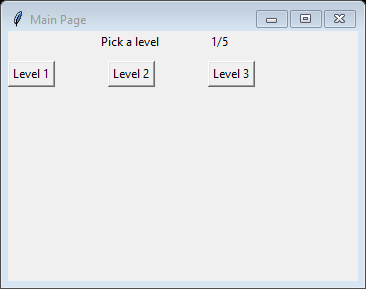
A screenshot of the question page that is loaded after the triangle level 2 button has been pressed

Appendix 31:



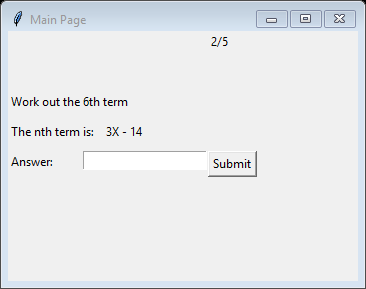
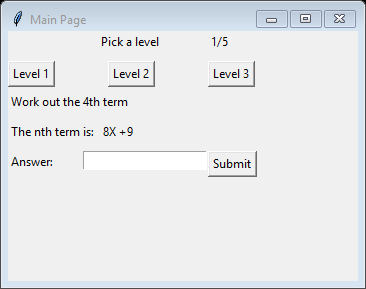
A screenshot of the question page that is loaded after the triangle level 3 button has been pressed

Appendix 32:



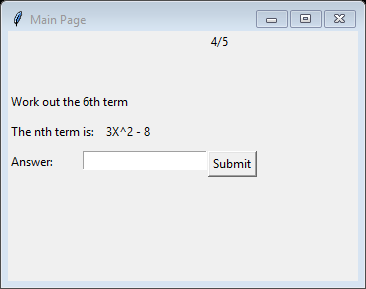
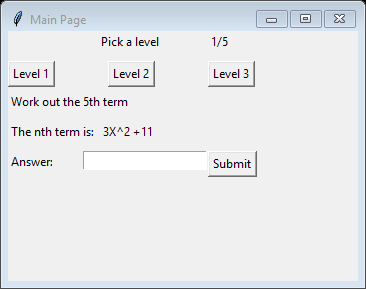
A screenshot of the question page that is loaded after the sequence button has been pressed

Appendix 33:



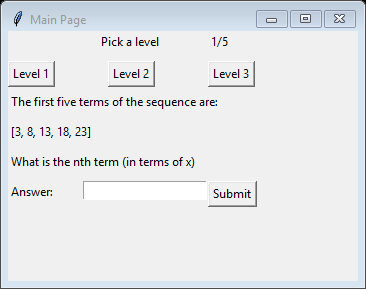
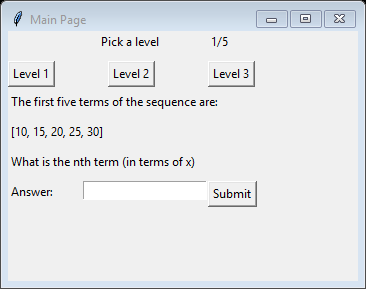
A screenshot of the question page that is loaded after the sequence level 1 button has been pressed

Appendix 34:



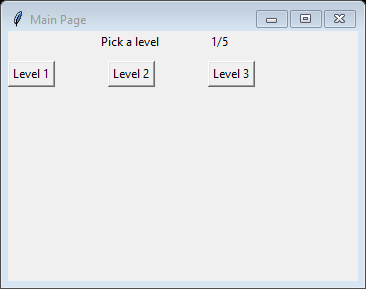
A screenshot of the question page that is loaded after the sequence level 2 button has been pressed

Appendix 35:



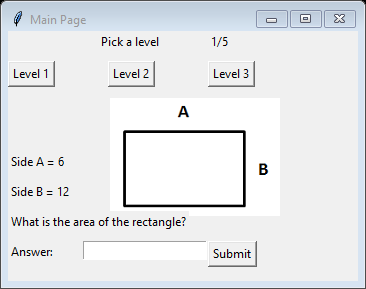
A screenshot of the question page that is loaded after the sequence level 3 button has been pressed

Appendix 36:



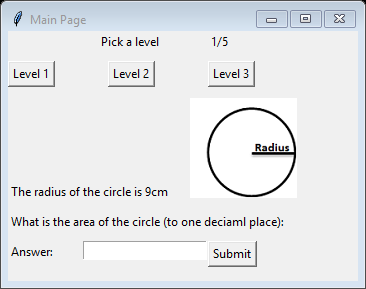
A screenshot of the question page that is loaded after the area button has been pressed

Appendix 37:



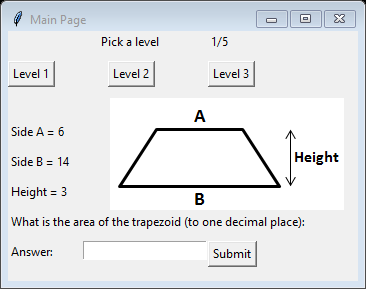
A screenshot of the question page that is loaded after the area level 1 button has been pressed

Appendix 38:



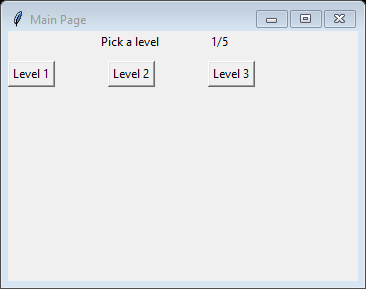
A screenshot of the question page that is loaded after the area level 2 button has been pressed

Appendix 39:



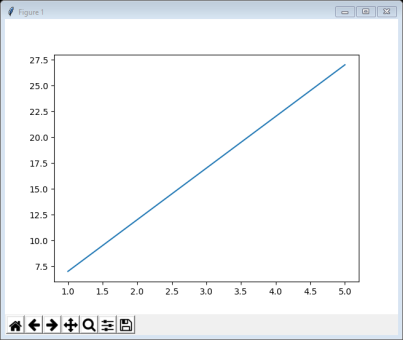
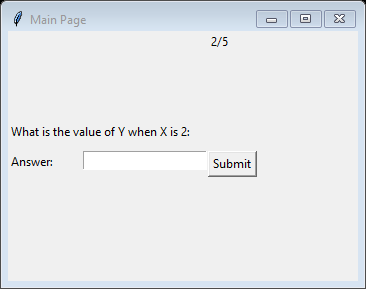
A screenshot of the question page that is loaded after the area level 3 button has been pressed

Appendix 40:



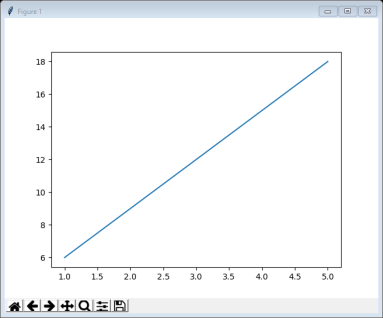
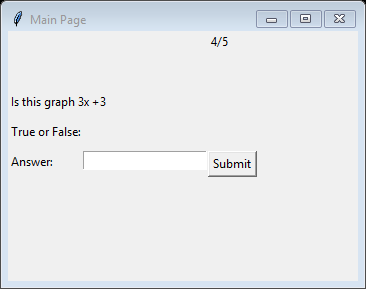
A screenshot of the question page that is loaded after the graph button has been pressed

Appendix 41:

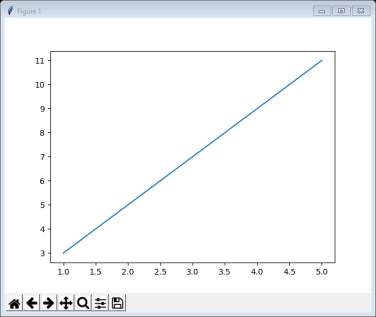
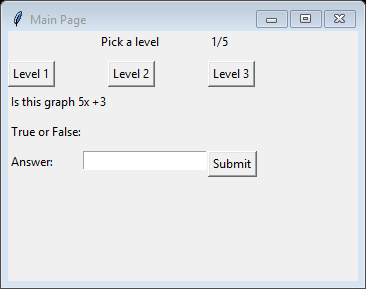


A screenshot of the question page that is loaded after the graph level 1 button has been pressed and the graph window created

Appendix 42:

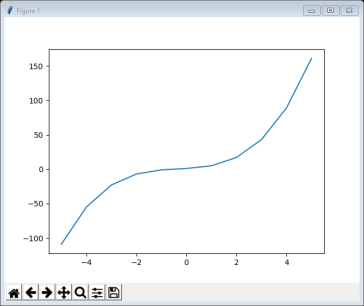
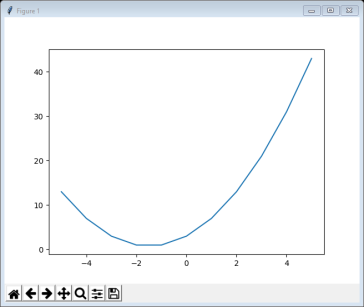
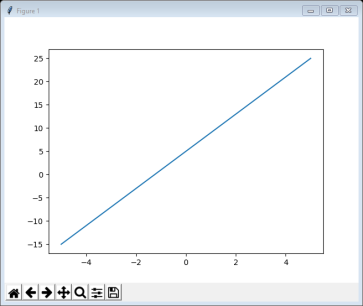
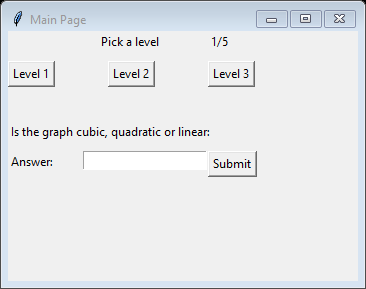


A screenshot of the question page that is loaded after the graph level 2 button has been pressed and the graph window created

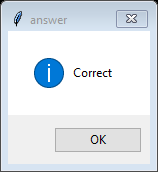
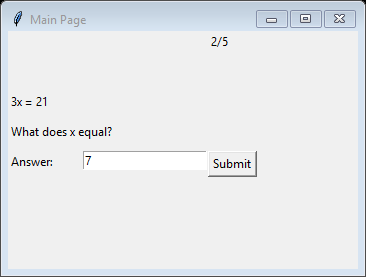


A screenshot of the question page that is loaded after the graph level 3 button has been pressed and the graph windows created

Appendix 43:

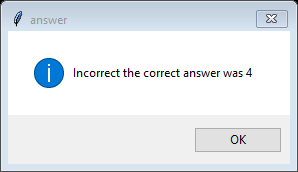
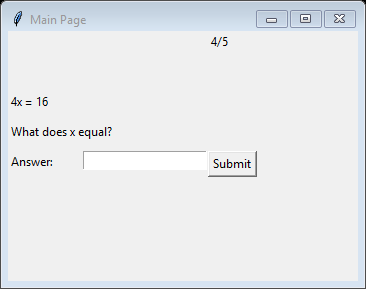


Appendix 44:



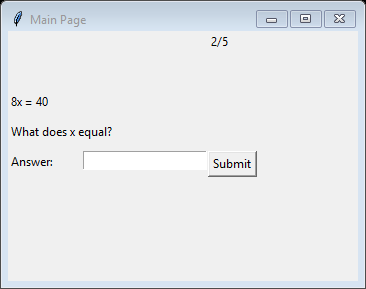
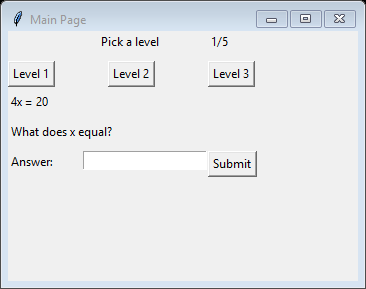
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 45:



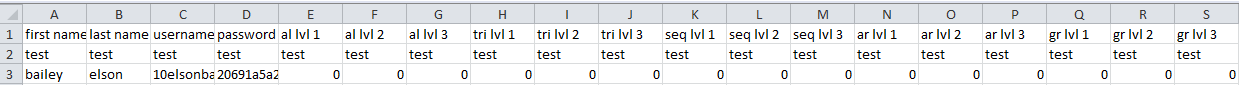
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

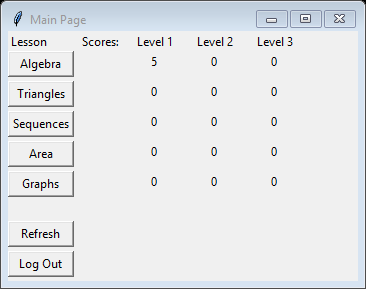
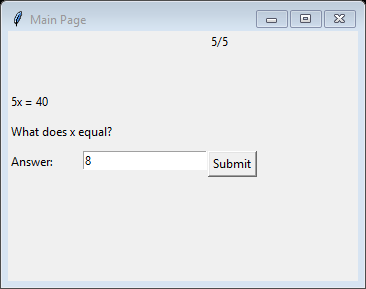
Appendix 46:



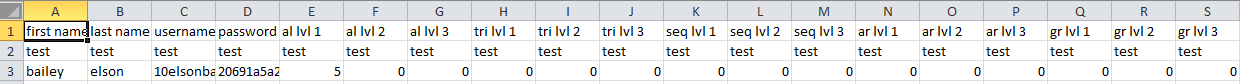
A screenshot of the question page loading another question after the submit button is pressed

Appendix 47:

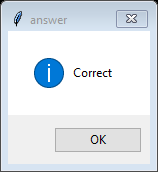
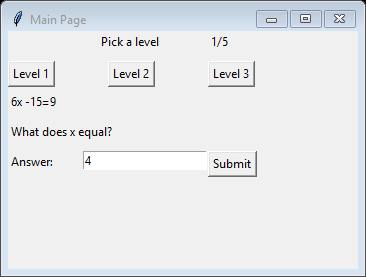




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated



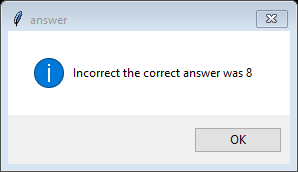
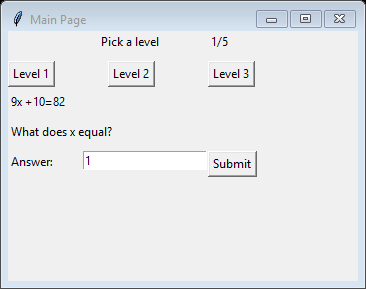
Appendix 48:



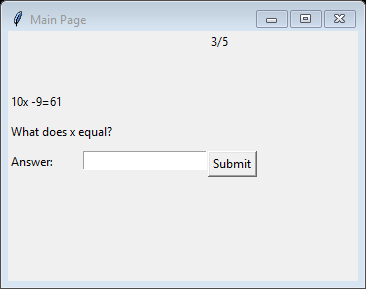
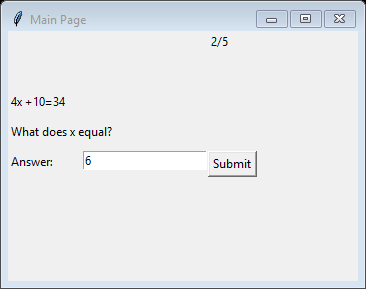
A screenshot of the question page and the pop up message created when the correct answer is inputted

A screenshot of the question page and the pop up message created when the incorrect answer is inputted

Appendix 49:

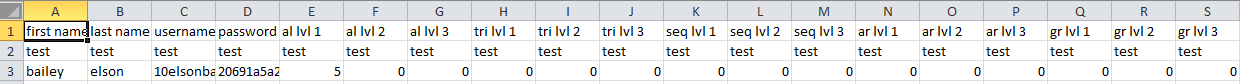


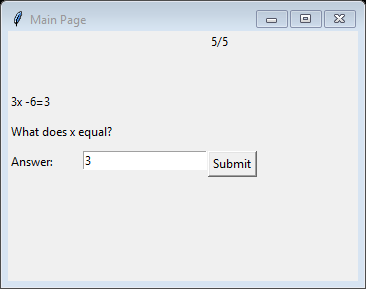
Appendix 50:

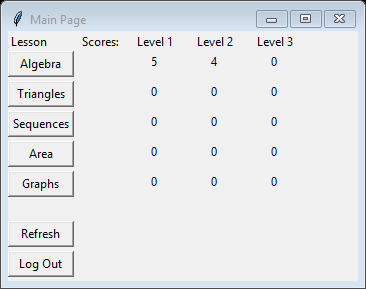
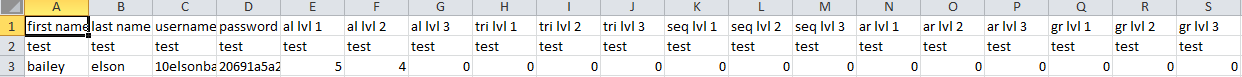


A screenshot of the question page loading another question after the submit button is pressed

Appendix 51:



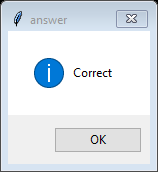
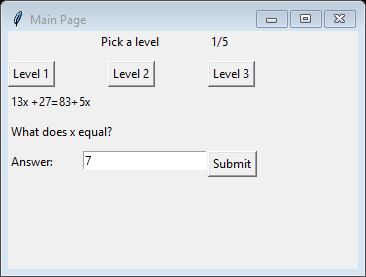


A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

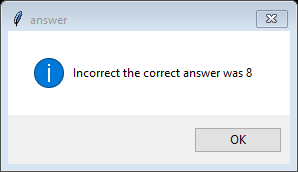
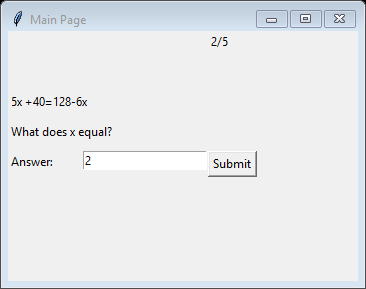
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

Appendix 52:

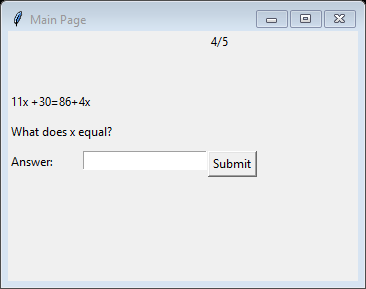
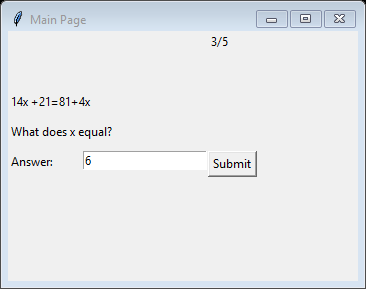


A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 53:

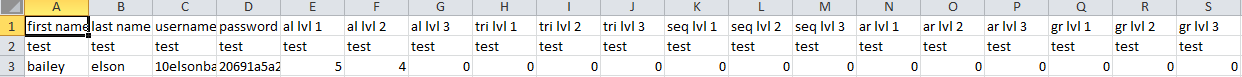


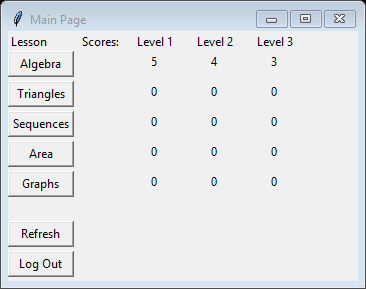
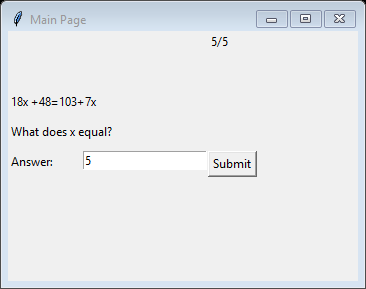
Appendix 54:



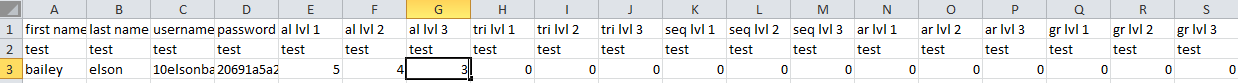
A screenshot of the question page loading another question after the submit button is pressed

Appendix 55:

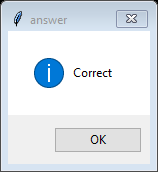
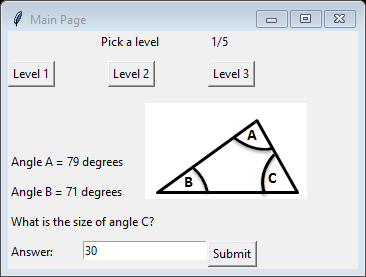




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

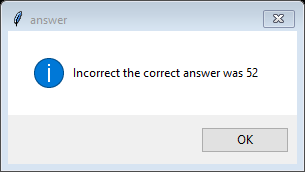
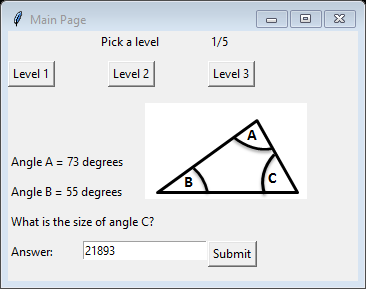


Appendix 56:



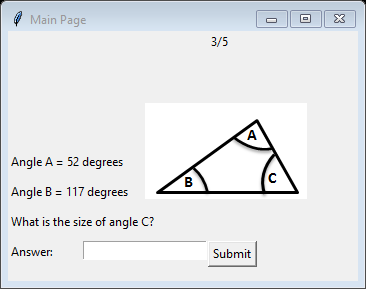
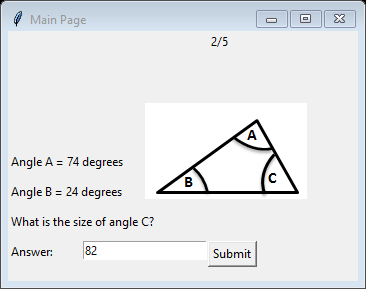
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 57:

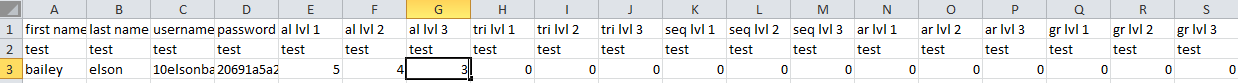


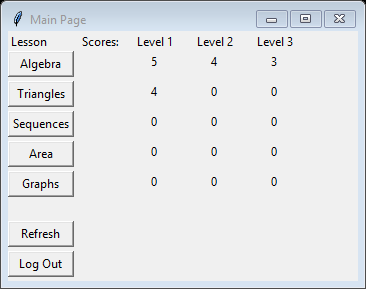
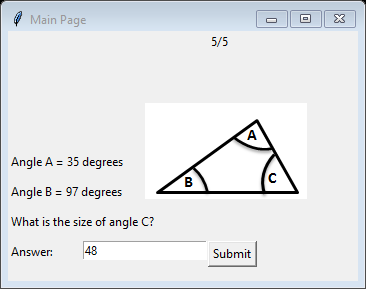
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

Appendix 58:

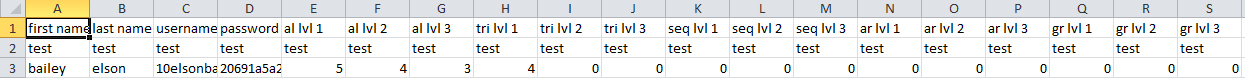


A screenshot of the question page loading another question after the submit button is pressed

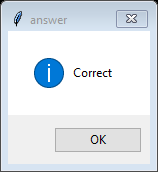
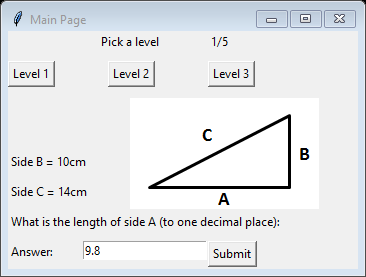
Appendix 59:



A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

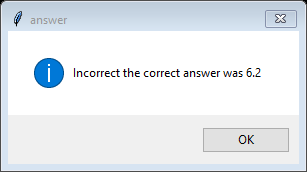
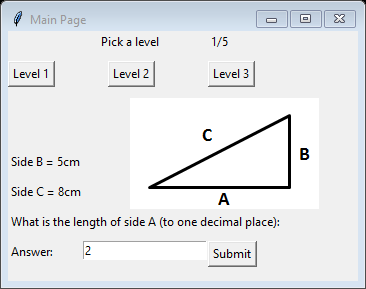


Appendix 60:



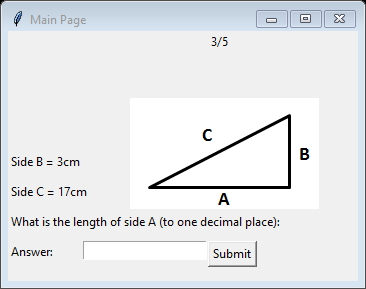
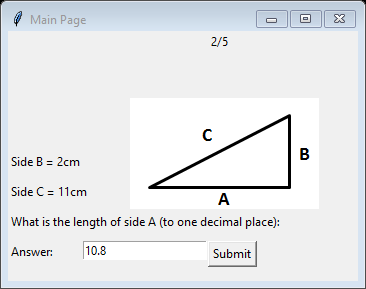
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 61:



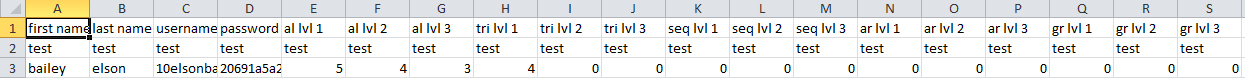
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

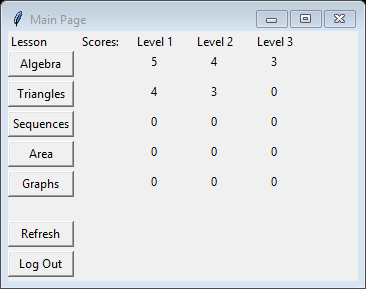
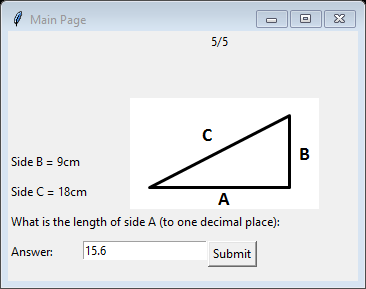
Appendix 62:



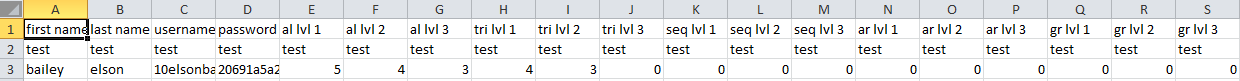
A screenshot of the question page loading another question after the submit button is pressed

Appendix 63:

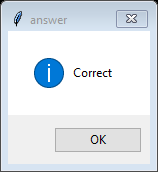
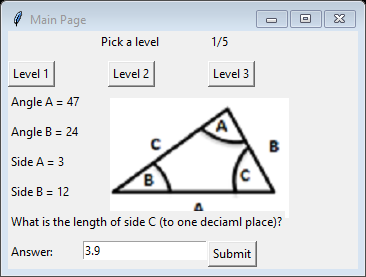




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

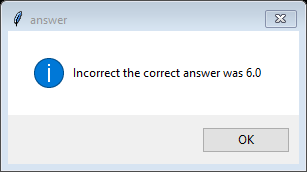
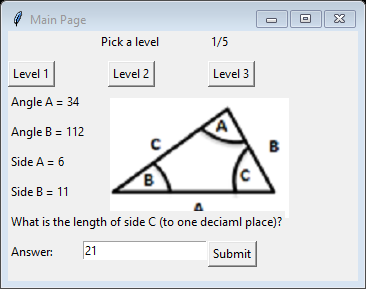


Appendix 64:



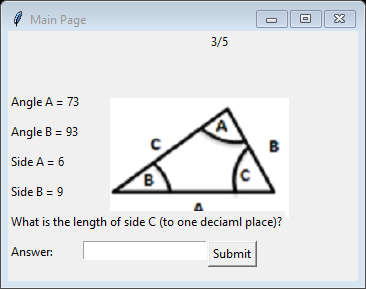
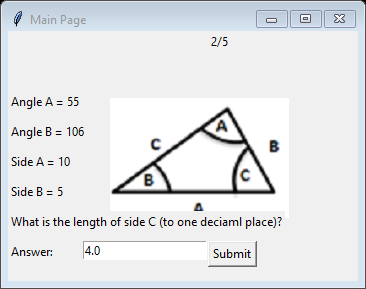
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 65:



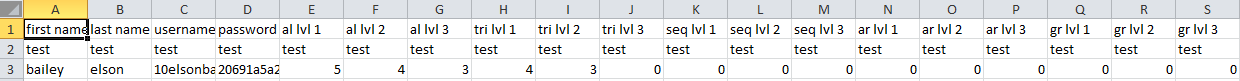
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

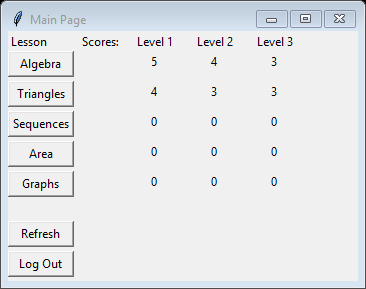
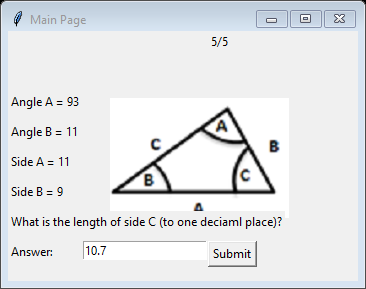
Appendix 66:



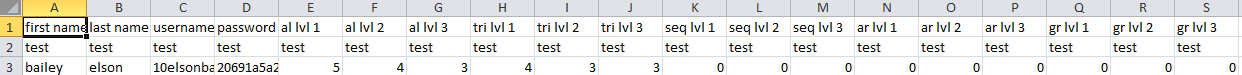
A screenshot of the question page loading another question after the submit button is pressed

Appendix 67:

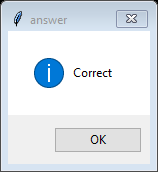
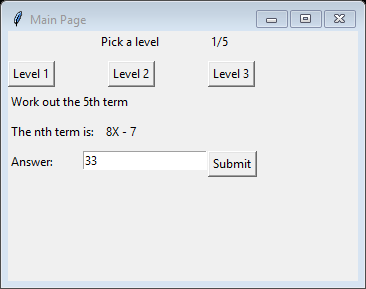




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

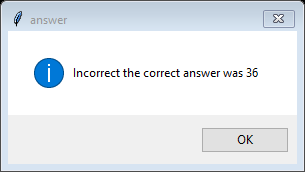
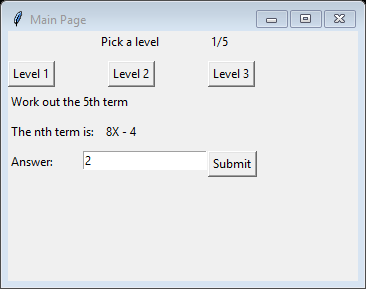


Appendix 68:



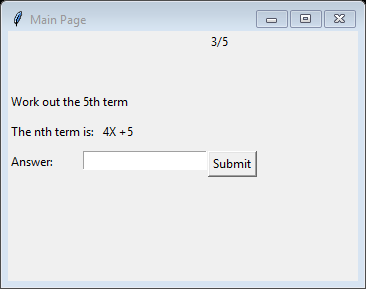
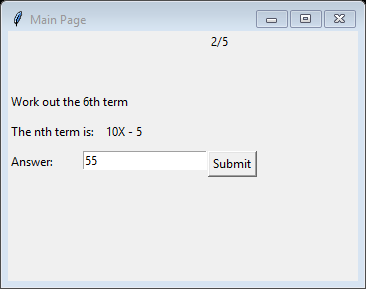
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 69:



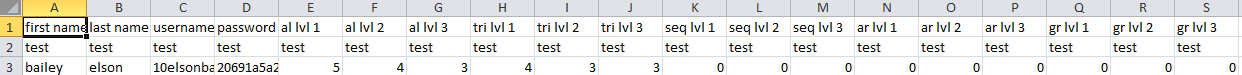
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

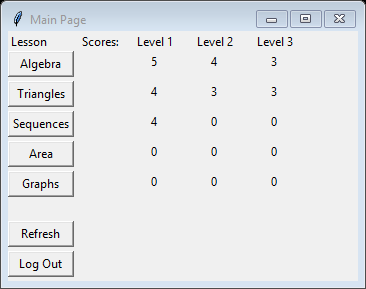
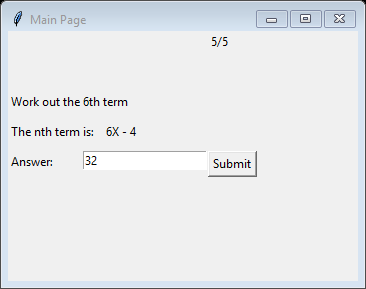
Appendix 70:



A screenshot of the question page loading another question after the submit button is pressed

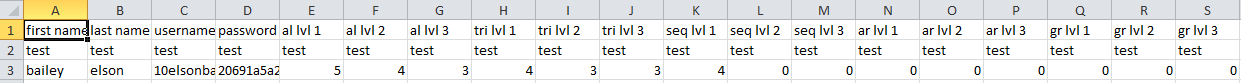
Appendix 71:



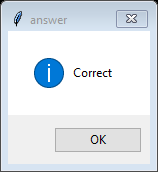
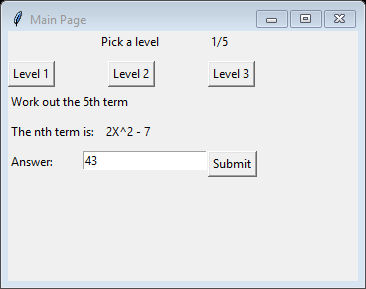


A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

A screenshot of the question page and the pop up message created when the incorrect answer is inputted

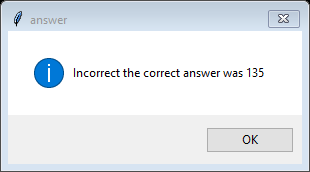
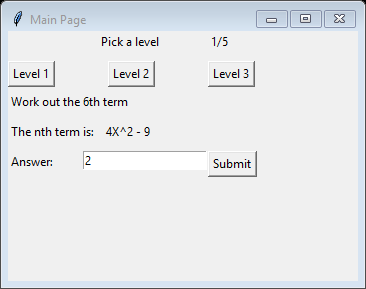


Appendix 72:

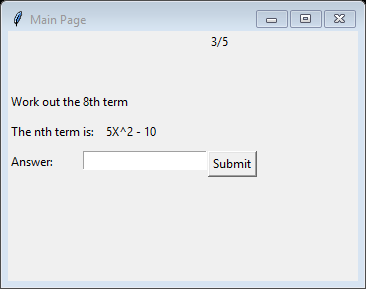
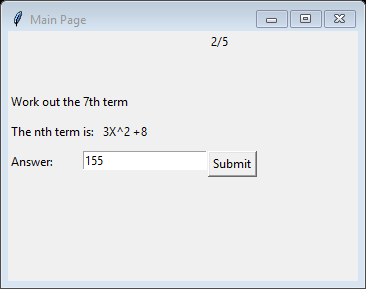


A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 73:

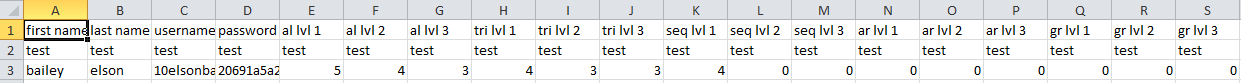


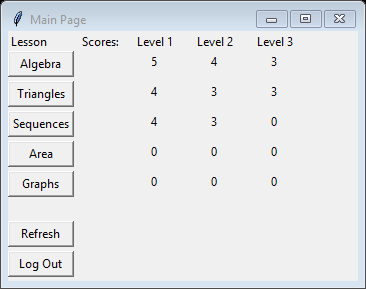
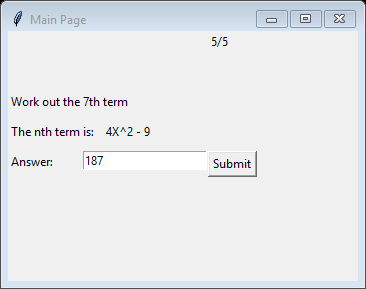
Appendix 74:



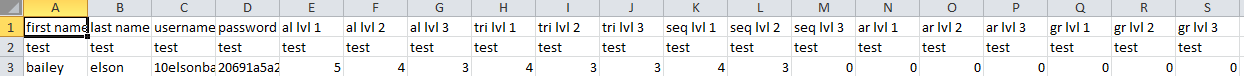
A screenshot of the question page loading another question after the submit button is pressed

Appendix 75:

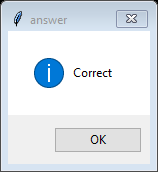
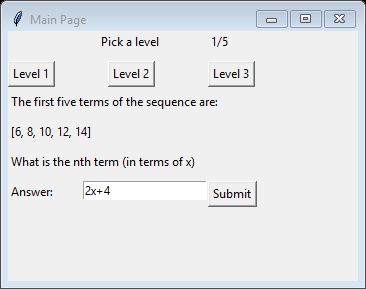




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

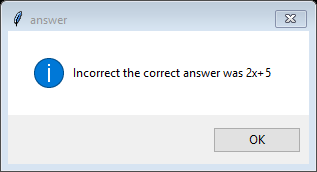
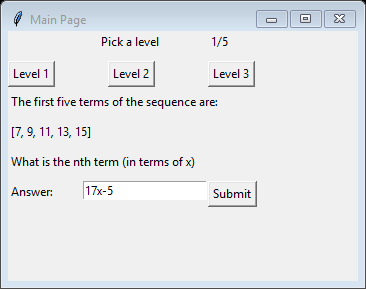


Appendix 76:



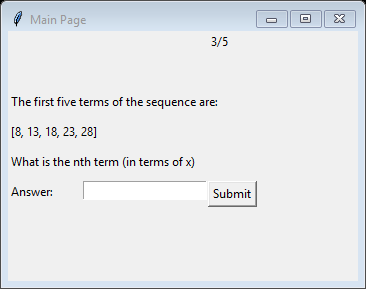
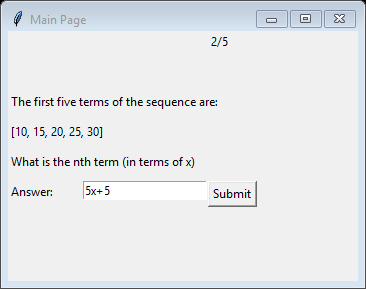
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 77:



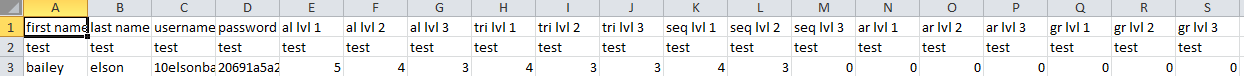
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

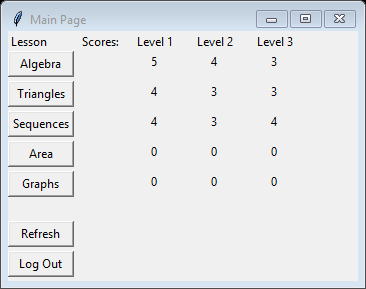
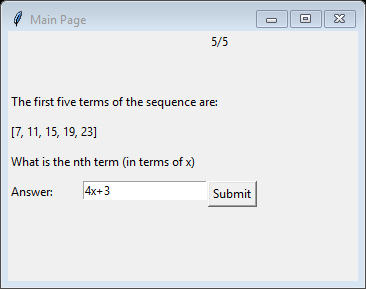
Appendix 78:



A screenshot of the question page loading another question after the submit button is pressed

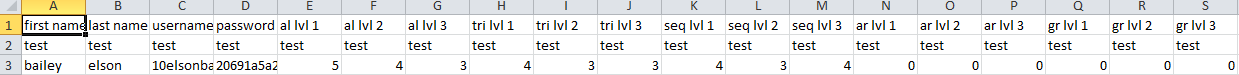
Appendix 79:



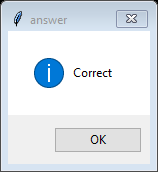
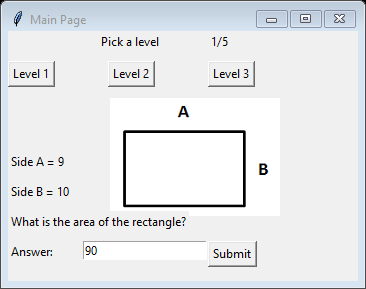


A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

A screenshot of the question page loading another question after the submit button is pressed

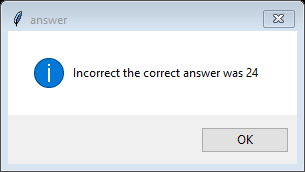
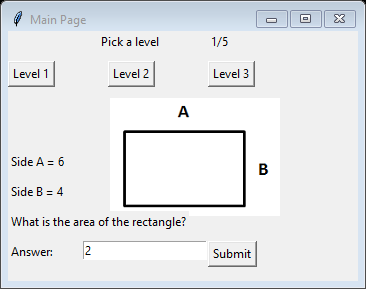


Appendix 80:



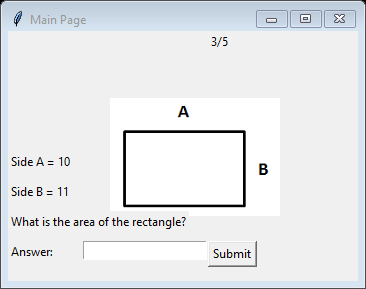
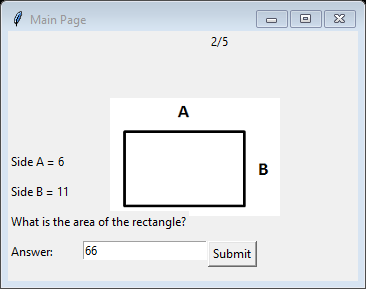
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 81:

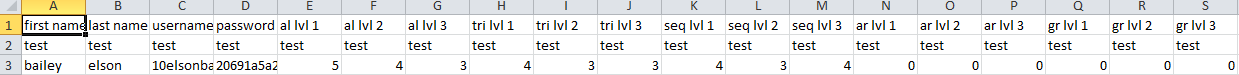


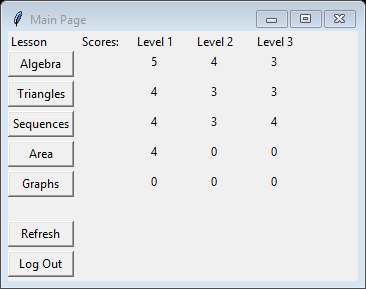
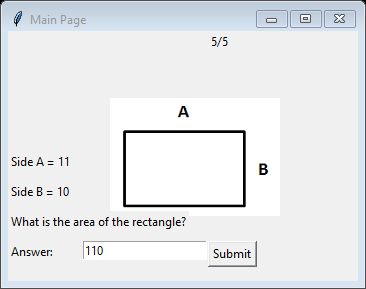
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

Appendix82:

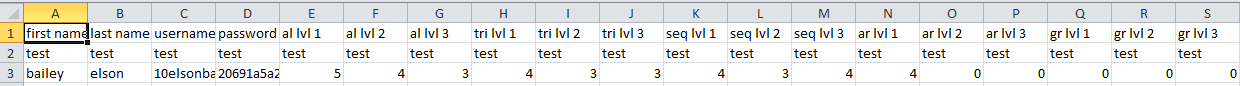


Appendix 83:

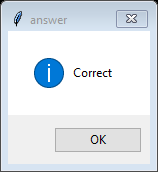
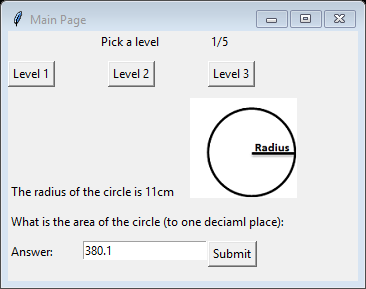




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

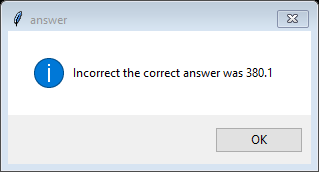
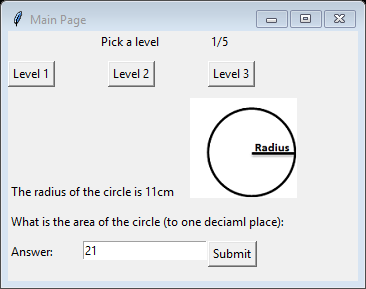


Appendix 84:



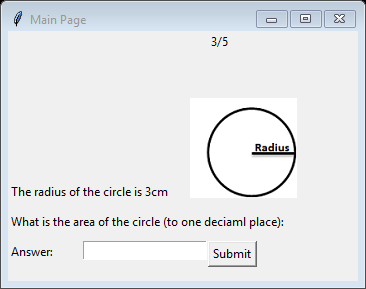
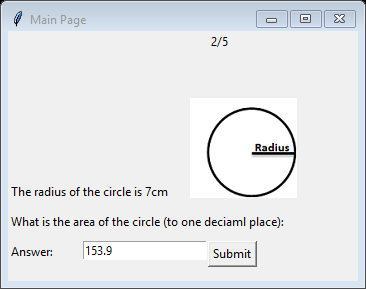
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 85:



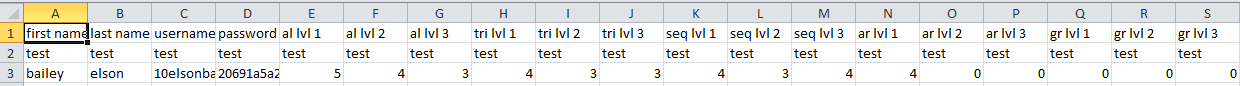
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

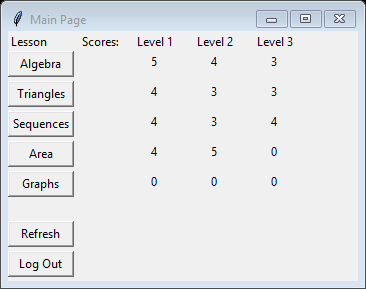
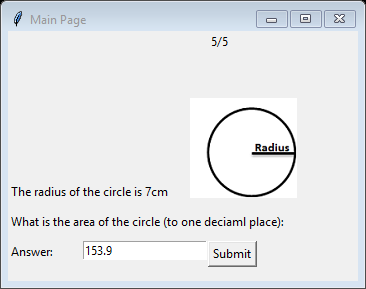
Appendix 86:



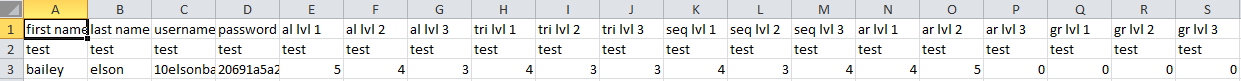
A screenshot of the question page loading another question after the submit button is pressed

Appendix 87:

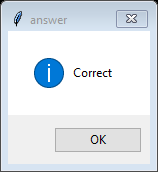
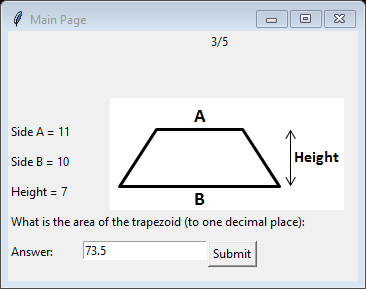




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

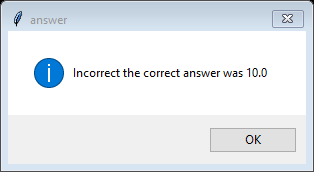
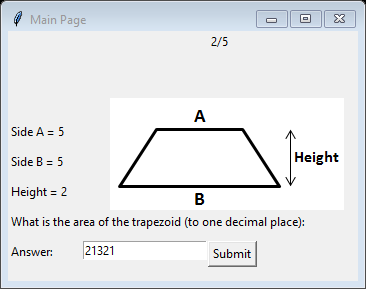


Appendix 88:



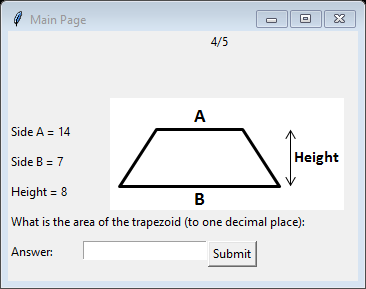
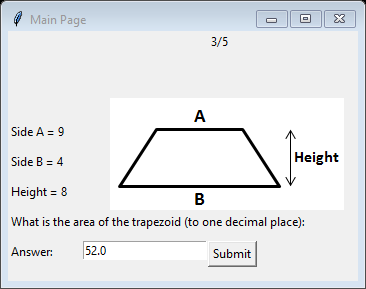
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 89:



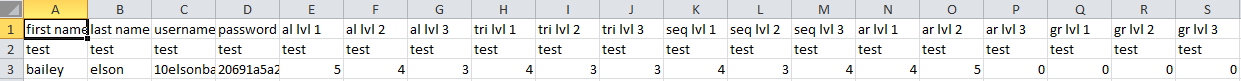
A screenshot of the question page and the pop up message created when the incorrect answer is inputted

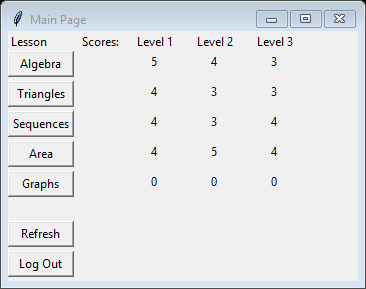
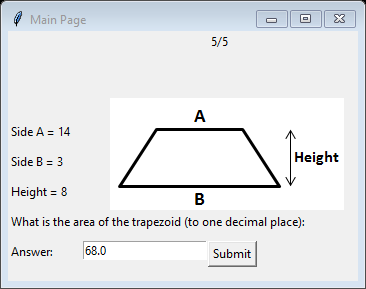
Appendix 90:



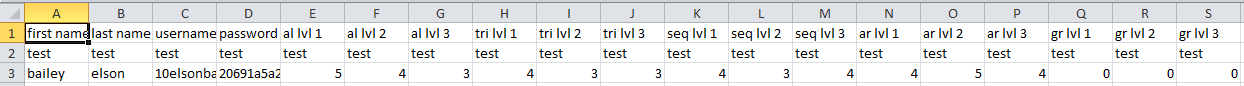
A screenshot of the question page loading another question after the submit button is pressed

Appendix 91:

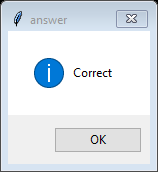
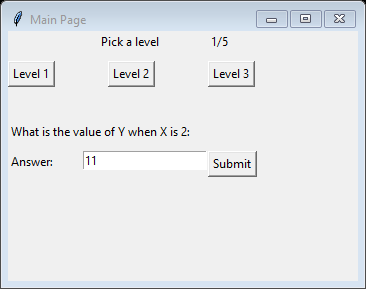
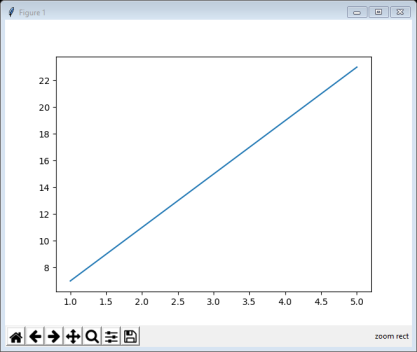




A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated



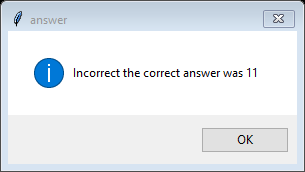
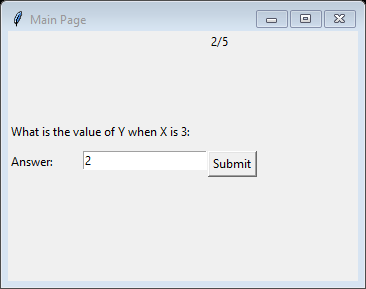
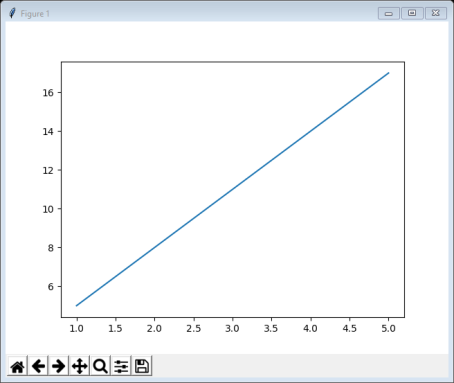
Appendix 92:



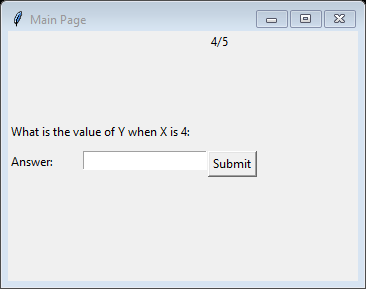
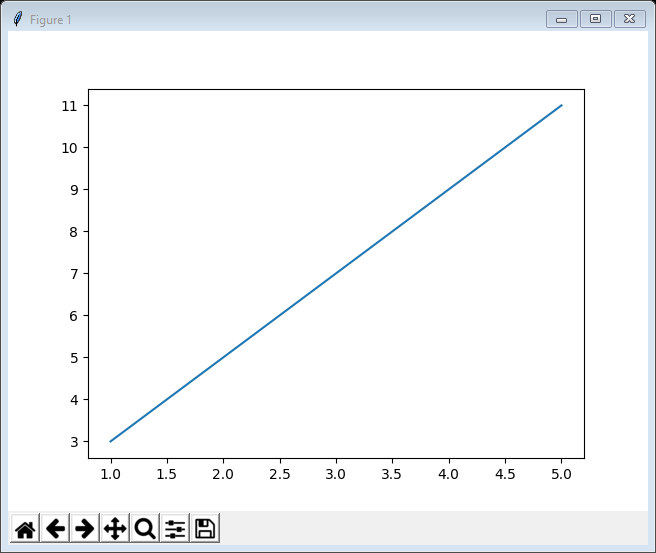
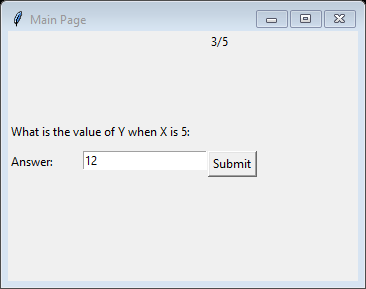
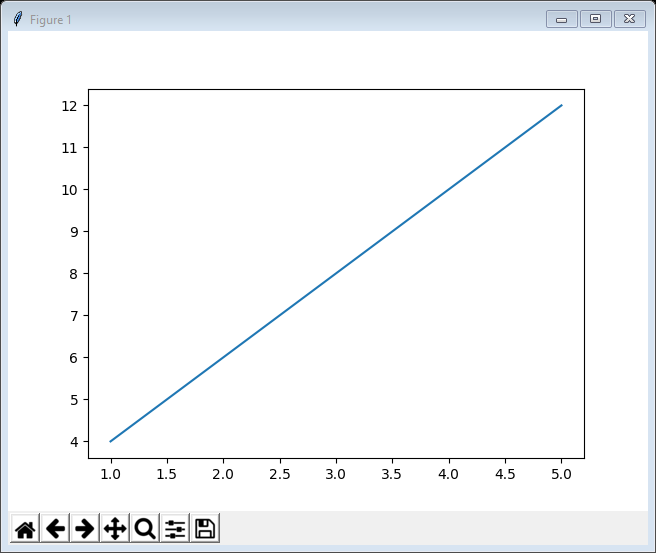
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 93:

A screenshot of the question page and the pop up message created when the incorrect answer is inputted



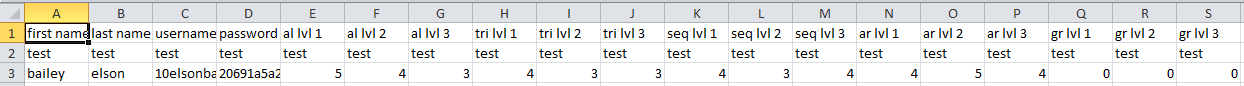
Appendix 94:

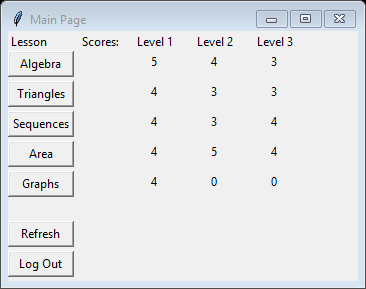
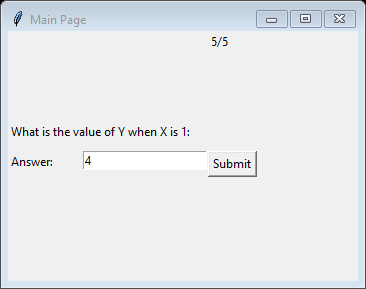
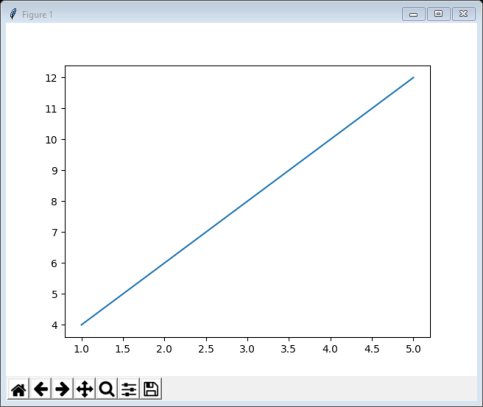


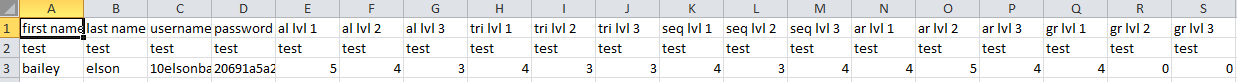
A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

A screenshot of the question page loading another question after the submit button is pressed

Appendix 95:

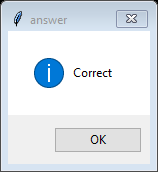
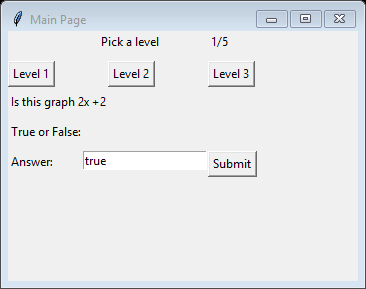
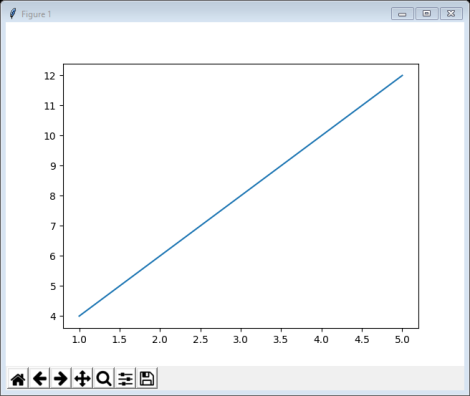






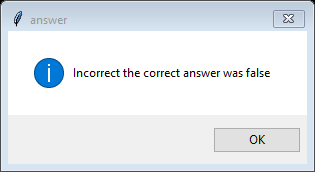
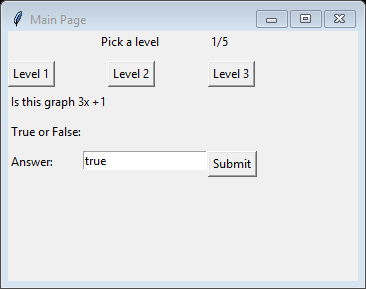
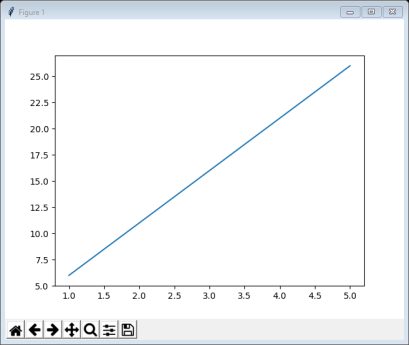
Appendix 96:

A screenshot of the question page and the pop up message created when the correct answer is inputted

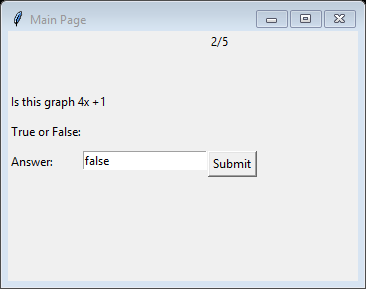
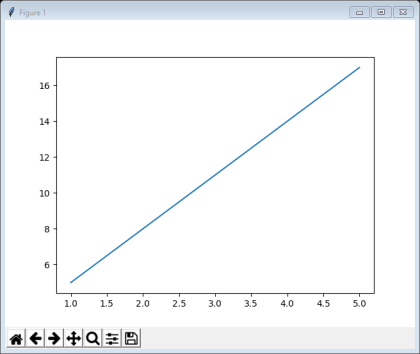


Appendix 97:

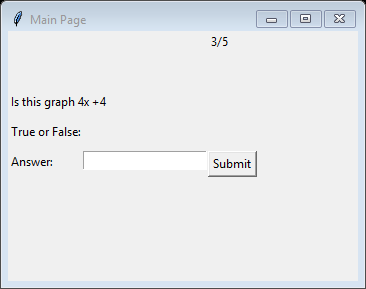
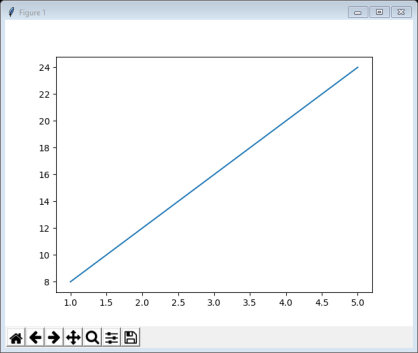
A screenshot of the question page and the pop up message created when the incorrect answer is inputted



Appendix 98:

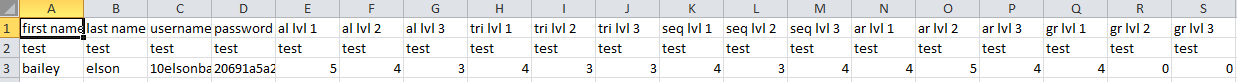


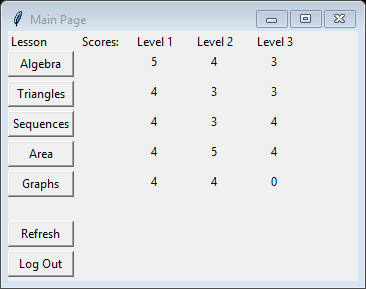
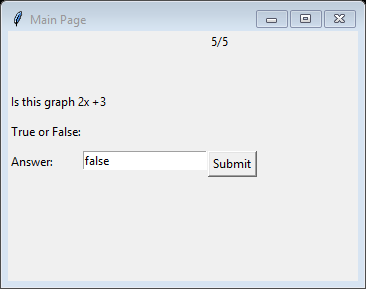
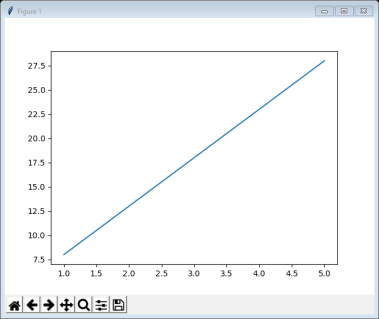
A screenshot of the question page loading another question after the submit button is pressed

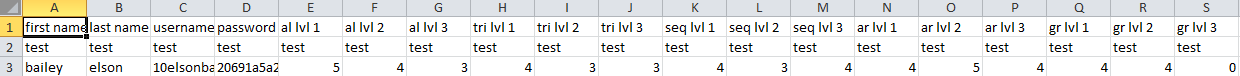


A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

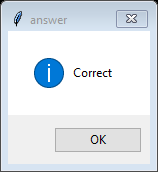
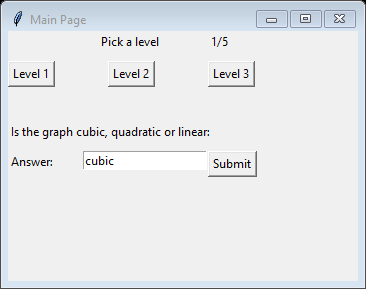
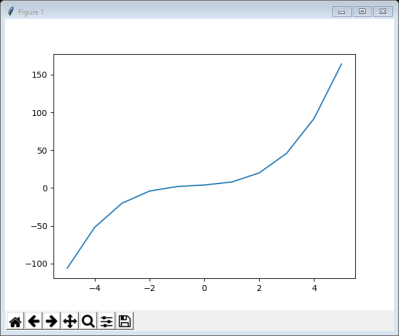
Appendix 99:







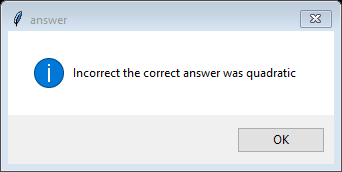
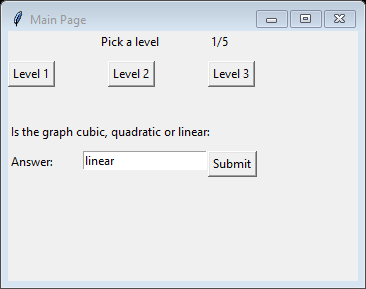
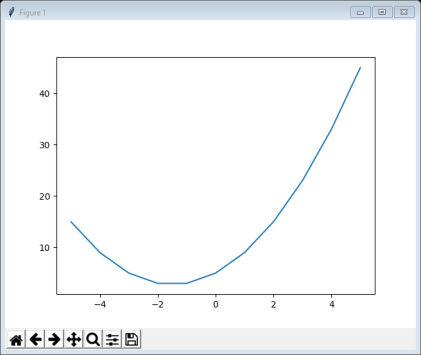
Appendix 100:



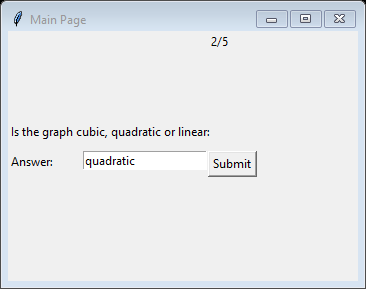
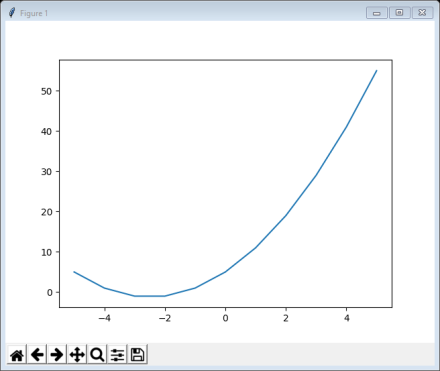
A screenshot of the question page and the pop up message created when the correct answer is inputted

Appendix 101:

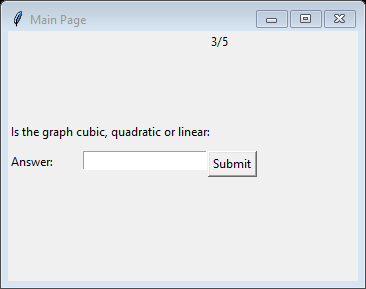
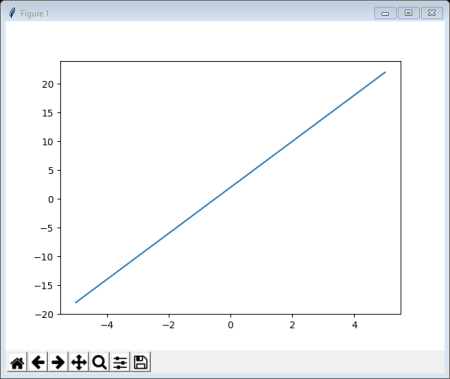
A screenshot of the question page and the pop up message created when the incorrect answer is inputted



Appendix 102:

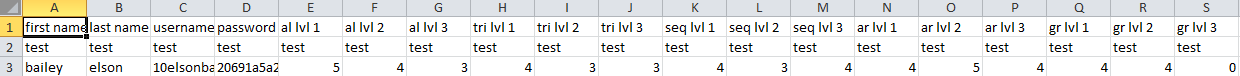


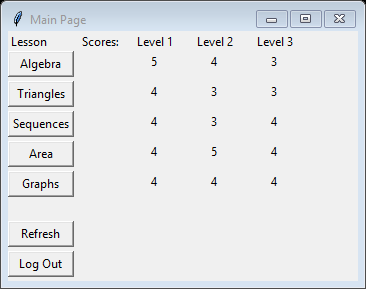
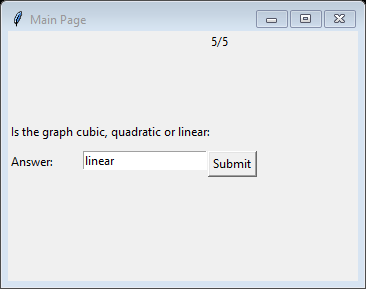
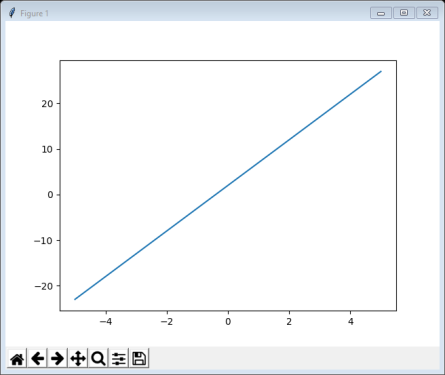
A screenshot of the question page loading another question after the submit button is pressed

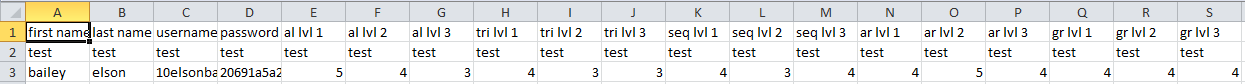


A screenshot of the question page on the final question, the main page loaded after the submit button is pressed and the csv file before and after its been updated

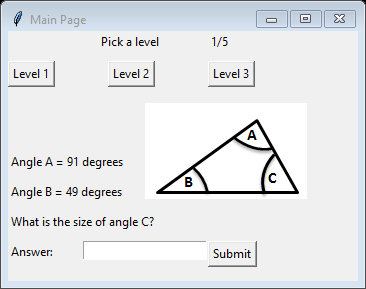
Appendix 103:





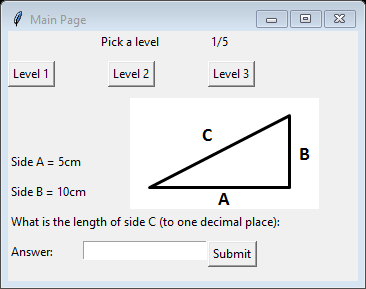


Appendix 104:



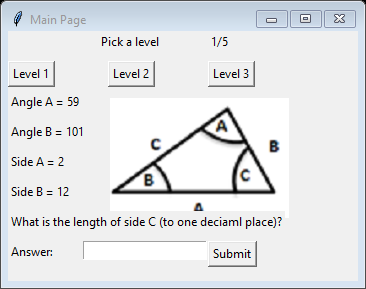
A screenshot of the question page along with the correct image loaded on the question page

Appendix 105:



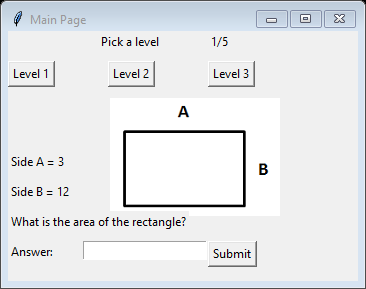
A screenshot of the question page along with the correct image loaded on the question page

Appendix 106:



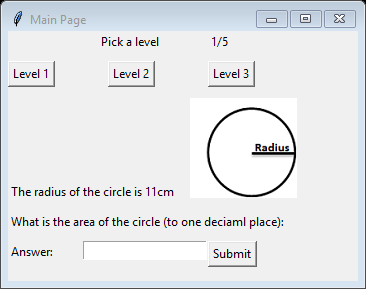
A screenshot of the question page along with the correct image loaded on the question page

Appendix 107:



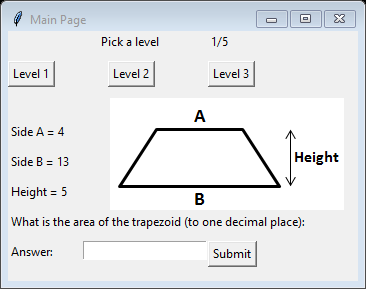
A screenshot of the question page along with the correct image loaded on the question page

Appendix 108:



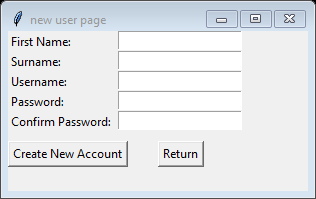
A screenshot of the question page along with the correct image loaded on the question page

Appendix 109:



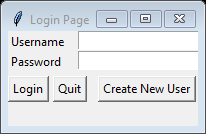
A screenshot of the question page along with the correct image loaded on the question page

Appendix 110:



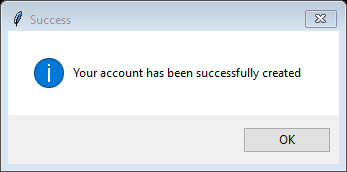
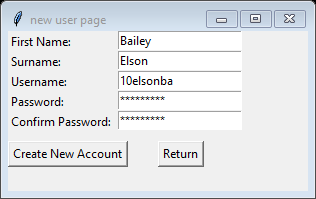
A screenshot of the new user page that is loaded when the create new user button is pressed

Appendix 111:

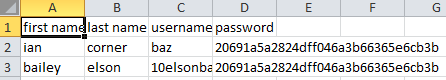


A screenshot of the login page that is loaded when the return button is pressed

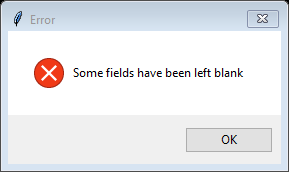
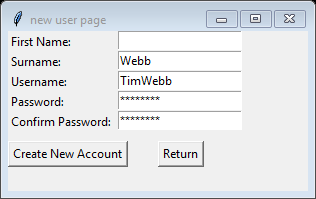
Appendix 112:



A screenshot of the filled out new user page, the pop up message and the csv file where the details are stored when the create new user button is pressed

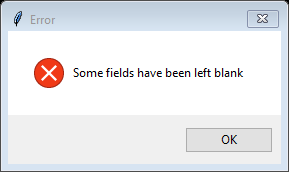
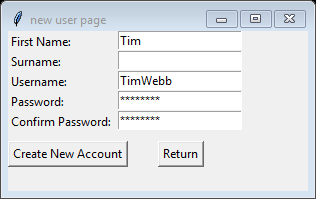


Appendix 113:



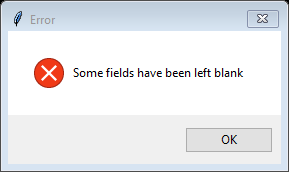
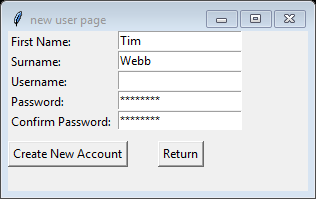
A screenshot of the new user page with the empty field and the pop up error message

Appendix 114:



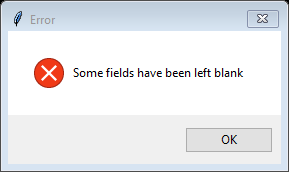
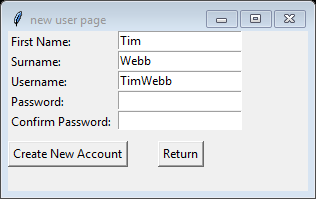
A screenshot of the new user page with the empty field and the pop up error message

Appendix 115:



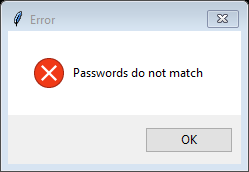
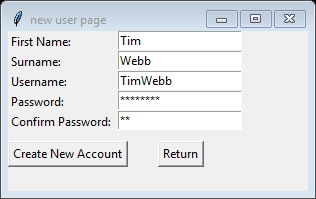
A screenshot of the new user page with the empty field and the pop up error message

Appendix 116:



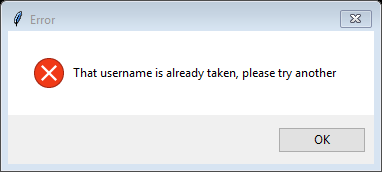
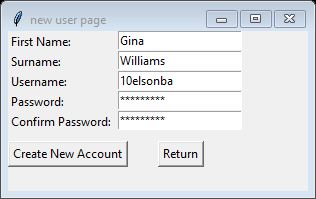
A screenshot of the new user page with the empty field and the pop up error message

Appendix 117:



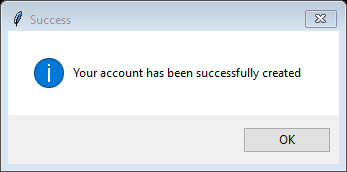
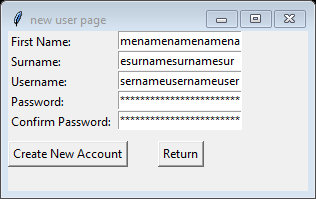
A screenshot of the new user page with the fields password and confirm passwords not matching and the pop up error message

Appendix 118:

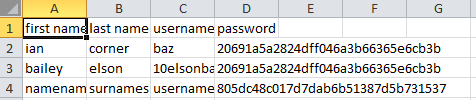


A screenshot of the new user page with a non-unique username and the pop up error message

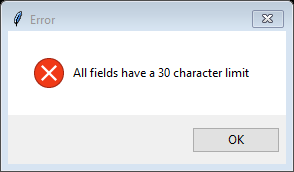
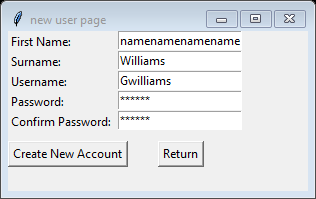
Appendix 119:



A screenshot of the filled out new user page with all fields containing 30 characters, the pop up message and the csv file where the details are stored when the create new user button is pressed

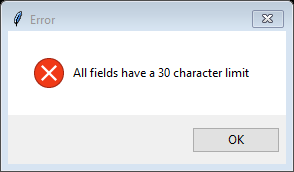
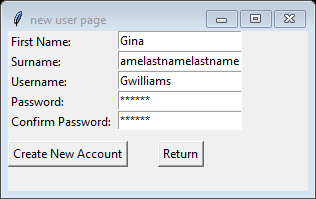


Appendix 120:



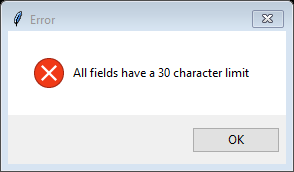
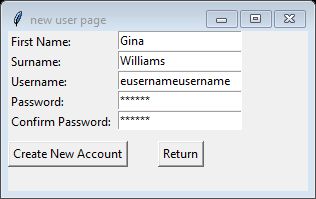
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 121:



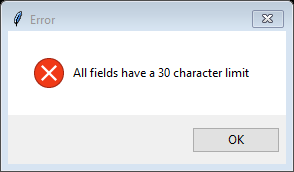
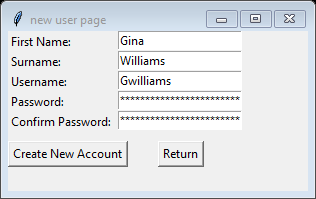
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 122:



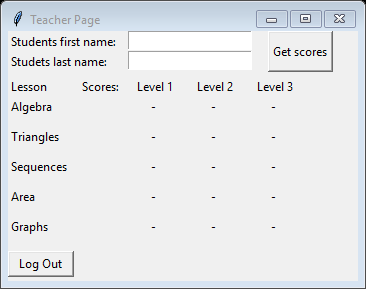
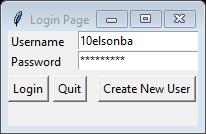
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 123:



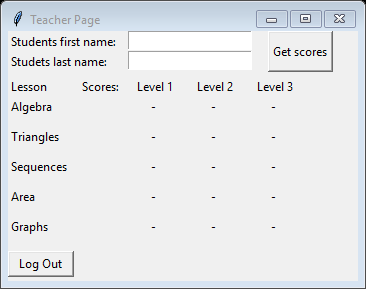
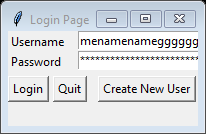
A screenshot of the new user page a field with more the 30 characters and the pop up error message

Appendix 124:



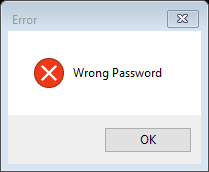
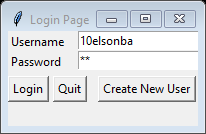
A screenshot a login page with a correct username-password combination and the teacher page that loads when the user logs in

Appendix 125:



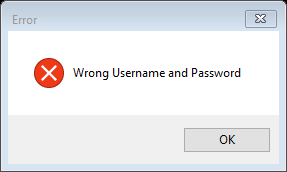
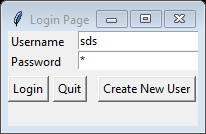
A screenshot a login page with a correct username-password combination (both of 30 characters) and the main page that loads when the user logs in

Appendix 126:



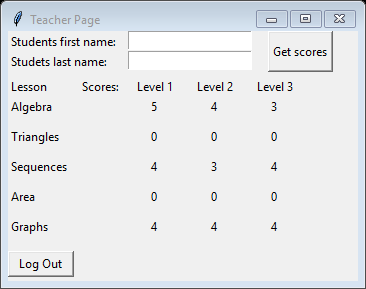
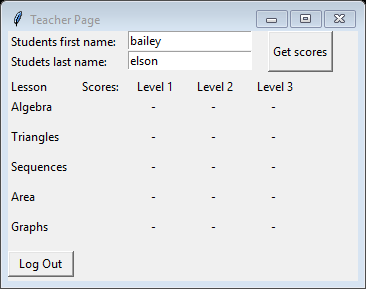
A screenshot a login page with an incorrect username-password combination and pop up error message

Appendix 127:

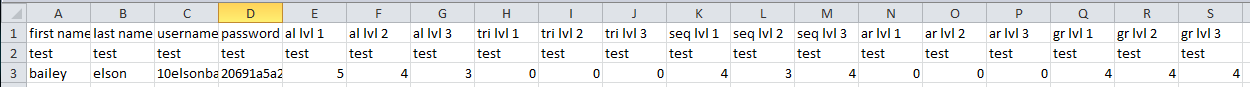


A screenshot a login page with an incorrect username-password combination and pop up error message

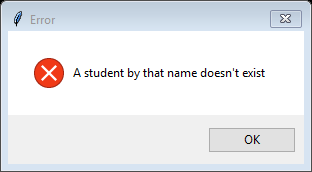
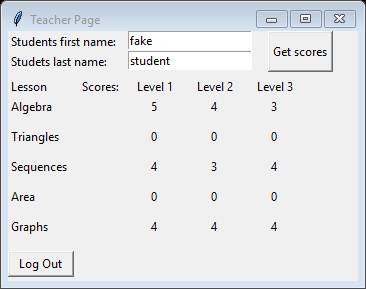
Appendix 128:



A screenshot of the teacher page before and after the get scores button has been pressed when a correct student name has been entered.

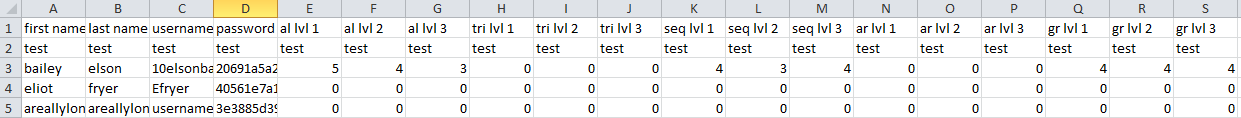
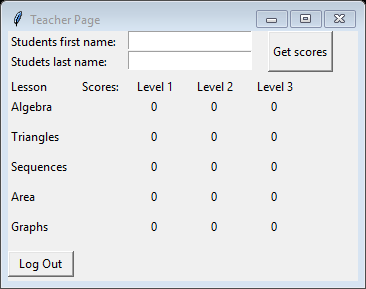
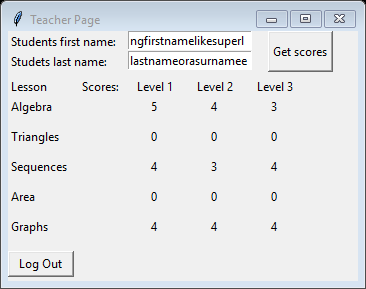


Appendix 129:



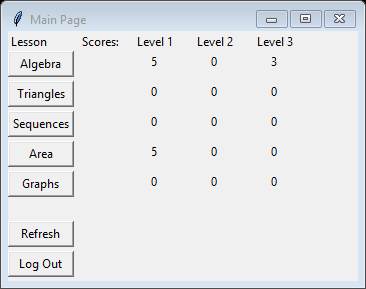
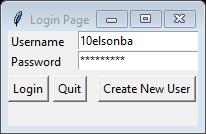
A screenshot of the teacher page before the get scores button has been pressed when an incorrect student name has been entered and the pop up error message

Appendix 130:



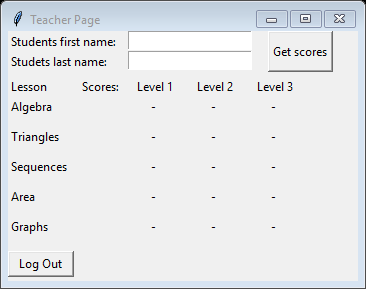
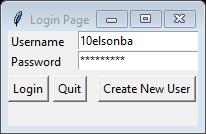
A screenshot of the teacher page before and after the get scores button has been pressed when a correct student name has been entered. (The students name is of 30 characters)

Appendix 131:



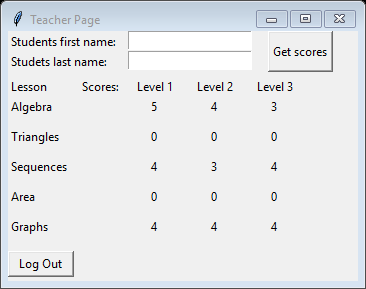
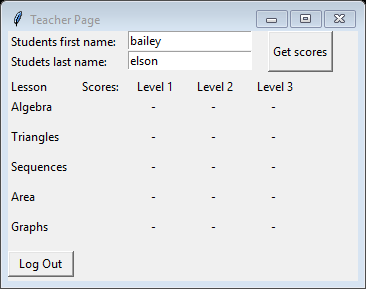
A screenshot a login page with a correct username-password combination and the main page that loads when the user logs in

Appendix 132:



A screenshot a login page with a correct username-password combination and the teacher page that loads when the user logs in

Appendix 133:



A screenshot of the teacher page before and after the get scores button has been pressed when a correct student name has been entered.

# Evaluation *(4 marks)*

## Comparison of project performance against the objectives

Objective 1 – a new user page GUI that will generate a new user account only if the username is unique and isn’t already in use

Objective 1 has been successfully completed. When the user fills out the new user page with their details and then press the “create new user” button the program will compare the entered usrname against all the usernames in the csv file (student\_accounts.csv for students and teachers\_account.csv for teachers) and will only create a new account if the username isn’t already in use.

Objective 2 – a new user page that will show an error message if the password and confirm password sections do not match

Objective 2 has been successfully completed. If the user enters a password that doesn’t match the input for confirm password then the user will receive an error and an account won’t be created. Instead the new user page will still be there where the user can re-enter the password and confirm password fields.

Objective 3 – a new user page that will display an error message if the username is already in use

Objective 3 has been successfully completed. When the user enters a username that is already in use then the program will create a message box inform the user that the username is already taken and to choose another one. The new user page will still be there where the user can re-enter the username field and try to create an account again.

Objective 4 – the new user page will store the new user accounts details (first name, last name, username, hashed password, and score 0 for all lesson and levels for students and first name last name username and hashed password for teachers)

Objective 4 has been successfully completed. When the user successfully creates a new account the account details will be saved in to the corresponding csv file. For students the student\_accounts.csv file will gain an extra row with the student’s first name, last name, username, hashed password and 15 separate fields of the value 0. For the teacher’s the teacher\_accounts.csv file will gain an extra row with the teacher’s first name, last name, username and hashed password.

Objective5 – the program will hash the user’s password before saving it to the csv file

Objective 5 has been successfully completed. For both user’s, teacher and student, the program, will hash the users password using the MD5 hashing algorithm and converting to a 32 bit hexadecimal value before saving the hexadecimal value in the corresponding csv file

Objective 6 – a fully functioning login page that will log the user in if a correct username-password combination has been entered.

Objective 6 has been successfully completed. The login page will login in the user only if they enter a username-password combination that matches one saved in the corresponding csv file. When a student user logs in the main page will load and when a teacher logs in the teacher page will load.

Objective 7 – the login page will display an error message and will not allow the user onto the rest of the program if an incorrect username-password combination has been entered

Objective 7 has been successfully completed. If the username-password combination doesn’t match one saved in the corresponding csv file but the user enters a correct username with a wrong password the error message “wrong password” will appear but if the user name is not connected to a saved account then the error message “incorrect username and password” will appear.

Objective 8 – a main page that the user can use to navigate to different lessons

Objective 8 has been successfully completed. The main page has a button for each lesson that is clearly labelled and when the button is pressed it will create the corresponding question page and ask the user to choose a level.

Objective 9 – a five different lesson all with three levels that randomly generate questions

Objective 9 has been successfully completed. All five lessons have three different difficulties that the user can choose from. Depending on the lesson and the level they choose a different set of questions will be generated

Objective 10 – all three levels on the graph lesson will produce a graph for the user to see

Objective 10 has been successfully completed. All three levels of the graph lesson automatically generate a graph in a separate window that corresponds to the question that has been generated. The graphs generated will only ever be linear, quadratic or cubic.

Objective 11 – for the lessons that require an image (triangles and area), an image will load on the GUI

Objective 11 has been successfully completed. All three levels for both lessons triangles and graphs successfully create a label that displays a corresponding image and places it on the GUI window.

Objective 12 – the user’s sores to be stored in a csv file (students only)

Objective 12 has been successfully completed. When a student user creates a new account the default scores of 0 will be stored in the csv file (student\_accounts.csv) later in the program when the user completes any level of any lesson then the scores are updated to their most recent score on each lesson level in the csv file

Objective 13 – the user’s scores to be updated inside the csv file (students only)

Objective 13 has been successfully completed. When the user completes any level of any lesson then their new score will automatically be saved into the current users row in the student\_accounts.csv file.

Objective 14 – to be able to check the user’s answer against the actual answer for all the levels inside the five lessons

Objective 14 has been successfully completed. Every time the submit button is pressed on a question page the user’s inputted answer is compared against the actual answer an a message box will appear with either the message “correct” or “incorrect” depending on whether the user answer matches the actual answer or not.

Objective 15 – to be able to log back out of the program and this causing the login page to reload

Objective 15 has been successfully completed. When the log out button is pressed the user gets logged out of the program and the login page will reload. When a student logs out the student log in page will load.

Objective 16 – level 1 of algebra will generate random questions in the format ax = b (a and b representing integer values) which are solvable

Objective 16 has been successfully completed. When level 1 of the lesson algebra is selected then the program randomly generates and displays a question in the format ax = b. All the questions generated are solvable as integers.

Objective 17 – level 2 of algebra will generate random questions in the form ax + b = c and ax – b = c (a,b and c representing integer values) which are solvable

Objective 17 has been successfully completed. When level 2 of the lesson algebra is selected then the program randomly generates and displays a question in the format form ax + b = c or ax – b = c per question. All the questions generated are solvable as integers.

Objective 18 – level 3 of algebra will generate random questions in the form ax + b = cx + d,

ax – b = cx – d, ax + b = cx – d and ax – b = cx + d (a,b,c and d representing integer values) which are solvable

Objective 18 has been successfully completed. When level 3 of the lesson algebra is selected then the program randomly generates and displays a question in the format form ax + b = cx + d,

ax – b = cx – d, ax + b = cx – d or ax – b = cx + d per question. All the questions generated are solvable as integers.

Objective 19 – level 1 of area will generate random questions, asking for the area of a rectangle or square with both side A and side B being randomly generated integers, which are solvable

Objective 19 has been successfully completed. When level 1 of the lesson area is selected then the program randomly generates and displays with both side A and side B being randomly generated integers per question. All the questions generated are solvable as integers.

Objective 20 – level 2 of area will generate random questions, asking for the area of a circle with the radius being a randomly generated integer, which are solvable

Objective 20 has been successfully completed. When level 2 of the lesson area is selected then the program randomly generates and displays the radius being a randomly generated integer per question. All the questions generated are solvable as real numbers to one decimal place.

Objective 21 – level 3 of area will generate random questions, asking for the area of a trapezoid with both side A, side B and the height being randomly generated integers, which are solvable

Objective 21 has been successfully completed. When level 3 of the lesson area is selected then the program randomly generates and displays the both side A, side B and the height as randomly generated integers per question. All the questions generated are solvable as real numbers to one decimal place.

Objective 22 – level 1 of triangles will generate random questions, asking for the third angle in a triangle with the other two angles being randomly generated integers which are solvable

Objective 22 has been successfully completed. When level 1 of the lesson triangles is selected then the program randomly generates and displays the both angle A and angle B as randomly generated integers per question. All the questions generated are solvable as integers.

Objective 23 – level 2 of triangles will generate random questions, asking for the value of the third side of a right angle triangle where the other two sides are randomly generated integers, which are solvable

Objective 23 has been successfully completed. When level 2 of the lesson triangles is selected then the program randomly generates and displays two sides of the triangle (either sides a and b or b and c) as randomly generated integers per question. All the questions generated are solvable as real numbers to one decimal place.

Objective 24 – level 3 of triangles will generate questions, asking for the value of the third side of a triangle where the other two sides and two angles are randomly generated integers, which are solvable

Objective 24 has been successfully completed. When level 3 of the lesson triangles is selected then the program randomly generates and displays two side A, side B, angle A and angle B as randomly generated integers per question. All the questions generated are solvable as real numbers to one decimal place.

Objective 25 – level 1 of sequences will generate random questions, in the form nth term = ax +/- b what is the c term (a,b and c representing randomly generated integers), which are solvable

Objective 25 has been successfully completed. When level 1 of the lesson sequences is selected then the program randomly generates a question in the form nth term = ax +/- b what is the c term All the questions generated are solvable as integers.

Objective 26 – level 2 of sequences will generate random questions, in the form nth term = +/- b what is the c term (a,b and c representing randomly generated integers), which are solvable

Objective 26 has been successfully completed. When level 2 of the lesson sequences is selected then the program randomly generates a question in the form nth term = +/- b what is the c term All the questions generated are solvable as integers.

Objective 27 – level 3 of sequences will generate random questions, in the form [a,b,c,d,e] is a sequence what is the nth term in terms of x (where [a,b,c,d,e] represent a random sequence), which are solvable

Objective 27 has been successfully completed. When level 3 of the lesson sequences is selected then the program randomly generates a question in the form [a,b,c,d,e] is a sequence what is the nth term in terms of x. All the questions generated are solvable as in the form ax+c or ax-c (a,b and c representing integers).

Objective 28 – level 1 of graphs will generate random questions, producing a graph showing a linear function and asking what does y equal when x = a (a representing a randomly generated integer), which are solvable

Objective 28 has been successfully completed. When level 1 of the lesson graphs is selected then the program randomly generates a question in the form [what does y equal when x = a and a linear graph. All the questions generated are solvable as integers.

Objective 29 – level 2 of graph will generate random questions, producing a linear graph and asking is this graph ax +/- b (a and b represent randomly generated integers), which are solvable

Objective 29 has been successfully completed. When level 2 of the lesson graphs is selected then the program randomly generates a question in the form is this graph ax +/- b and a linear graph. All the questions generated are solvable as true or false.

Objective 30 – level 3 of graph will generate random questions, producing a linear, quadratic or cubic graph and asking what type of graph is this, which are solvable

Objective 30 has been successfully completed. When level 3 of the lesson graphs is selected then the program randomly generates a question asking what type of graph it is and a linear graph. All the questions generated are solvable as linear, quadratic or cubic.

Objective 31 – a teacher’s page will load when a teacher successfully logs in

Objective 31 has been successfully completed. When a teacher logs in using a correct username-password combination then the teacher page will load. The teacher page contains two entry boxes, a button and multiple labels

Objective 32 – if a correct students first and last name are entered into the teacher page their score will successfully be displayed

Objective 32 has been successfully completed. When a student’s first and last name are entered on the teachers page and it matches a student’s accounts details in the student\_accounts.csv file then the students’ scores are displayed on labels in a table format

Objective 33 – if the student’s name that is entered doesn’t match any of the students in the csv file then an error message will be displayed

Objective 33 has been successfully completed. When a student’s first and last name are entered on the teacher’s page and it doesn’t match a student’s accounts details in the student\_accounts.csv file then an error message will appear in a message box.

## Effectiveness of the solution

Functionality – my program meets all of the 33 pre-set objectives. Therefore I would say that my program meets its function and that the program would be able to be used in the required setting effectively. This meaning that the students and teacher users would be able to use the program without any issues and the program covers all of the user’s needs.

Ease of use: the complexity of the program is suitable to the user abilities. The program is laid out in a simple manner that means that the users can easily navigate and use the program

Ease of implementation: the change over from the old to the new system would be a simple process and all it requires is the new system being used at the start of a new school year.

Reliability – as shown in the testing phase the program doesn’t have any bugs to my knowledge and the program runs smoothly. The only possible issue is if the user has the teacher\_accouts.csv file or the student\_accouts.csv file open when the program is running. This would cause the program to not be able to save the user’s details to the file or update the user’s details in the file.

Performance – the program is capable of handling the volume of data that is necessary for the program to be used effectively in the set environment. As show in the final three tests the program is easily able to handle over 150 users in each file without it having a knock on effect on the programs speed or capabilities.

Ease of maintenance and adaptability – as the program is split into different sections adding a new element to the program would be simple. For example if you wanted to add a new lesson all you would have to do is create a new class for that lesson, the class layout would be identical to all the other lesson classes, add a new button to the main page and modify the csv file.

## Analysis of user feedback

**User interview (student):**

The students were happy with the program created and were easily able to use it; there three main needs from the first interview were all met

**Log in credentials** – the students were happy with the username system and how it works. They like the fact that they are able to choose their own usernames.

**Layout** – the students liked the layout of the program saying that it was “easy to navigate” and there were able to access the whole program without any help.

**Answer checking** – from the students interaction with the program they found no issues with the answer checking system saying that the program made it clear what form it wanted the answer in.

**User interview (teacher):**

The teachers were happy with the program created and were easily able to use it; there three main needs from the first interview were all met.

**Student score viewing** – they like the system of just typing in the students first and last name and being able to view the students’ scores, they said that it was laid out in a easy to use wway and the program didn’t over complicate the process.

**Accessibility** – they found the GUI easy to use and they were able to navigate their way through the program without and help saying that the program didn’t over complicate anything.

**Multiple questions** – they were happy with the repeated questions saying it gave them a clearer vie on what level the students are on.

## Possible improvements

The only possible improvement I got from the feedback was possibly changing the program to store the user’s highest score on the level instead of their most recent. This would be an easy implementation. You would simply have to add an if statement that compares the user’s new score to their high score and only update the file if the new score is higher when the submit button is pressed on the fifth level of any lesson.