Contents

[Analysis 4](#_Toc34907342)

[Problem 4](#_Toc34907343)

[Problem Background 5](#_Toc34907344)

[Definitions 5](#_Toc34907345)

[Graphs and Networks 5](#_Toc34907346)

[Degree of vertex 5](#_Toc34907347)

[Distance Matrix 5](#_Toc34907348)

[Nearest Distance Matrix 5](#_Toc34907349)

[Graph Traversing 6](#_Toc34907350)

[Eulerian/semi-Eulerian graph 6](#_Toc34907351)

[Critical Path Analysis 6](#_Toc34907352)

[Precedence Table 6](#_Toc34907353)

[Linear Programming 6](#_Toc34907354)

[Decision Variables 6](#_Toc34907355)

[Constraints 6](#_Toc34907356)

[Objective Function 6](#_Toc34907357)

[Example 6](#_Toc34907358)

[Slacks 7](#_Toc34907359)

[Difference between Maximizing and Minimizing the Objective function. 7](#_Toc34907360)

[Existing Solutions 7](#_Toc34907361)

[AtoZmath.com 7](#_Toc34907362)

[My Findings 7](#_Toc34907363)

[Client’s and target audience’s findings 12](#_Toc34907364)

[Writing Method 13](#_Toc34907365)

[My Findings 13](#_Toc34907366)

[Client’s and target audience’s findings 14](#_Toc34907367)

[Client Interview (Helen Morgan) 14](#_Toc34907368)

[Client Interview (Karen Tunnicliff) 15](#_Toc34907369)

[Algorithms 16](#_Toc34907370)

[Linear Programming (Simplex Algorithm): 16](#_Toc34907371)

[Critical Path Analysis 17](#_Toc34907372)

[Minimum Spanning Tree (Kruskal’s) 18](#_Toc34907373)

[Minimum Spanning Tree (Prim’s) 19](#_Toc34907374)

[Chinese Postman (Route Inspection) Problem 19](#_Toc34907375)

[Travelling Salesman Problem 19](#_Toc34907376)

[Upper Bound 20](#_Toc34907377)

[Lower Bound 20](#_Toc34907378)

[Graphical User Interface (GUI) 20](#_Toc34907379)

[Tkinter 20](#_Toc34907380)

[Pygame 21](#_Toc34907381)

[Clients Preference 22](#_Toc34907382)

[Solution Objectives 22](#_Toc34907383)

[Necessary Objectives 22](#_Toc34907384)

[Unattainable Objectives 24](#_Toc34907385)

[Documented Design 24](#_Toc34907386)

[Algorithms (continued) 24](#_Toc34907387)

[Login Screen 24](#_Toc34907388)

[Admin Screen 24](#_Toc34907389)

[Main Menu 25](#_Toc34907390)

[Linear Programming 25](#_Toc34907391)

[Critical Path Analysis 26](#_Toc34907392)

[Graph Problems 27](#_Toc34907393)

[Matrix 27](#_Toc34907394)

[Dijkstra’s Algorithm 28](#_Toc34907395)

[Minimum Spanning Tree 29](#_Toc34907396)

[Chinese Postman 29](#_Toc34907397)

[Travelling Salesman Problem 30](#_Toc34907398)

[Answering Questions 31](#_Toc34907399)

[Data Structures 31](#_Toc34907400)

[Matrix 31](#_Toc34907401)

[Array 31](#_Toc34907402)

[Activity 32](#_Toc34907403)

[Activity Network 32](#_Toc34907404)

[Dictionary 32](#_Toc34907405)

[GUI (continued) 32](#_Toc34907406)

[Login Screen 32](#_Toc34907407)

[First Use Screen for Users 32](#_Toc34907408)

[Admin Screen 32](#_Toc34907409)

[Class Creation Screen 33](#_Toc34907410)

[Class Editor Screen 33](#_Toc34907411)

[Success Rate Viewer 34](#_Toc34907412)

[User Creation Screen 34](#_Toc34907413)

[New User Password Display 34](#_Toc34907414)

[Algorithm Window 34](#_Toc34907415)

[Main Menu 35](#_Toc34907416)

[Linear Programming 35](#_Toc34907417)

[Critical Path Analysis 36](#_Toc34907418)

[Minimum Spanning Tree, Chinese Postman, Travelling Salesman problem 36](#_Toc34907419)

[Overall Clients thoughts on GUI 36](#_Toc34907420)

[Changes after client feedback 37](#_Toc34907421)

[Password Storing 37](#_Toc34907422)

[Problem 1 37](#_Toc34907423)

[Solution 37](#_Toc34907424)

[Storing 37](#_Toc34907425)

[Testing 37](#_Toc34907426)

[Problem 2 37](#_Toc34907427)

[Solution 37](#_Toc34907428)

[Database 37](#_Toc34907429)

[Data Required 37](#_Toc34907430)

[Normalisation 39](#_Toc34907431)

[No normal form 39](#_Toc34907432)

[First Normal Form 39](#_Toc34907433)

[Second Normal Form 41](#_Toc34907434)

[Third Normal Form 41](#_Toc34907435)

[Removing old Questions 41](#_Toc34907436)

[Entity Relationship Diagram 42](#_Toc34907437)

[Creating Question String 43](#_Toc34907438)

[SQL Statements 43](#_Toc34907439)

[Creating Database 43](#_Toc34907440)

[Editing Password for first time login 44](#_Toc34907441)

[Logging in 44](#_Toc34907442)

[Updating success rate 44](#_Toc34907443)

[Adding new questions to database 45](#_Toc34907444)

[Creating new class 45](#_Toc34907445)

[Opening class 45](#_Toc34907446)

[Viewing Success Rates 45](#_Toc34907447)

[Changing available solvers 46](#_Toc34907448)

[Adding new Student 46](#_Toc34907449)

[Deleting Student 46](#_Toc34907450)

[Class Relationship diagram: 47](#_Toc34907451)

[Overall Design 50](#_Toc34907452)

[Imports 50](#_Toc34907453)

[Tkinter 50](#_Toc34907454)

[Sys 50](#_Toc34907455)

[Functools 50](#_Toc34907456)

[Copy 50](#_Toc34907457)

[SQLite3 51](#_Toc34907458)

[String 51](#_Toc34907459)

[Random 51](#_Toc34907460)

[Hashlib 51](#_Toc34907461)

[Uuid 51](#_Toc34907462)

[First use of Program (Admin and User) 51](#_Toc34907463)

[How the GUI subsections mix 52](#_Toc34907464)

[Users 52](#_Toc34907465)

[Admins 53](#_Toc34907466)

[Database Interactions 54](#_Toc34907467)

[Logging in 54](#_Toc34907468)

[With admins 54](#_Toc34907469)

[With users 54](#_Toc34907470)

[Changes Made during production 54](#_Toc34907471)

[Technical Solution 55](#_Toc34907472)

[Testing 135](#_Toc34907473)

[Testing Plan 135](#_Toc34907474)

[Link to testing video 137](#_Toc34907475)

[Evaluation 137](#_Toc34907476)

[Final Questionnaire 137](#_Toc34907477)

[Questionnaire results and review 137](#_Toc34907478)

[Helen Morgan 137](#_Toc34907479)

[Karen Tunnicliff 138](#_Toc34907480)

[User 1 139](#_Toc34907481)

[User 2 139](#_Toc34907482)

[Commenting on criteria in objective order stated in Analysis 139](#_Toc34907483)

[References 141](#_Toc34907484)

# Analysis

## Problem

**Project:** Discrete Mathematics Revision tool and Solver.

**Client:** Sharnbrook Academy Maths Department and Further Mathematics A-Level students.

## Problem Background

For Further Maths students studying the AQA A-Level Further Mathematics course they have the option to study Discrete Mathematics (Mathematical structures that are fundamentally discrete rather than continuous). In this part of their course they will learn many algorithms such as: Prims Algorithm, Kruskal’s Algorithm, Nearest Neighbour Algorithm, etc. As well as learning to solve different problems such as: Travelling Salesman Problem, Chinese Postman (Route Inspection) Problem, Critical Path Analysis and Linear Programming. For most of these students all of these concepts and problems will be completely new and they may struggle with getting to grips with using them. This project is intended to give students the resources to Learn Discrete Mathematics and help them solve some of the Problems.

The Problems I am looking to solve with this programme are:

* Linear Programming problems (Using simplex algorithm)
* Critical Path Analysis – finding earliest start and latest finish time.
* Minimum spanning Tree – Kruskal’s and Prims
* Chinese Postman (Route Inspection) Problem
* Travelling Salesman Problem – finding upper and lower bounds for specified starting node

## Definitions

### Graphs and Networks

A graph consists of a number of points (vertices) connected by a number of lines (edges).

#### Degree of vertex

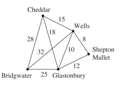
The degree of a vertex is the number of edges that meet at a vertex. E.g.



Point B has degree 3 and point C has degree 4.

#### Distance Matrix

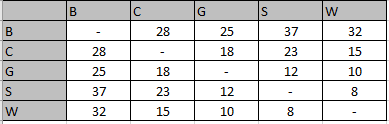
A matrix representing the distances between directly linked nodes on a graph. E.g.

   
This graph can be represented by the following Distance Matrix



#### Nearest Distance Matrix

Similar to a distance matrix but the towns don’t need to be directly linked. The distance of the shortest path between 2 nodes is used. Following the above example, the nearest distance matrix is



#### Graph Traversing

* A Connected Graph is a graph you can travel from one node to any other node in the graph
* A walk is a continuous journey around a graph with no restrictions
* A walk with no repeated edges is called a trail. If a trail returns to its starting node it is a Close Trail.
* A walk with no repeated edges or vertices is called a path.
* A closed path is called a Cycle.
* A connected graph with no cycles is a Tree.
* A Spanning Tree is a tree containing every node of a graph.

#### Eulerian/semi-Eulerian graph

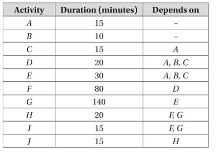
A graph in which a trail can be produced using every edge in which the trail starts and ends at the same node is said to be Eulerian. For a graph to be Eulerian all nodes must have an even degree.

A semi-Eulerian graph is similar to an Eulerian graph but the starting and finishing points are different. For a graph to be semi-Eulerian exactly 2 nodes must have an odd degree.

### Critical Path Analysis

#### Precedence Table

Is the table used to list the Activities with their durations and predecessors. Looks as follows:



### Linear Programming

#### Decision Variables

The variables which you can change in the problem.

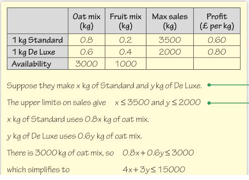
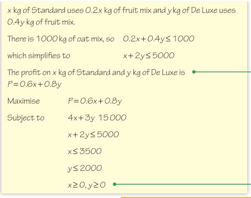
#### Constraints

The inequalities that must be abided by.

#### Objective Function

The function (made up of the Decision variables) that needs to be either Maximised or Minimised.

#### Example

Constraints

Objective Function

Decision Variables

#### Slacks

When using the Simplex algorithm slacks are introduced. Slacks are values used to turn an inequality into an equality. In the above example we can take the first constraint (4x+3y<=15000) add a constraint called “s”. “s” can take any value and the use of it is to be added to 4x+3y to make exactly 15000. This leaves us with the equality 4x+3y+s=15000.

However, if the inequality uses the >= sign a different process has to be used. First of all, the entire equation is multiplied by -1. This then flips the inequality sign to the sign we want for our problem. From this we can add a slack as shown before.

#### Difference between Maximizing and Minimizing the Objective function.

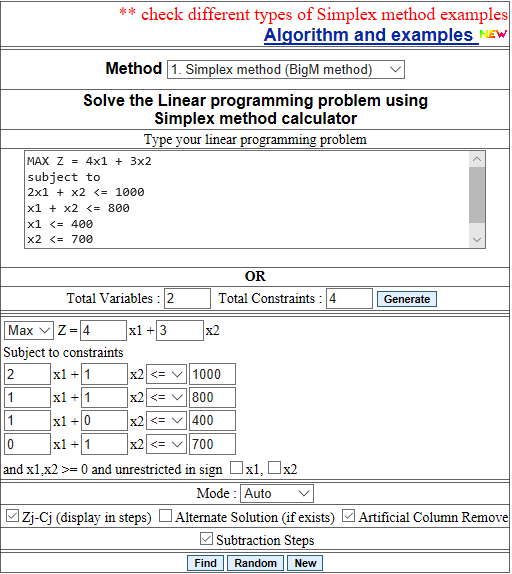
When maximizing the cost function, the simplex algorithm can be used directly with the cost function given. However, when minimising the Objective function (C) we set a variable P = -C. Then we maximise P with respect to the constraints.

## Existing Solutions

### AtoZmath.com

#### My Findings

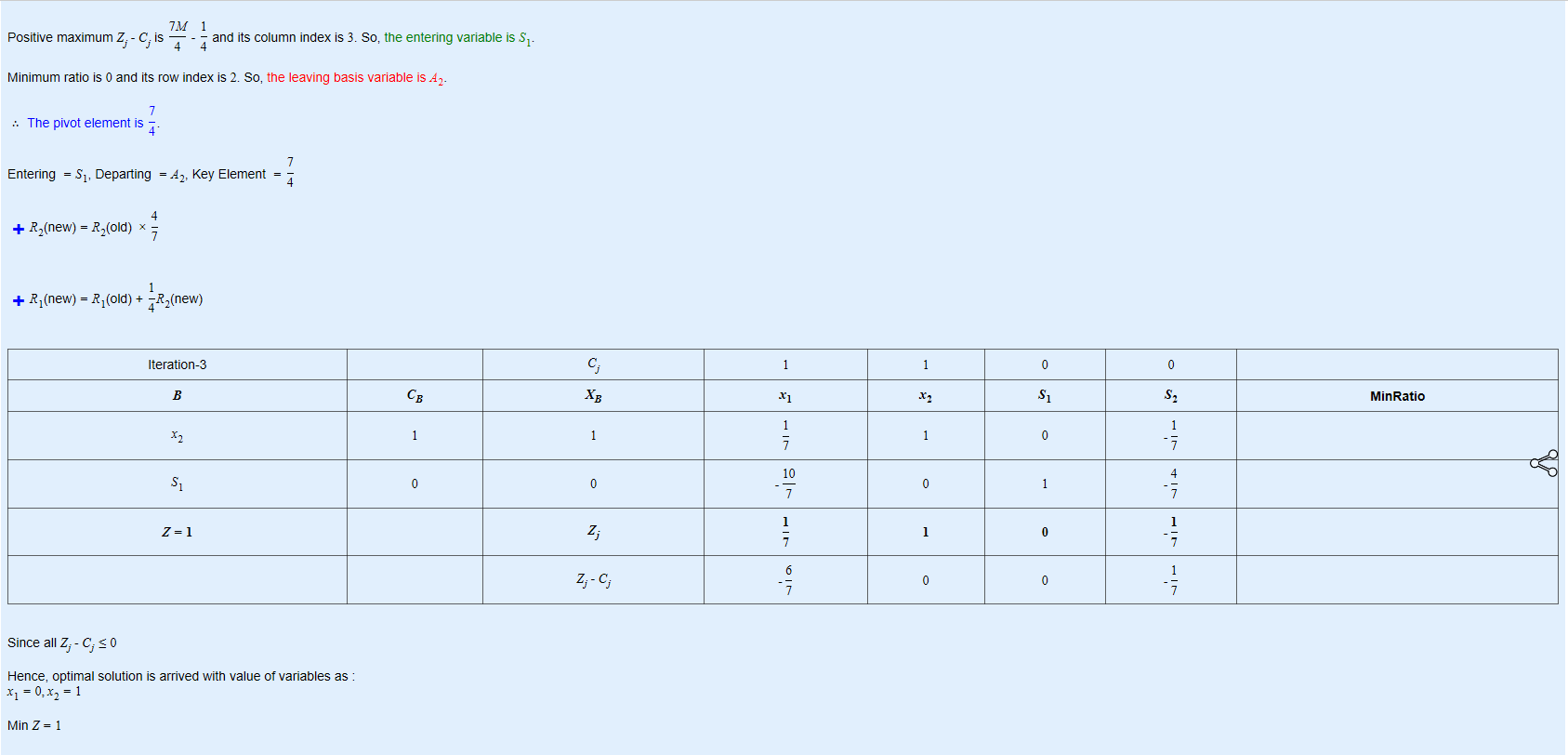
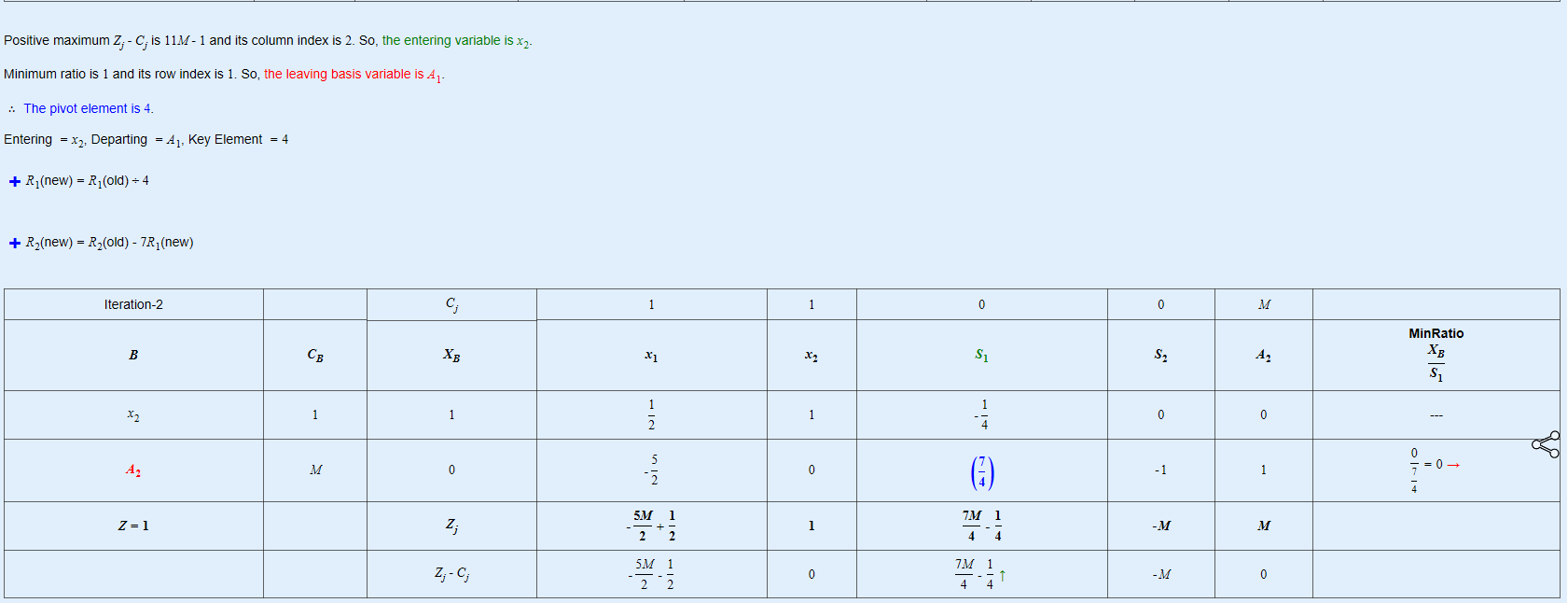
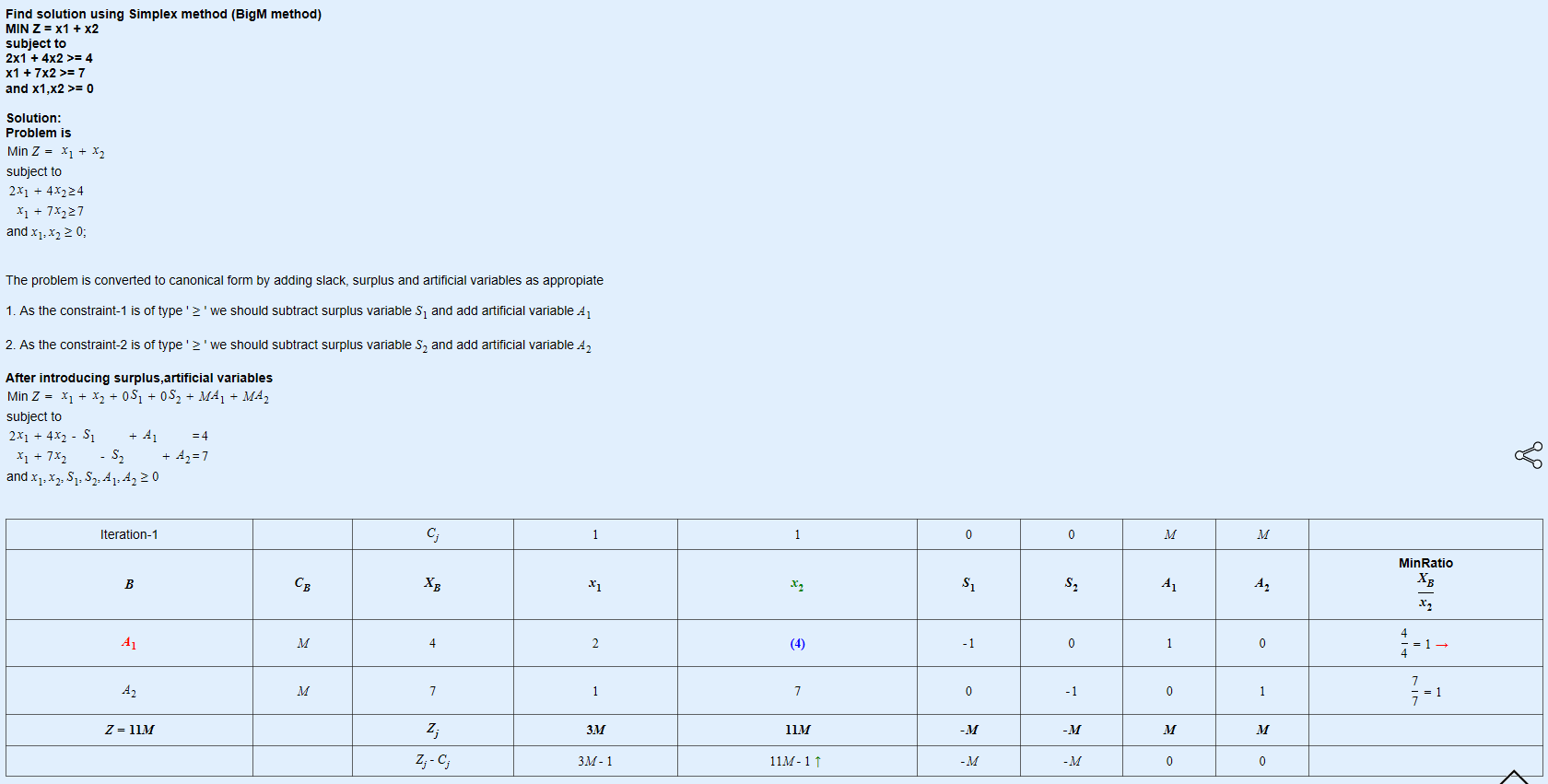
This Website Is used to solve some of the Problems I Specified in my [Problem Background](#_Problem_Background). I will analyse each part of the website Individually starting with the Linear Programming solver:



(Simplex method calculator, 2019)

The Layout of the Input boxes for this Problem is very useful for the user. This is due to the Availability of using 2 different inputs. For users that use the website often and have learnt how to format their problems they can quickly type out their problems in the text box Shown by a yellow box in the above image. This stores the entire question as a large string. However, for less common or new users of this website they have also made a more formatted input section using dropdown menus and smaller input boxes (shown by a Red box in the image above). This will store the variables (x1,x2, etc.) as Integers; the inequality signs either as strings or objects and the other options (squared in blue) as Boolean variables. I prefer the latter of the two as it leaves less room for user error to interfere with the programme.

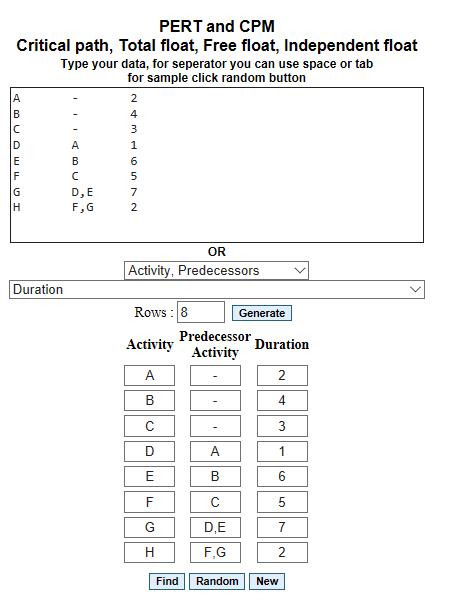
Another bonus of this programme is that it shows a step-by-step of the solution. However, I find this step-by-step guide extremely hard to understand and I believe my target audience would also think the same.



(Simplex method calculator, 2019)

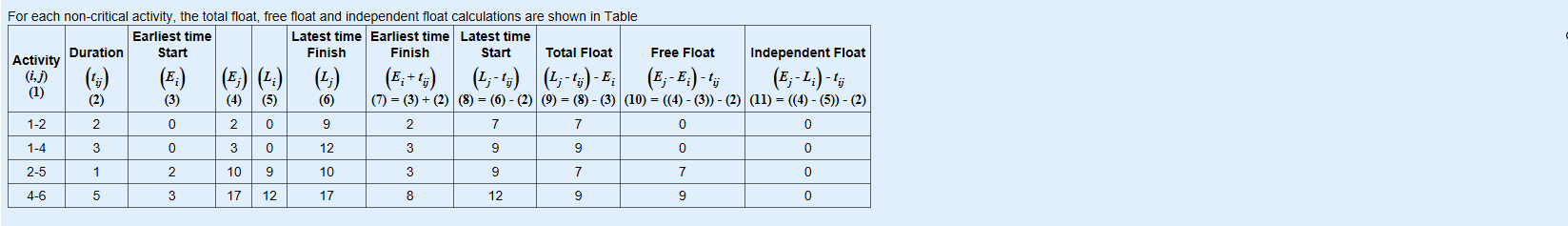
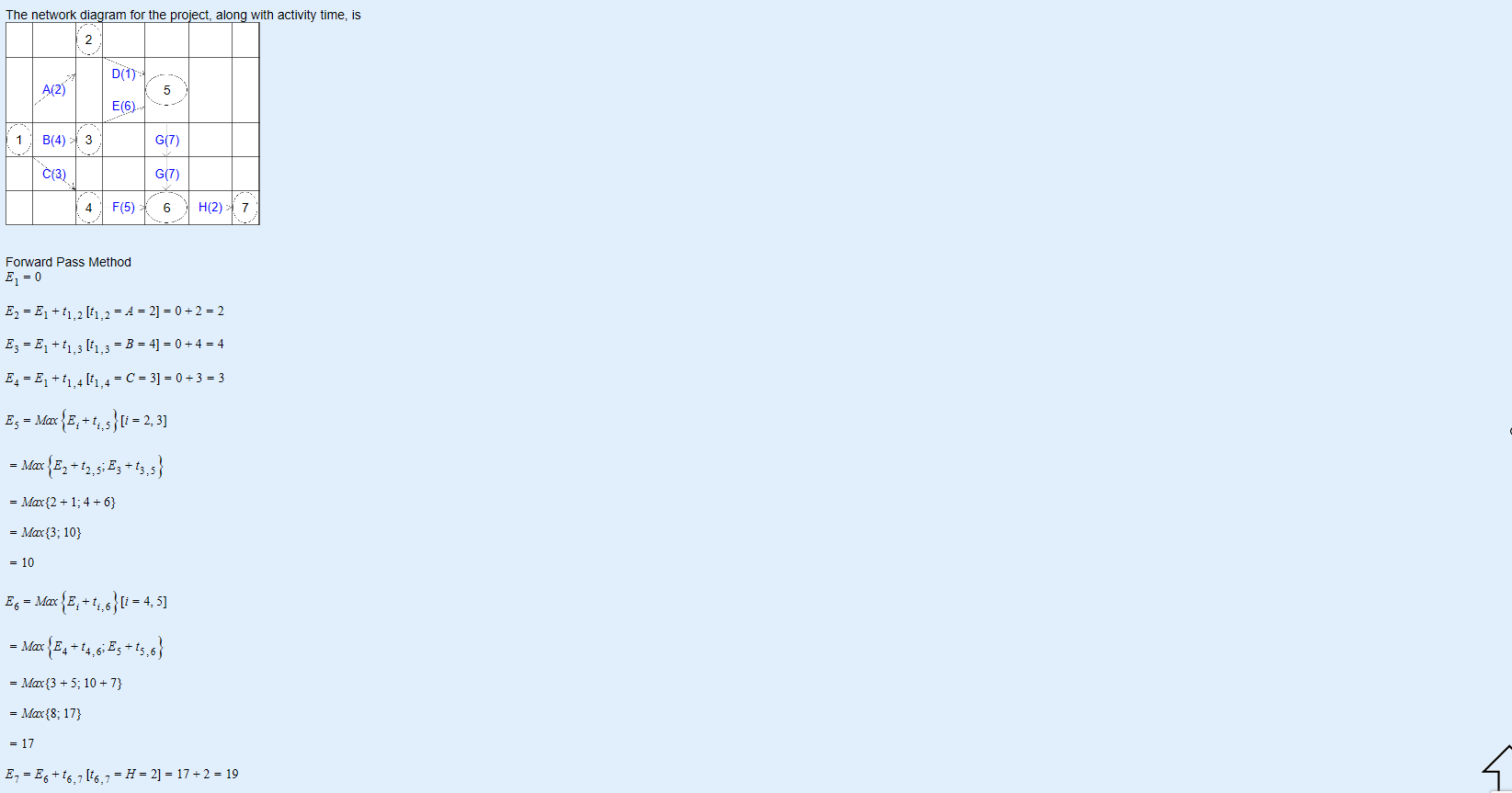
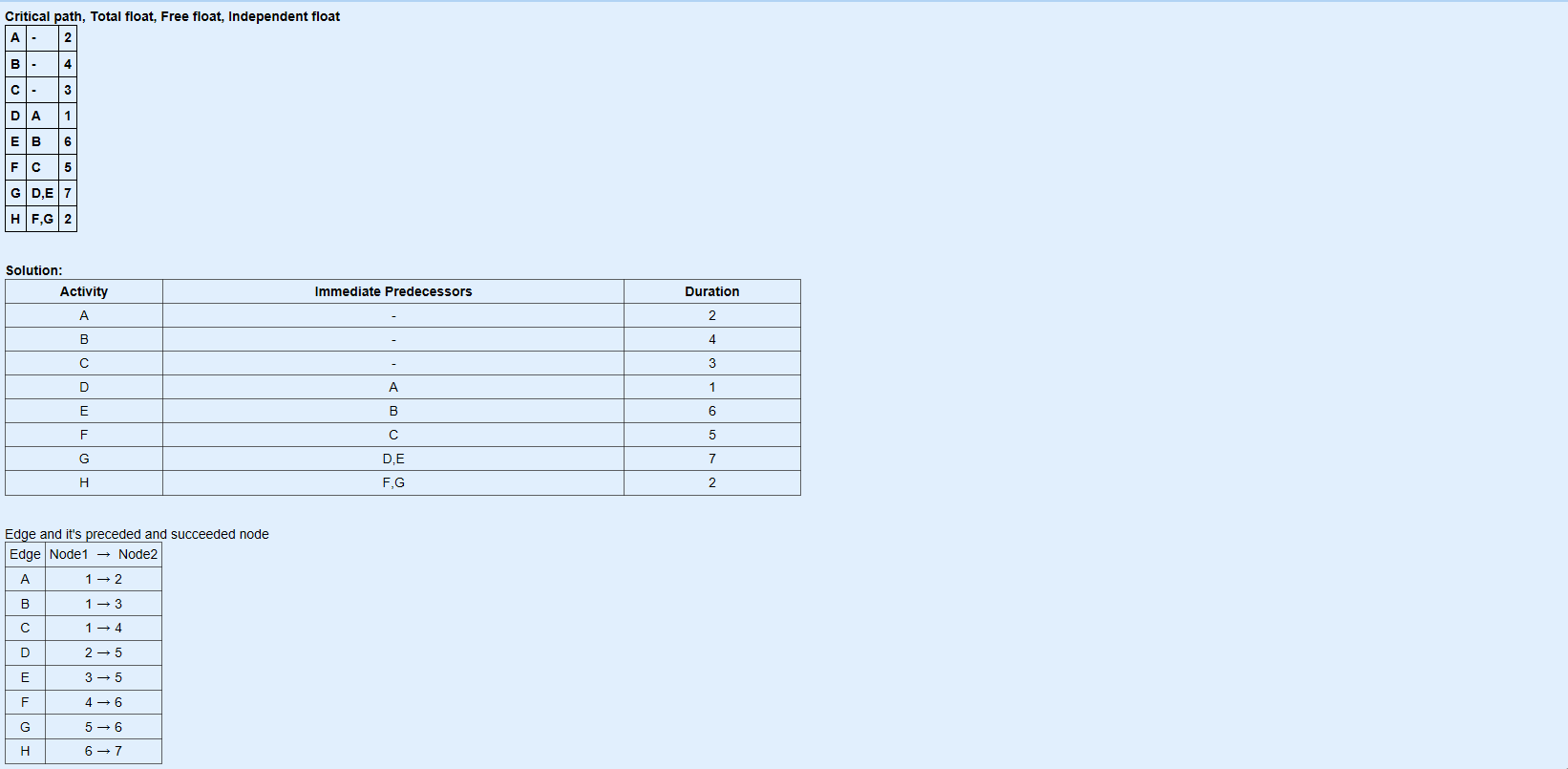
Above is the given step-by-step guide for an example question. The method used here is slightly different to what is taught on the AQA specification. This makes this step-by-step guide useless for any AQA A-Level students which is whom this product is intended for.

Critical Path Analysis:



(Critical path, Total float, Free float, Independent float calculator, 2019)

Similarly, to the Linear Programming solver, the Critical Path analysis solver also has 2 methods of input. Both being the same style as the linear programming solver. Input boxes in the “Activity” and “Predecessor Activity” column will be inputted as strings which will be split by commas. The Duration column will be stored as Integers.



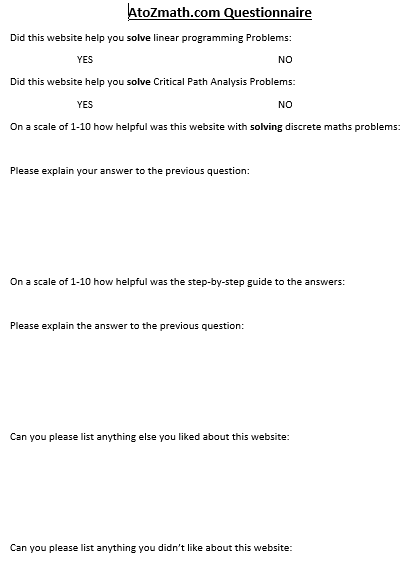
(Critical path, Total float, Free float, Independent float calculator, 2019)

Above is the step-by-step solution for the critical path analysis problem. Again, following the step-by-step guide is extremely hard and I believe it could be simplified down so A-Level Further Maths students can understand it and learn from it. The main Problem with the step-by-step solution is that nothing is explained. New variables are shown (e.g. En) and there is no explanation on what it is.

Overall this website contains many different algorithms to solve many different problems. I’m not going to analyse all of them as they all have a similar format and will seem very similar to the user. There is no accessible source code so I can’t analyse the algorithms that have been written.

#### Client’s and target audience’s findings

I gave my Further Maths class and teacher the link to AtoZmath.com and the following questionnaire to try and get an understanding on how useful this website was.



1. Did this website help you solve linear programming problems?
   1. Everyone checked yes on this question so the website successfully completed its task
2. Did this website help you solve Critical Path Analysis Problems?
   1. Again, everyone checked yes so, the website successfully completed its task
3. On a scale of 1-10 how helpful was this website with solving discrete maths problems:
   1. Scored an average of 9. The reason being it wasn’t scored a 10 was the fact some users found it hard to locate where the answer was on the webpage
4. On a scale of 1-10 how helpful was the step-by-step guide to the answers:
   1. Scored an average of 5. Most of my clients could not understand the notation in which the questions were answered. One of my clients stated, “The use of E1 and L1 should be described to better understand what the values are representing”
5. Can you please list anything else you liked about the website:
   1. Most of my clients liked the help section available on the webpage.
6. Can you please list anything else you didn’t like about the website:
   1. My clients did not like the presentation of the website; saying it was “Too bland” and “not colourful”.

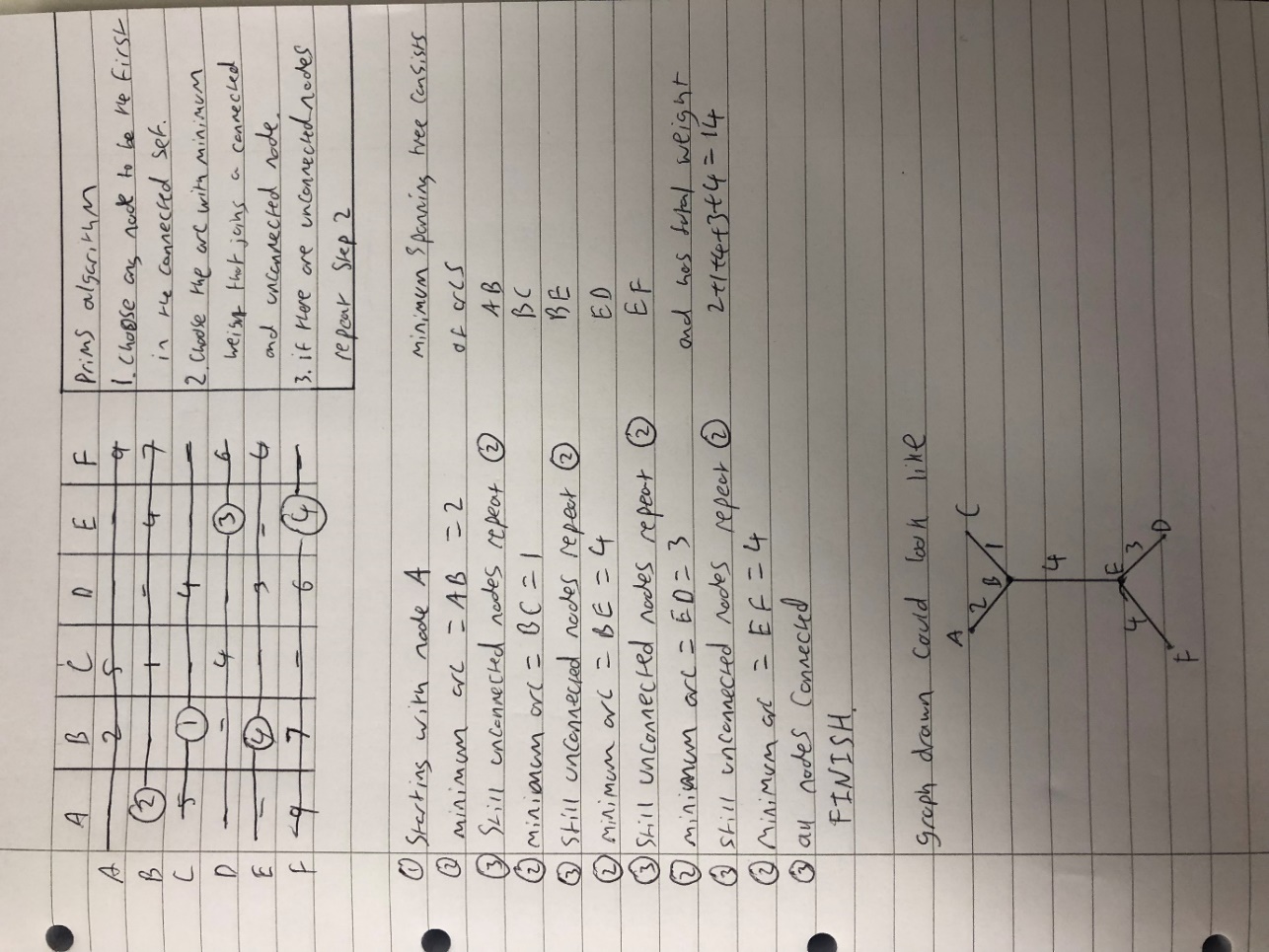
From the response I have received from my clients I have identified some necessary objectives for my programme:

* The GUI must be colourful and nice to look at
* Use common variables in the AQA Further Mathematics so the intended audience understands the program better.
* There must be a help section to:
  + Explain how the inputs work
  + Could be used to explain variables

### Writing Method

#### My Findings

Currently in our Further Maths class we work out the answers to these problems using pen and paper. We have to memorise the Algorithms (or read them from the book) and apply them to the Questions we are given. An example of working out the Minimum Spanning Tree using the Prims algorithm is shown below:



(Fitzmaurice, 2019)

The image above shows the Algorithm used in the top right and how it is implemented in a question. Working through one of these questions can take around 4-5 minutes. This can be significantly improved by a computerised implementation of the Algorithm.

This process takes a long time and can easily be affected by human error as it is all completed by a human.

#### Client’s and target audience’s findings

* “Working out discrete maths problems by hand can be very rewarding if you get the right answer. However, there are many places for small silly mistakes”
* “Way too time consuming”
* “impractical for actually implementing the problems in real life scenarios”

## Client Interview (Helen Morgan)

Q: On a scale of 1-10 (10 being the hardest) how hard is it to teach a class discrete mathematics

A: 7

Q: Please explain any difficulties that could occur in teaching discrete mathematics

A: Some topics are very hard to get to grips with, such as the travelling salesman problem. Also, the resources our Maths Department currently have do not give accurate answers to questions 100% of the time.

Q: could having an available program to solve these problems be useful in teaching discrete mathematics and if so why?

A: Yes. Pupils in this day and age are a lot more technologically minded so having technological resources available to them to help them with questions will be helpful in teaching discrete mathematics.

Q: Do you believe the program should have text versions of the algorithms and examples available to learn from?

A: Definitely. Examples are always beneficial when teaching how to use algorithms in discrete mathematics

Q: Have you ever tried to use a program similar to the one described?

A: No

Q: Which sections of Discrete Mathematics would be best to have a solver available for

A: Linear Programming, Graph Problems, Critical Path Analysis

Q: If not, why haven’t you needed one? If so, could you please tell me what you have used before?

A: Haven’t looked for one

## Client Interview (Karen Tunnicliff)

Q: On a scale of 1-10 (10 being the hardest) how hard is it to teach a class discrete mathematics

A: 6

Q: Please explain any difficulties that could occur in teaching discrete mathematics

A: The only hard part of this topic is ensuring everyone remembers the algorithms involved to solve the given problem

Q: could having an available program to solve these problems be useful in teaching discrete mathematics and if so why?

A: Yes. Will help in checking answers to questions

Q: Do you believe the program should have text versions of the algorithms and examples available to learn from?

A: Yes, as the algorithms are the most important part in answering the questions

Q: Have you ever tried to use a program similar to the one described?

A: Yes, but it was limited

Q: Which sections of Discrete Mathematics would be best to have a solver available for

A: Travelling Salesman Problem, Game Theory

Q: If not, why haven’t you needed one? If so, could you please tell me what you have used before?

A: \*No answer given\*

From this interview I found that my client is very interested in this product and believes will be useful in completing its desired task of helping teach Discrete Mathematics. This has also let me know how the program must not only just answer the questions but give either a step-by-step or access to the algorithms for the students. As well as this I have a set of solvers which my clients want me to make. Game theory and Linear programming link nicely and are both covered using the simplex algorithm.

## Algorithms

As stated in the Problem Background, the Problems I am looking to solve with this programme are:

* Linear Programming problems (Using simplex algorithm)
* Critical Path Analysis – finding earliest start and latest finish time.
* Minimum spanning Tree – Kruskal’s and Prims
* Chinese Postman (Route Inspection) Problem
* Travelling Salesman Problem – finding upper and lower bounds for specified starting node

### Linear Programming (Simplex Algorithm):

This Algorithm is quite complex and has quite a few steps. The algorithm is used to find the optimal constraints for a set of given inequalities.

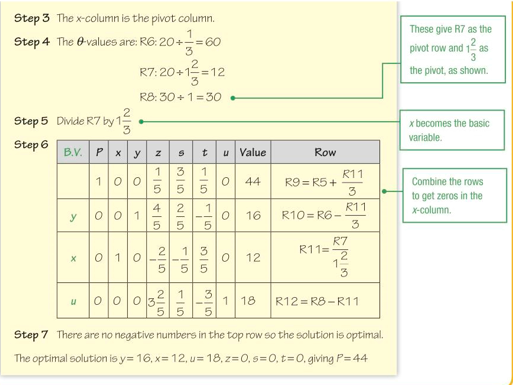
The algorithm given in the AQA Further Maths text book is as follows:



(30.1 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

And an example of using this Algorithm:





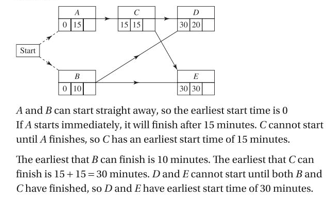
(30.1 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

### Critical Path Analysis

This algorithm is used to find the minimum time required for a set of activities to finish. This process also gives you the earliest time each individual activity can start and the latest time it is allowed to finish.

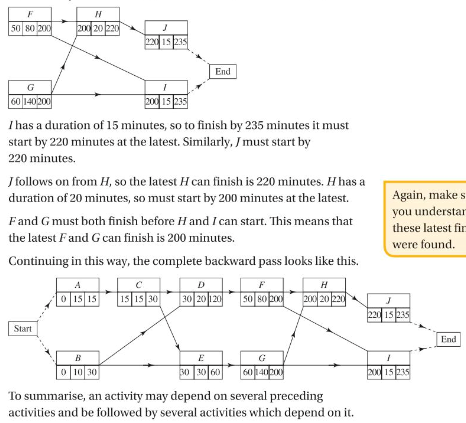
This process consists of 2 stages. The forward pass and the backward pass.

Below is a brief explanation on how the forward pass works.

  
(13.1 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

This algorithm works by looking at an activity’s (from Start to End) dependencies and finding the dependency whose earliest start time + duration is the largest. That is then this activity’s earliest start time.

Next is the backward pass shown below.

  
(13.1 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

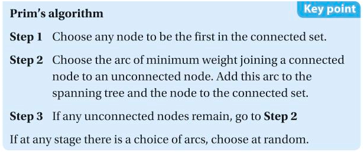
Briefly the backward pass looks at an activity’s (From End to Start) following activities. It finds the minimum of their latest finish - duration to work out its latest finish time.

### Minimum Spanning Tree (Kruskal’s)

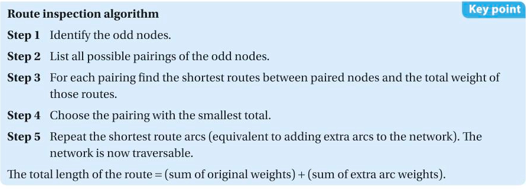
A minimum spanning tree is the least weighted set of connections between arcs so that you can travel to any node on the graph. For example if 5 towns want to be connected so that you could get to all the towns by road but they wanted to build as little road as possible, this algorithm could be used.  
  
(12.3 AQA A Level Further Maths Year1 + Year 2, Kerboodle, 2019)

### Minimum Spanning Tree (Prim’s)

This is another algorithm used to find a minimum spanning tree. This algorithm is more suited for computerization as it can be formatted as a matrix which is much easier to use to solve this problem then as a graph.

  
(12.4 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

### Chinese Postman (Route Inspection) Problem

This problem is used to find the shortest route where every arc in a network is traversed at least once and returns to the starting point.   
(12.5 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

This algorithm works by making a non-eularian graph into an Eulerian graph (an Eulerian graph is a graph whit an Eulerian cycle which is a Trail that traverses every arc exactly once and returns to the starting position). This is why odd degree nodes are identified as for a graph to be eulerian there must be no vertices with odd degree. The algorithm then just repeats some arcs so all vertices have even degree.

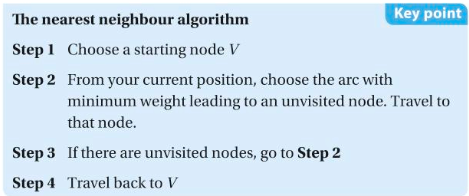
### Travelling Salesman Problem

This problem requires each node on a graph to be visited and to return to the start node in the minimum distance. This problem is extremely hard for computers to solve by brute force as for N nodes there are (N-1)! Solutions. Which means it is extremely inefficient to solve by brute force. So instead we will combine 2 algorithms to find a suitable solution (not necessarily the best solution).

Solving this problem requires finding an Upper Bound and Lower Bound. The optimal Solution will have a total weight in between these 2 values.

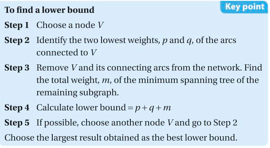
#### Upper Bound

To find an Upper bound an algorithm called “The Neatest Neighbour algorithm” is used. All this algorithm does is finds a Tour that fits the criteria for the travelling salesman problem. Any known tour can be treated as an upper bound.

  
(12.6 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

#### Lower Bound

To find a lower bound you complete the following algorithm



It is to be noted that if a lower bound forms a tour, then that tour is the optimal solution. This could be useful in finding the optimal solution.

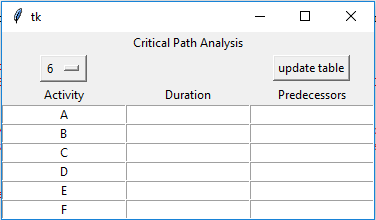
## Graphical User Interface (GUI)

The GUI of a programme is necessary to ensure understanding and usability of the program. My clients may not be well skilled when it comes to using computers. This rules out any chance of using a CLI (Command Line Interface) as the user will find a lot of difficulty understanding how to use the programme.

I have narrowed down my options for a GUI between Tkinter and Pygame. Both have their benefits.

### Tkinter

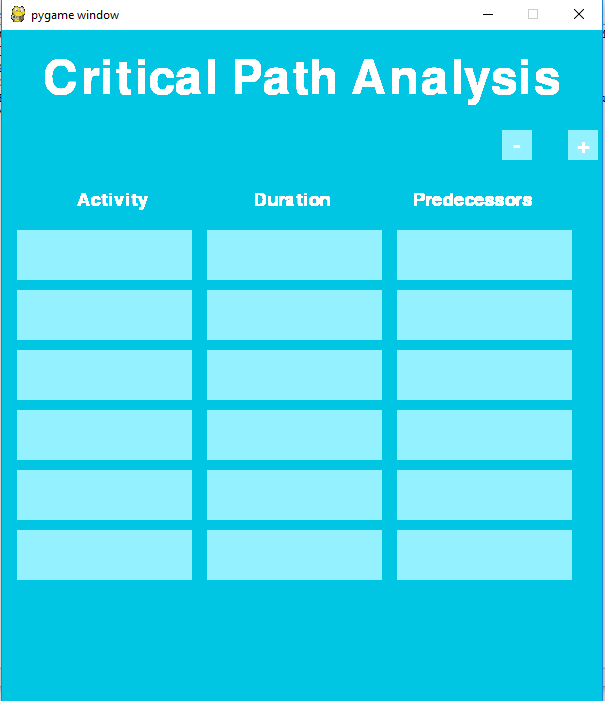
Tkinter has a very tidy design and has many useful features such as drop-down boxes, text input boxes and buttons all readily available. All of these can be neatly packed next to each other leaving a sharp and easy to use GUI for the User.



This would be a possible design for the Critical Path Analysis section of the programme. It has a nice layout which I believe is easy to understand and use. My clients had the following opinions on this layout:

### Pygame

Pygame can make the GUI more unique and individual. There’s a lot more that can be changed. However, there is no built in Button functions or input functions. So, these would have to be coded in manually.



This is a basic pygame display idea for the critical path analysis problem. It has a lot less formal tone compared to the Tkinter version. However, can seem harder and less obvious how to use.

### Clients Preference

After showing my clients the 2 GUI options, it was quite clear which they preferred. They found the Tkinter version of the GUI was much nicer due to the formal neat layout.

* “The Tkinter version seems a lot more practical”
* “The pygame version looks too childish”
* “The tkinter version looks more technical which fits better with the theme of a mathematics solver”

## Solution Objectives

### Necessary Objectives

1. Correctly calculate and solve the following problems:
   1. Linear Programming
   2. Critical Path Analysis
   3. Minimum Spanning Trees (Kruskal’s and prim’s)
      1. The solver will just use Prim’s algorithm as it is best suited for computerisation
      2. However, the text version of both algorithms will be stored to look over
   4. Chinese Postman (Route Inspection) Problem
2. Use TKinter to create a GUI for the project including (Decided by client):
   1. A login Menu
   2. Admin Menu – Links to multiple different Admin related screens such as:
      1. Class creation screen
      2. Admin creation screen
      3. Class editor screen
      4. User creation screen
      5. User viewer screen
   3. A Main Menu
      1. Labelled buttons for the user to direct themselves to the correct window to solve their problem
      2. A title of the Programme
   4. Individual windows for each problem (opened when selected from main menu)
      1. Has the correct input boxes for the specified problem
      2. Displays the answers in this new window
      3. Alternate option to attempt to answer questions
3. A Logon system for users and Administrators
   1. Only Admin accounts can create or delete user and admin accounts
   2. All users must be unique in order to be stored in a database
   3. For users to get an account their admin must give them their Username and temporary password.
   4. User has the ability to change passwords on first log in.
   5. Admins can change a user’s password in case they forget it
   6. All passwords are encrypted to ensure security
      1. Encryption includes a salt to increase security against rainbow tables
4. A Database
   1. Must be normalised to reduce redundant data
   2. Must include the following data
      1. Usernames
      2. Passwords
      3. classes
      4. Available Solvers
      5. Success rates
      6. Have they logged on before?
      7. Are they An Admin
      8. Recently Asked Questions
   3. Use SQL database queries to access needed data from the database
5. Ability for admins to lock certain aspects of the program to possibly stop students from cheating on homework.
   1. Can lock Individual classes from certain solvers.
6. Algorithms used wrote out in English for students to study
   1. Stored as Image files and just displayed in a Tkinter window
   2. Must be understandable to students on the AQA A-Level Further Mathematics course
7. Answer mode
   1. Student can put in the question as well as their answer
   2. Success rate on each problem is stored in a database
   3. Admins can see success rates of all users they have created
   4. Programme will remember the last 10 questions asked by the user so they cannot cheat on the answer section
8. Ability for users to change their password if they have forgotten it.
   1. Will need to ask Admin to change password.
9. Travelling Salesman Problem

### Unattainable Objectives

1. Interactive step-by-step guide. Where the programme prompts what needs to happen at this step of the algorithm but the user has to actually work out what to do.
2. Multiple working input methods for some/all of the problems.
3. Step-By-Step of how the solution was reached
   1. Must be concise and correct.
   2. Must be in a format which will be followable by an AQA A-Level Further Maths student.
4. Use networking to allow for users to access the program on their own computers.
   1. Have the database centralised on a server and all Admins/Clients will access the database remotely to run the program

# Documented Design

## Algorithms (continued)

### Login Screen

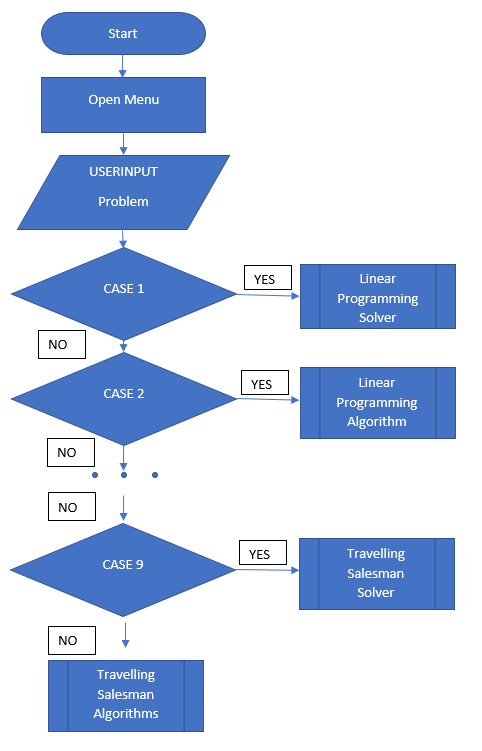
Login screen functionality is rather intuitive. The user inputs their username and password and then selects the Login button. The programme first checks that neither fields are empty and if so shows a new screen explaining the problem. If the username and password inputted are in the database under the same User the user is logged on to their account. Otherwise a message saying “Either the username or password entered was incorrect.”

### Admin Screen

The admin screen will allow the admin to select one of their classes to make changes or just view their progress. The initial admin screen will just consist of a dropdown menu and a button to open a specific class. The following window will hold information on the class; a set of tick boxes to change which solver is available and a button to create a new user for the class. A save button will also be on the screen to save these changes to the database.

The initial admin screen will also have a create class button which will open a screen to create a new class. In this screen the admin will need to input a class name. Also, on this screen, the class ID will be shown to the Admin.

### Main Menu



### Linear Programming

Table = USERINPUT #Inputted matrix

WHILE not solved #while no solution is found

Usable\_cols = [I, pos for pos, I in enumerate(Table[0]) if i<0]#state which cols have a c value<0

Pivot\_col = min(usable\_cols)[1] #finds the column used as the pivot

Usable\_rows = [I, pos for pos, I in enumerate(Table) if i[pivot\_col]>=0]#finds usable rows

Val\_div\_piv = [(Table[i][pivot\_col]/Table[i][END], i) for I in usable\_rows]div piv col by val

Pivot\_row = min(val\_div\_col)[1]#finds pivot row

Original\_row = Table[pivot\_row]

Table[pivot\_row] = Table[pivot\_row]/Table[pivot\_row][pivot\_col]#alters pivot row

FOR row in Table

divider = row[pivot\_col]/Original\_row[pivot\_col]

row = row – Original\_row\*divider #alters all other rows in table

ENDFOR

IF all values in Table[0]>=0 THEN #Condition needed to be met for a solved solution

Table is solved

ENDIF

ENDWHILE

### Critical Path Analysis

CLASS Activities #Activity data structure

SUB initialise(duration, predecessors, followers, name)

Self.duration = duration

Self.predecessors = predecessors

Self.followers = followers

Self.earliest\_start = 0

Self.latest\_finish = 0

Self.name = name #Class states

SUB forward\_pass(Activites)

FOR act in Activities: #for every activity

FOR fol in act.followers #for activities that follow the activity

If act.earliest\_start + act.duration > fol.earliest\_start THEN

#if the acts earliest start time + duration is greater than the followers earliest start time then

Fol.earliest\_start = act.earliest\_start + act.duration

#followers earliest start time is changed.

SUB backward\_pass(Activities)

Project\_finish = 0 #intialises project finish time as 0

FOR act in Activities:#loops through activities

IF act.earliest\_start+act.duration>Project\_finish THEN

#if the activity finishes after project finish time

Project\_finish = act.earliest\_start+act.duration

#alter project finish time

FOR act in activities:

Act.latest\_finish = Project\_finish#set all activities latest finish to project finish time

FOR act in reversed(activities):#loop through activities in reverse order

FOR pre in act.predecessors:#loop through activities predecessors

If act.latest\_finish-act.duration<pre.latest\_finsih THEN

pre.latest\_finish = act.latest\_finish-act\_duration

#if necessary alter predecessors latest finish

### Graph Problems

The first algorithm based decision to be made for the selection of graph problems is the way in which the graph is to be represented as a data structure. 2 feasible methods are an adjacency matrix or adjacency list.

Pros for adjacency matrix:

* Adding edges or testing for edges is simple and quick
* Convenient and easier to work with
* Will map well to the chosen input method for the graph problems (Shown in the GUI section of the documented design)

Pros for adjacency list:

* More space efficient

My final decision is to go with an adjacency matrix because there will not be big enough graphs in the Further Mathematics A-Level course for the space efficiency problem to occur.

#### Matrix

CLASS Matrix

SUB initialise(matrix)

Self.matrix = matrix

SUB complete\_matrix() #creates a completed matrix

New\_matrix = [[0 for \_ in range(len(self.matrix))] for \_ in range(len(self.matrix))] #makes a 0 matrix

For I in range(len(matrix)) #for every row it finds the shortest distance to all the other nodes

Distances = self.djikstra (i)

For j in range(i, len(matrix))

If distances[j] != 0 THEN

Matrix[i][j] = distances[j]

Matrix[j][i] = distances[j]

ELSE

Matrix[i][i] = -1

SUB min\_distance (dist, sptSet) #finds minimum distance

Min = \*Large Number\*

For v in range(len(self.matrix))

If dist[v] < Min and sptSet[v] == False THEN

Min = dist[v]

Min\_index = v

SUB djikstra\_algorithm(start)

Distance = [\*Large Number\* for \_ in range(len(self.matrix))]

Distance[start] = 0

sptSet = [False for \_ in range(len(self.matrix))]

FOR \_ in range(len(self.matrix))

U = self.minimum\_distance(distance, sptSet)

sptSet[u] = True

FOR v in range(len(self.matrix))

If self.matrix[u][v] > 0 and sptSet[v] == FALSE and distance[v] > distance[u] + self.matrix[u][v] THEN

Distance[v] = distance[u] + self.matrix[u][v]

RETURN distance

SUB remove\_node(c\_row)

For row in self.matrix

Row.pop(c\_row)

Self.matrix.pop(c\_row)

#### Dijkstra’s Algorithm

Dijkstra’s algorithm is a well-known and well optimised algorithm. I have decided to take an online version of the algorithm to implement in my program due to the previously stated fact.

SUB min\_distance (dist, sptSet)

Min = \*Large Number\*

For v in range(len(self.matrix))

If dist[v] < Min and sptSet[v] == False THEN

Min = dist[v]

Min\_index = v

SUB dijkstra\_algorithm(start)

Distance = [\*Large Number\* for \_ in range(len(self.matrix))]

Distance[start] = 0

sptSet = [False for \_ in range(len(self.matrix))]

FOR \_ in range(len(self.matrix))

U = self.minimum\_distance(distance, sptSet)

sptSet[u] = True

FOR v in range(len(self.matrix))

If self.matrix[u][v] > 0 and sptSet[v] == FALSE and distance[v] > distance[u] + self.matrix[u][v] THEN

Distance[v] = distance[u] + self.matrix[u][v]

#### Minimum Spanning Tree

Matrix = USERINPUT #NxN matrix which is symmetrical by the diagonal an empty cell is declared by a -1

Weight = 0

Cols\_completed = [0] #starts at 0 as we will start with the first row

WHILE len(Cols \_completed) < len(Matrix) #While unattached nodes

FOR I in Cols \_completed #loop through currently connected nodes

Row = matrix[i]

Usable\_row\_values = [(num, pos) for pos, num in enumerate(row) if num>0 and pos not in self.rows\_got]

IF usable\_row\_values not empty

Next\_value = min(usable\_row\_values) #Finds minimum value in rows of currently connected nodes

TRY

IF Next\_value[0] < Min\_value[0] THEN

Min\_value = Next\_value

ENDIF

EXCEPT

Min\_value = Next\_value

ENDIF

Cols\_completed.append(Min\_value[1]) #adds the just completed row to the list

Weight += Min\_value[0] #increases the total of the weight

ENDFOR

ENDWHILE

#### Chinese Postman

CLASS Network(Matrix)

SUB initialise(matrix)

Matrix.initialise(matrix)

Self.completed\_matrix = self.complete\_matrix

SUB ri()#This function is called to solve the problem

Odd\_rows = self.get\_odd\_row\_indexes()

Dist = self.min\_distance(odd\_rows)

Dist += self.total\_distance()

RETURN dist

SUB get\_odd\_row\_indexes()

Rows = []

For index, row in enumerate(self.matrix)

Counter = 0

For val in row

If val>0 THEN

Counter += 1

If counter%2 != 0 THEN

Rows.append(index)

RETURN rows

SUB total\_distance()

Total = 0

For row in self.matrix

For value in row

If value>0

Total += value

Return int(total/2)

SUB min\_distance(rows)#rewrote over the Matrix definition

Perms = list(all\_pairs(rows))#all\_pairs is a generator that creates all possible pair of node connections

Min\_dist = \*Large Number\*

For perm in perms

Dist = 0

For pair in perm

Dist += self.completed\_matrix[pair[0]][pair[1]]

IF dist<min\_dist THEN

Min\_dist = dist

RETURN min\_dist

#### Travelling Salesman Problem

##### Upper bound

Matrix = complete\_matrix(USERINPUT)

Start\_node = USERINPUT

Node = start\_node

Weight = 0

Unvisited\_nodes = list(\*node indexes\*)

WHILE there are unvisited\_nodes

If there are more than 2 unvisited nodes THEN

Remove node from unvisited\_nodes

Adjacent\_nodes = [(val, i) for i, val in enumerate(self.matrix[node]) if i>0 and i in unvisited\_nodes]

Next\_node = min(adjacent\_nodes)[1]

Weight += min(adjacent\_nodes)[0]

Node = next\_node

ELSE

Unvisited\_nodes.remove(node)

Weight += self. matrix[node][unvisited\_nodes[0]]

Node = unvisited\_nodes[0]

Unvisited\_nodes.remove(node)

Weight += self. matrix[node][start]

Return weight

##### Lower bound

Matrix = USERINPUT

Node = USERINPUT

Values = [I for I in self.matrix[node] if I != 1]

Values.sort()

P=values[0]

Q=values[1]

New\_matrix = remove\_row(node)

Weight = Minimum\_spanning\_tree(new\_matrix)

Return p+q+weight

### Answering Questions

When answering a question in the “Test Answer” section of a problem. The code will use the above algorithms to work out the answer and then test that answer against the answer inputted by the User.

## Data Structures

### Matrix

Matrices are a data structure similar to 2D arrays. However, in this program we need a Matrix data structure as we need more specialised methods than just a 2D array. We need a method to create the nearest distance matrix from the given matrix and we need to be able to remove specified rows and their corresponding columns. This will be a self-defined Data Structure.

### Array

Arrays are used in many locations in the program.  
One use is a 2D array is used to store the value in the matrices used in graph problems.  
arrays are used when generating the StringVar’s for the tkinter entry boxes so the Entries can be identified easily as a matrix

### Activity

An activity is used in Critical Path Analysis. It holds information regarding the activity’s duration, its predecessors and in this case its followers too. This information is then used by the activity network to solve the problem.

### Activity Network

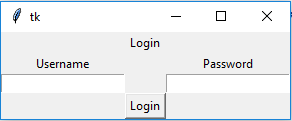
An Activity Network is used in Critical Path Analysis and represents the way in which the activities in the problem are linked. In this program we are representing the Activity network as a Precedence table (Explained in the Analysis). This will be a self-designed data structure.

### Dictionary

Dictionaries will be used to store results from the Linear Programming section of the program. The key will be the variable being represented (e.g. x1, x2, Cost, etc…) and the value paired with it will be the value of that variable which optimises the inputted problem.

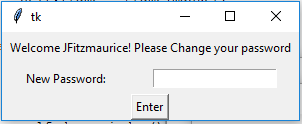
## GUI (continued)

### Login Screen



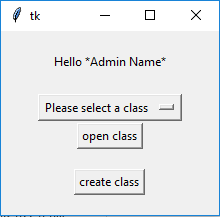
This is a basic design of a log in menu. There is no separate log on menu for a normal user or an admin. Both tables will be searched through in the Database when a password is inputted. A login menu is needed as Different users need to be identified when using the program.

### First Use Screen for Users



This screen is for users to change their temporary password given by their admin to something more memorable for them.

### Admin Screen

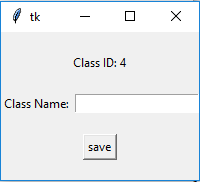


Very simple design. All the initial admin screen is required for is to open one of the admins classes. When a Class is opened a new window is shown showing the details of the class, including: users and their details; which solvers are accessible to the class and the overall success rates of the class. A create class button is also available for the admin to create a new class. This window design is shown below. This screen is needed for the Admin to search through their classes.

### Class Creation Screen

This screen will just display the class id of the new class created. Then an input box is given for the Admin to input what they want the class to be called.

This screen is necessary as when an Admin creates a new class they will need to give it a name by which they will recognise the class.



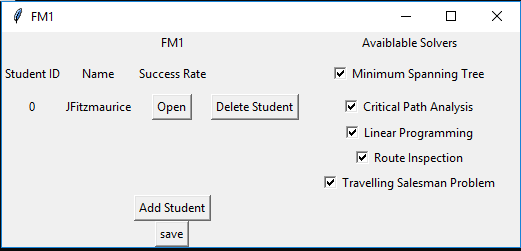
### Class Editor Screen

This screen will show in a list all the students in the class along with their student ID and success rate. A “Delete student” button will also be there to delete individual students from a class. When this button is clicked an “are you sure?” prompt will be shown to ensure no accidental removal of students.

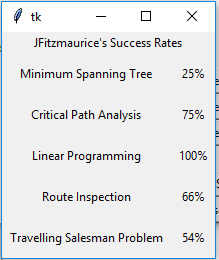
In this screen is also a selection of tick boxes to determine which solvers are available for the class. When the box is ticked the solver is available for the students.

There is also an “Add student” button. This button is used to open a new window where a new student can be added to the class.

This screen is necessary for admins to evaluate the success of their students in each class and to also edit the class.



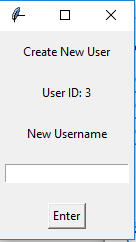
### Success Rate Viewer



The purpose of this screen is to show to the admins the success rate of the selected user in their class. The success rates are retrieved from the database then displayed as shown above.

I like the neat formal layout that the above design brings

### User Creation Screen



This screen is used for the admin to create a new user. They input the new username in the entry box. The screen also shows this new users ID

### New User Password Display

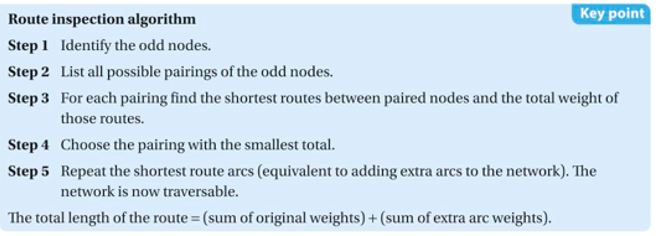
Very simple design just a new window that display the password for the Admin to write down or show the User so they can log in.

### Algorithm Window

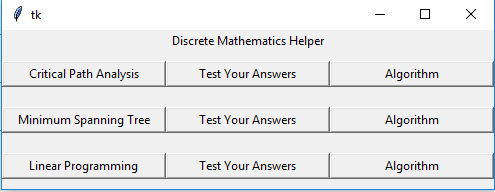
Algorithm window will just consist of an image which shows the steps to how to solve the problem by hand.

This window is necessary as the program (as asked by the client) needs to have the written algorithms available for the users to learn from them.

Example: (take note the images used won’t be from the Kerboodle e-book)

 (12.5 AQA A Level Further Maths Year 1 + Year 2, Kerboodle, 2019)

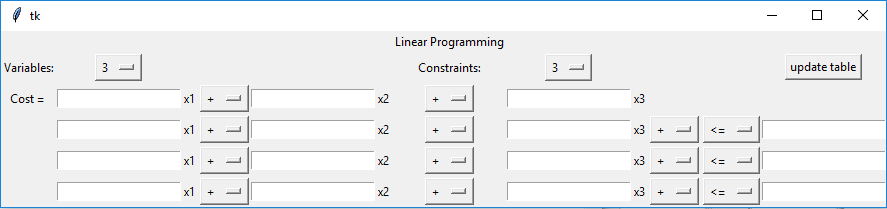
### Main Menu



This is the current design setup for the Main Menu. This design only shows 3/5 of the algorithms but it shows how the design will work. Each Algorithm has its own row and has 3 options: Solver, Answer tester and Algorithm. The solver (Labelled above as the name of the algorithm) opens the solver; The Answer tester (Labelled above as “Test Your Answers”) allows the user to answer a question and have it checked by the code. The algorithm Button opens a new window showing an image of the Algorithm in text form for the users to learn.

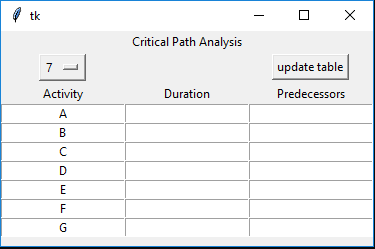
I have decided to include a Main Menu to make navigating the program easy for the user. This is beneficial as the users shouldn’t need to know in detail how to use the program it should be intuitive for further mathematicians to use.

### Linear Programming

Above is the design idea for the Linear Programming section of the programme. The user can set the number of Variables required and constraints required to suit their particular problem. After that they can select update table to make the table required. The first line of inputs (For the actual problem) is the cost function. This is the function in which the programme wants to optimise. The following lines of entry are for the user to enter the constraints in which the values of x1, x2, x3… need to fall in.

I have chosen to show this window in this way as it is familiar to how the Further Maths students will actually write out their answers to a question like this. When the students are given a question they first write out all of the inequalities they can get from the question as well as the cost function (or Objective function). This gives them the question in the format required for this programme. This is why I chose to have the inputs set up as shown above.

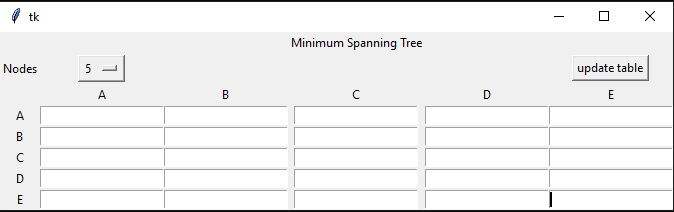
### Critical Path Analysis



Above is the design idea for the Critical Path Analysis section of the programme. Similarly, to the Linear Programming section the user can select the number of Activities required for their problem. Also show in the table is that the Activities are automatically labelled. The user then inputs the Duration of each activity and the predecessors of each activity.

I have chosen to design this window as shown above because this is a much more convenient way for the users to input the question. Most questions in the Further Maths A-Level course have the questions given in the format above. This makes it extremely easy for the user to copy in the question they want to check. One flaw however is that they may insert the predecessors in incorrectly.

### Minimum Spanning Tree, Chinese Postman, Travelling Salesman problem



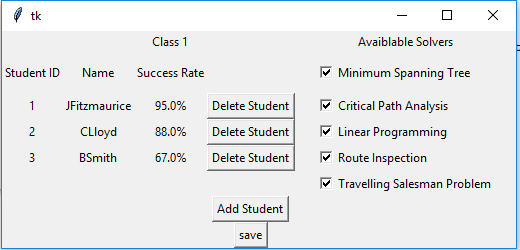
All 3 of these problems will have the same GUI interface. This is because all of them will be stored as a minimum distance matrix. Differences will include the Title of the window. Other possible changes would to make only half of the inputs usable. This will reduce the possibility of human error when inputting values as for any of these problems to be solved there must be a reflection across the diagonal. For example, the value in row A column C must match the value in row C column A.

I have chosen to design the window as shown above because it is easy for Further Maths students to understand. It also directly links in to how the data is going to be stored (distance Matrix).

### Overall Clients thoughts on GUI

* “Easy to the eye and easy to use.” – Karen Tunnicliff
* “Nice, neat and formal layout” – Helen Morgan
* “Could possibly be a little colourful” – Helen Morgan
* “Tick boxes should be in line with each other” – Helen Morgan

### Changes after client feedback

* The tick boxes in the admin screen has been changed to look as follows.

## Password Storing

### Problem 1

Due to the data protection act secure information such as passwords must be encrypted in such a way that if anyone were to access the database they wouldn’t be able to access the actual passwords of the user

### Solution

#### Storing

When passwords are stored into the database they are first hashed using a hashing algorithm. This hashing algorithm will cause the inputted password to become a seemingly random selection of characters. This hashed password is stored into the database

#### Testing

When testing whether an inputted password is correct the inputted password is hashed with the same hashing algorithm and is tested against the hashed password to the user with the same Username inputted.

This solution means that if someone accessed the database the passwords they would see would not be the correct passwords for each user.

### Problem 2

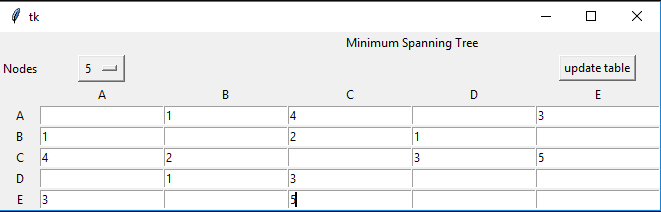
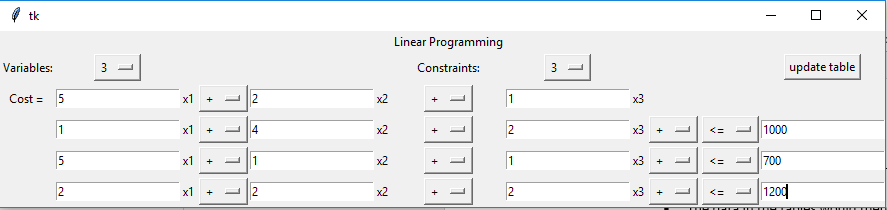
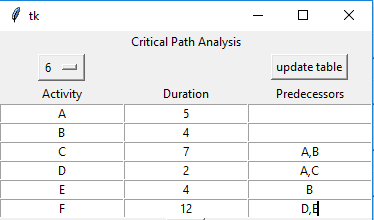
Even with hashing passwords are still relatively insecure. This is due to a rainbow tables which can be used to quite easily and efficiently solve hashes.

### Solution

Adding salts to the password. Before the password is hashed a randomly generated salt is appended to the end of the password. For the best security each salt should be different. Salts make using Rainbow tables useless as chances are a user’s password + the salt will not be a known and documented hashed password in the rainbow table. This means slower methods such as brute force will be required to work out the password. This makes the passwords being stored far more secure. The salt for each password will be stored in plaintext in the database along with the hashed password. To test a password the salt is appended onto the password, then hashed and finally compared to the saved hash in the database.

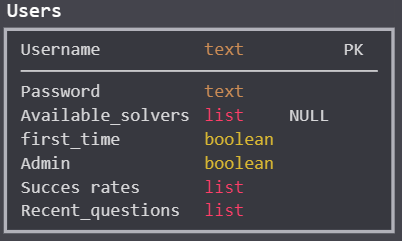
## Database

### Data Required

* Username
* Password
* Available solvers
* Logged on before?
* Admin?
* Success rates
* Recently asked questions
  + Questions are to be stored as follows
    - As a string
    - First 3 characters represent which problem type the question is (Minimum Spanning Tree would be “MST”)
    - The data in the tables would then all be appended to each other and split by “/” to represent the data in each box the question.
      * For Linear Programming questions a key character “C” is used to represent the start of the cost function and a key character “D” is used to represent the data in the constraints
      * Any empty spaces (Which have been allowed by the code i.e. the box in question is allowed to be empty) will be noted by a “0”
      * FOR Critical Path Analysis the data will be stored row by row with the characters in the Predecessors column being separated into their individual activities. E.g. if predecessors = A,B,C it will be split as such A/B/C
      * No predecessors for Critical Path Analysis will be denoted by “-”
  + Storing the data like this will ensure each question is unique so can be used to test whether a question asked is the same as a recently asked question.
  + Example 1:  
      
    This input for a Minimum Spanning Tree problem and the question in text form would look as follows:  
    “MST/0/1/4/0/3/1/0/2/1/0/4/2/0/3/5/0/1/3/0/0/3/0/5/0/0”
  + Example 2:  
      
    This is for a Linear Programming problem and the question in text form would look as follows:  
    “LPRC/5/2/1D/1/4/2/1000/5/1/1/700/2/2/2/1200”
  + Example 3:  
      
    This is for a Critical Path Analysis Problem and the text form to be stored in the database will be as follows:  
    “CPA/A/5/-/B/4/-/C/7/A/B/D/2/A/C/E/4/B/F/12/D/E”

### Normalisation

#### No normal form

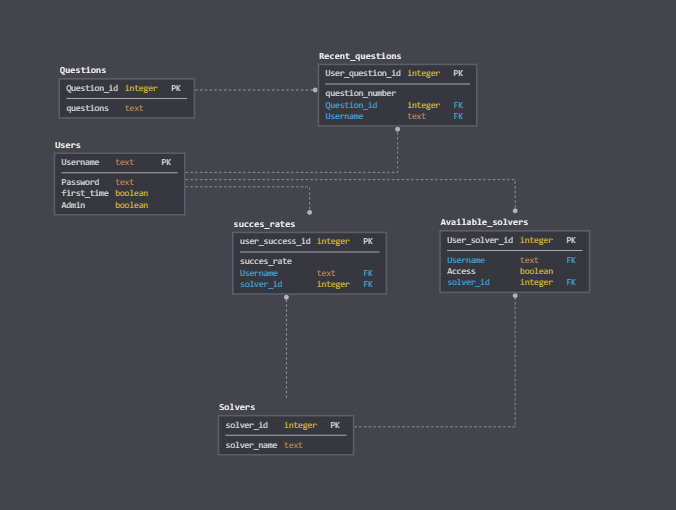


(sqlDBM, 2019)

#### First Normal Form

First normal form requires that each column in the table must be atomic, contain only a single value. So, in the table above has 3 columns that don’t abide by this rule so the table will have to be split so it does. Also, if any many-to-many relationships are generated these should be corrected by creating a link table

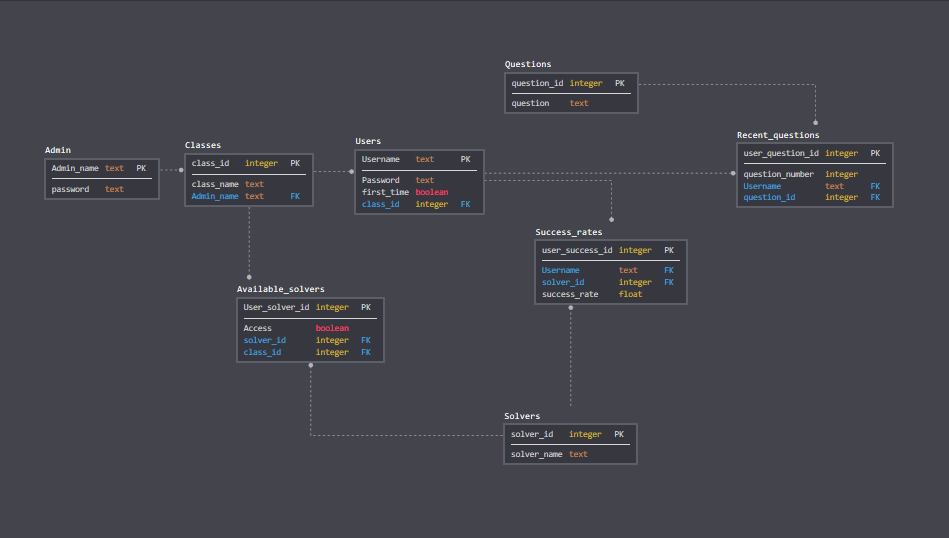
Tables needed to be split include Available\_solvers, success rates and recent\_questions. This is done by creating new tables for all these columns. This however will cause a many-to-many relationship as many users will have access to many solvers. To counteract this, we need a link table.

 (sqlDBM, 2019)

The above solution puts everything into 1st Normal Form.

**Problem**

Admins need to be able to create classes in which their students (users) will be in (a student can only be a part of 1 class). These classes decide which solvers are available to the users. This means a small altercation needs to be made to the database.

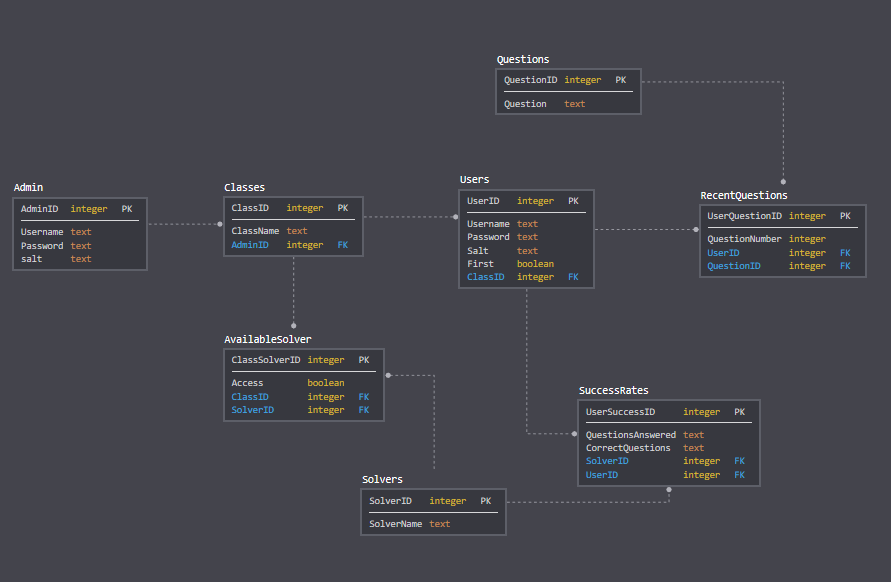
 (sqlDBM, 2019)

Above shows the altercations made. 2 new tables have been made (Admin and classes). This now allows for students to be a part of a class and the Admin can now lock certain solvers from the entirety of a class.

**Problem 1**

When updating success rates, we need to know how many questions they have attempted. This is so we can update the success rate accordingly. Because of this rather than storing a float of a “success\_rate” I am going to store Integers of “correct\_questions” and “questions\_answered”.

As well as this we need to store the salts in this database

 (sqlDBM, 2019)

#### Second Normal Form

For a table to be in 2nd Normal form there must be no Partial Dependencies and conform to 1st Normal Form. We already comply with 1st Normal form so now we must check for any partial dependencies

Partial dependency – Has no fields that aren’t dependant on the key.

There is no problem in the above database for Second Normal Form

#### Third Normal Form

For a table to be in 3rd Normal Form there must be no non-key dependencies and conform to 2nd Normal Form. We already comply to 2nd Normal form so no we must check for any non-key dependencies.

Non-key Dependency – No field is dependent on another field except the Primary key.

In “Available\_solvers” there is a non-key dependency. Access is dependent on the non-primary attributes “class\_id” and “solver\_id”. We need to resolve this dependency.

A similar Dependency can be noted in SuccessRates and RecentQuestions.

These non-key dependencies don’t have to be resolved. This is due to the fact the combination of ClassID and SolverID is unique for each primary key. E.g. if ClassSolverID was 1, ClassID was 1 and SolverID was 1. There would not be another ClassSolverID where ClassID was 1 and SolverID was 1. This means there won’t be any duplications of data so there is no requirement to resolve these tables into 3rd Normal Form.

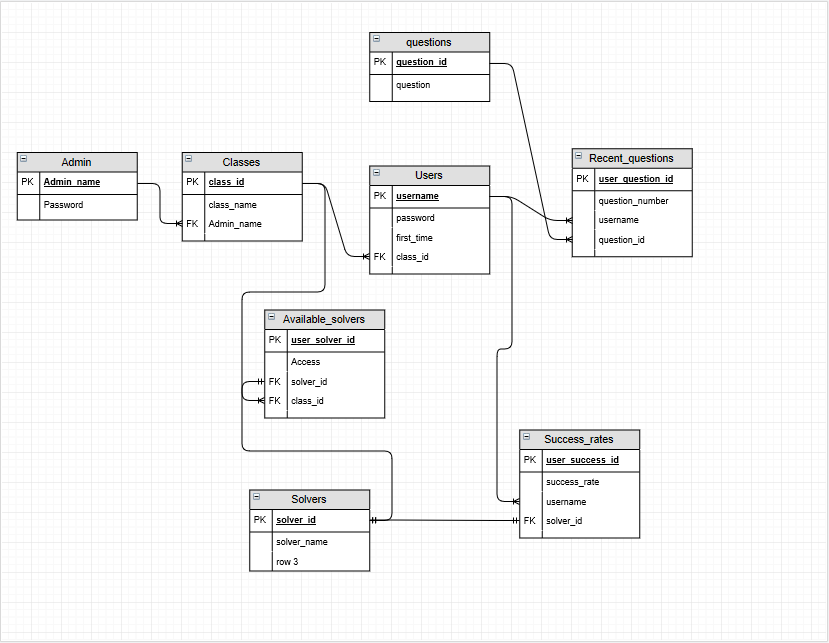
### Removing old Questions

The database only needs to store the 10 most recent questions for each user. To do this we will append new questions to the Database as usual up to and including the 10th question. But when the 11th question is added the question with the smallest question number (for that user) will first be deleted. But the question number for the new question will still be (in this case) 11.

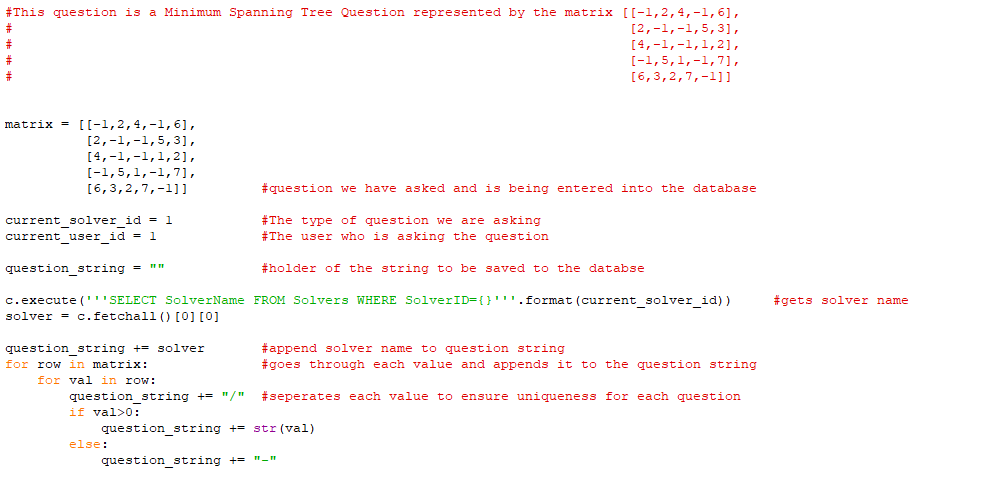
### Entity Relationship Diagram

Tables are:

* Users
* Classes
* Admins
* Recent\_questions
* Questions
* Success\_rates
* Solvers
* Available\_solvers

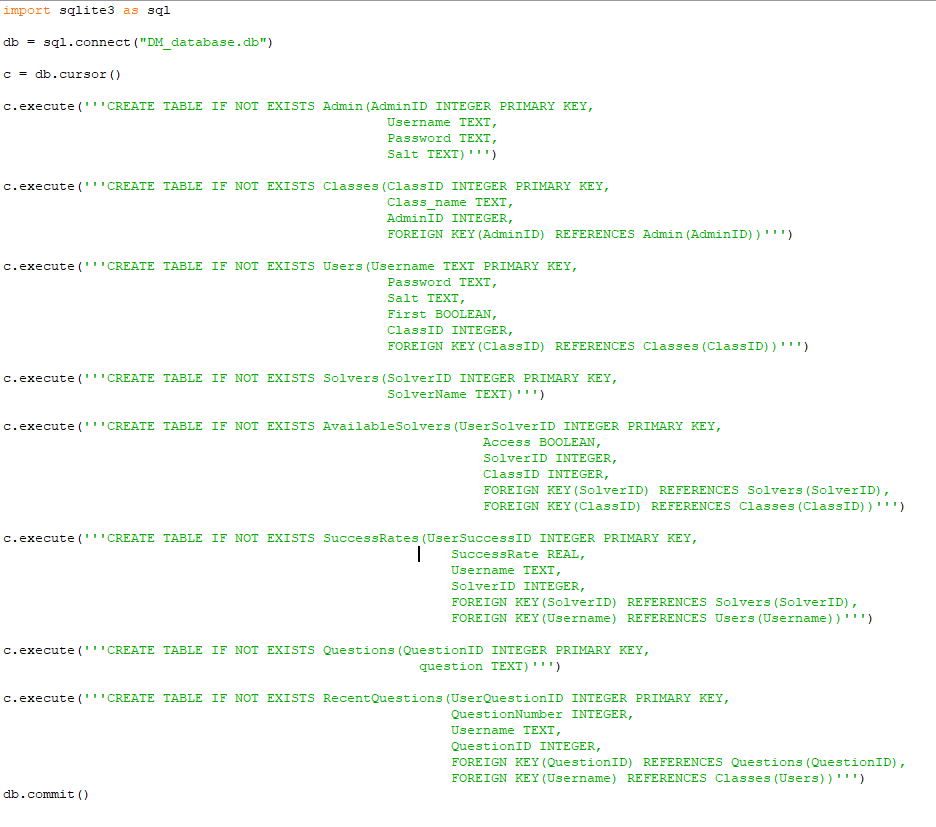


### Creating Question String

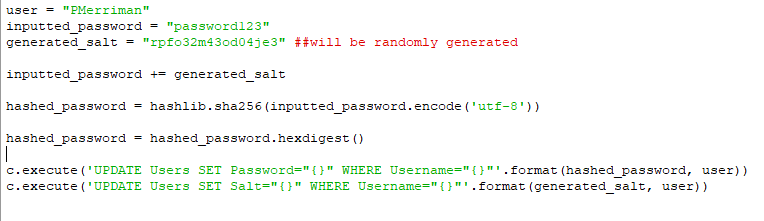


### SQL Statements

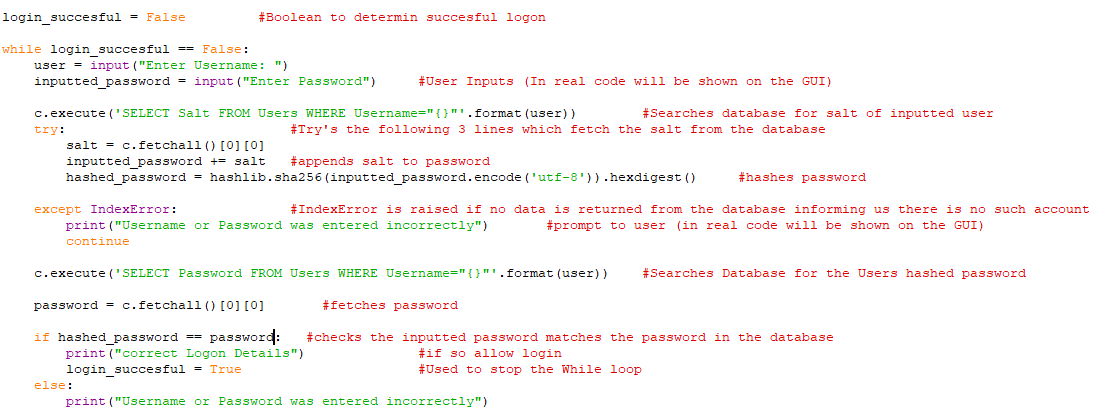
#### Creating Database



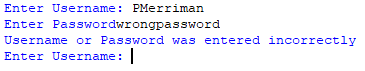
#### Editing Password for first time login



#### Logging in



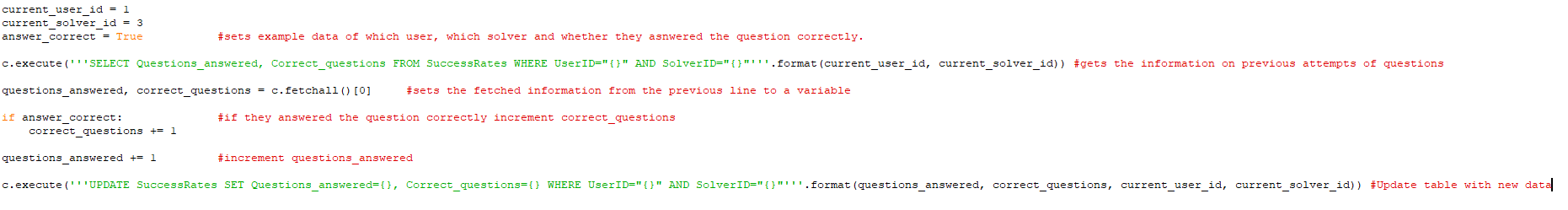
##### Example input of incorrect credentials



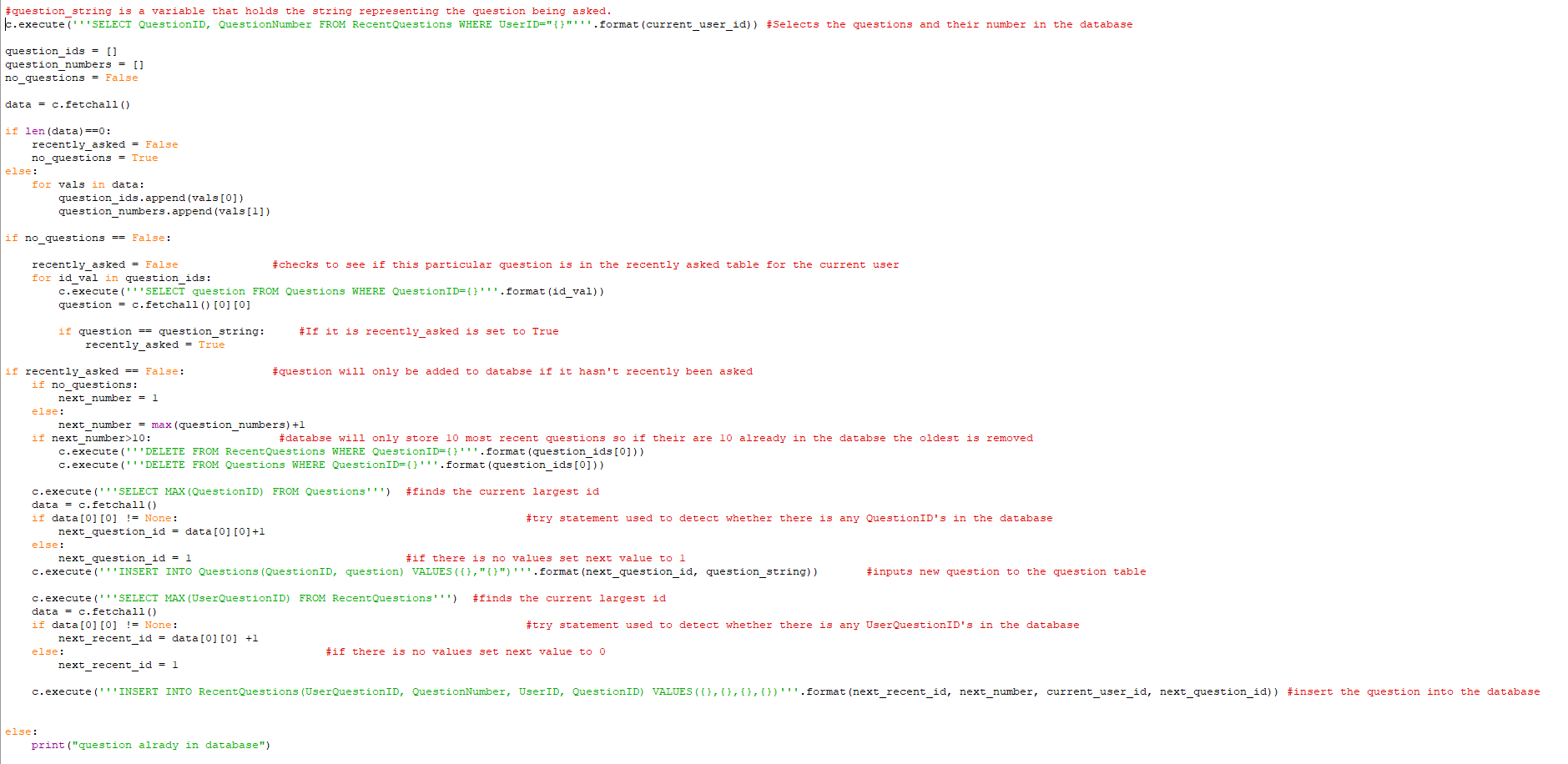
##### Example input of correct credentials



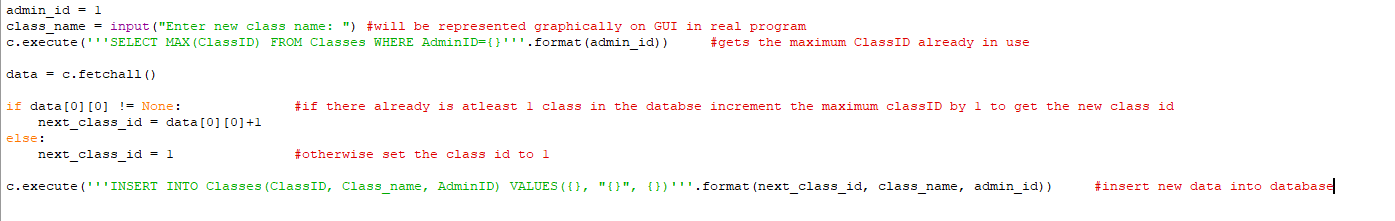
#### Updating success rate



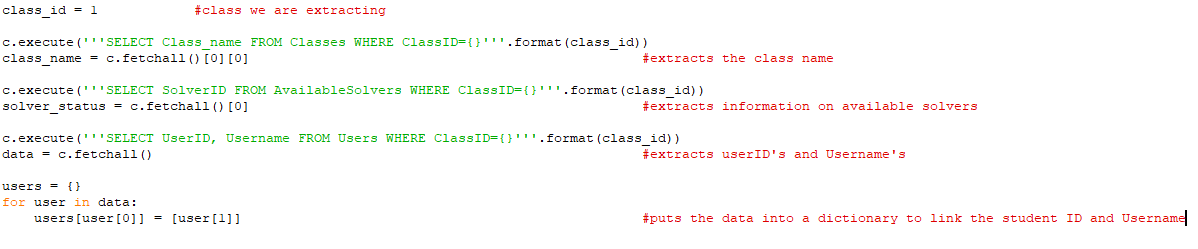
#### Adding new questions to database



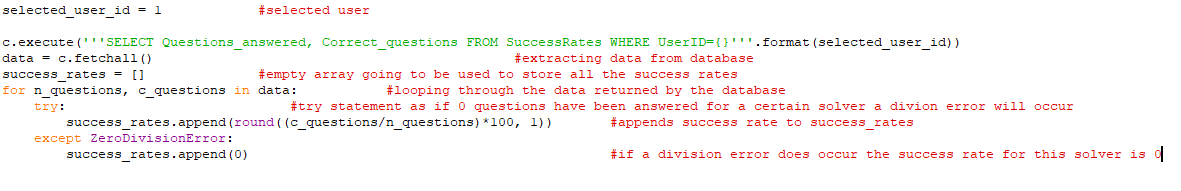
#### Creating new class



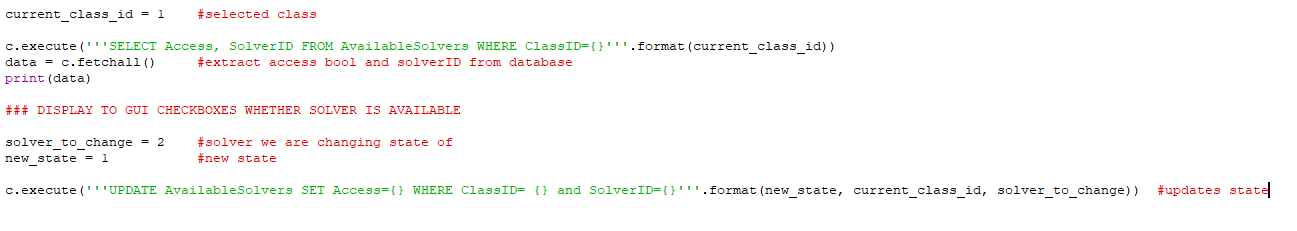
#### Opening class



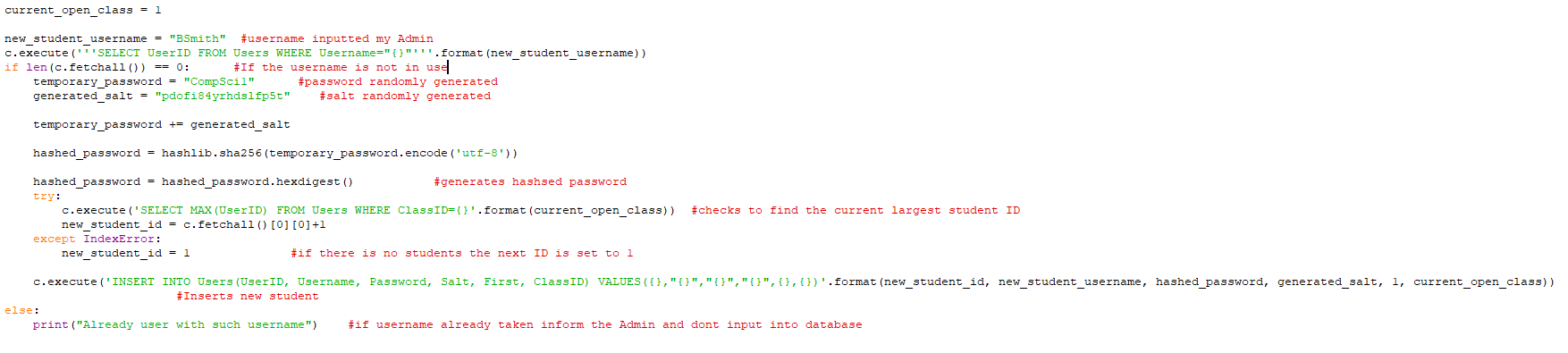
#### Viewing Success Rates



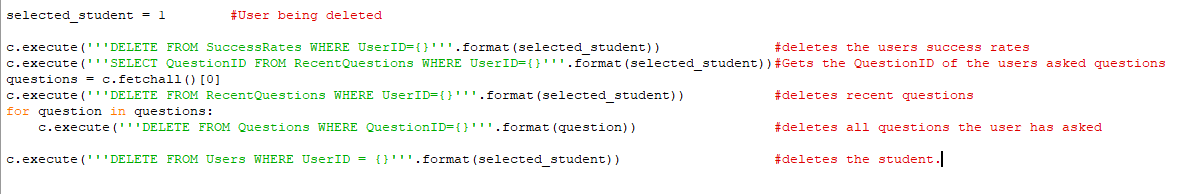
#### Changing available solvers



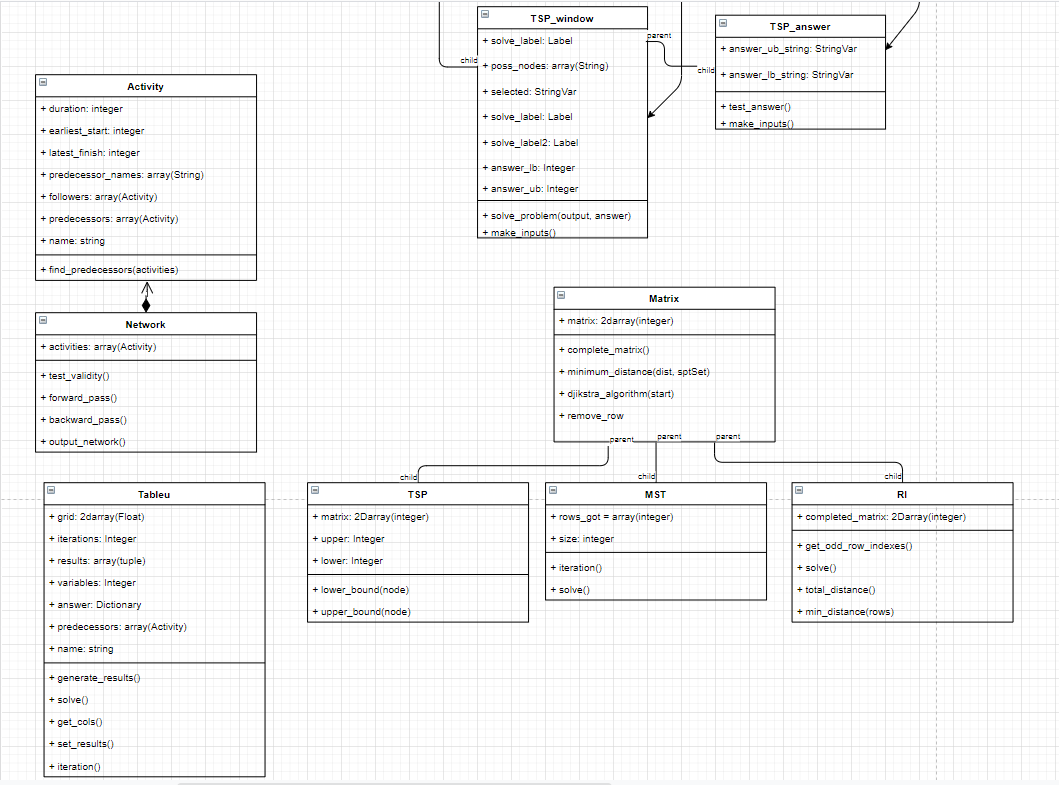
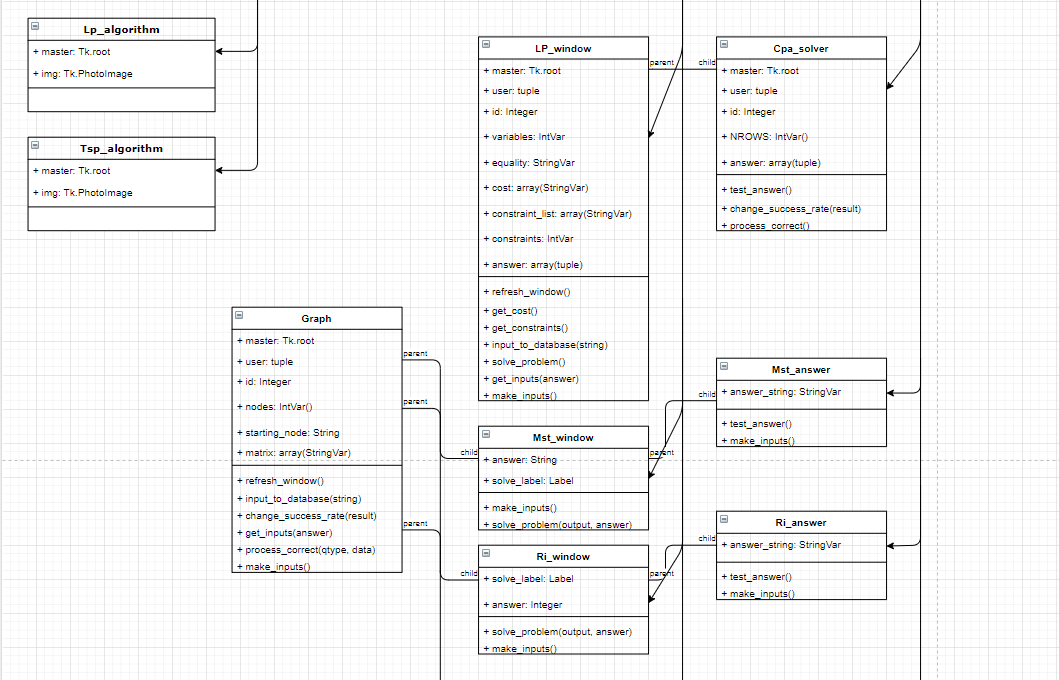
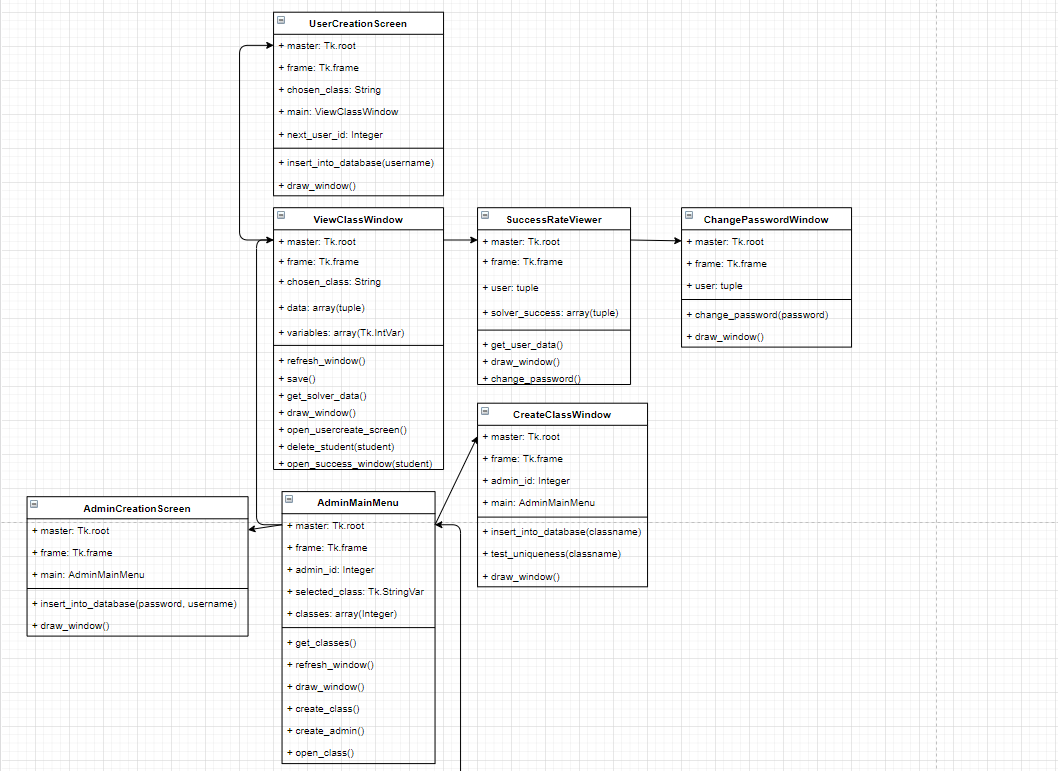
#### Adding new Student



#### Deleting Student



## Class Relationship diagram:



All unlabelled arrows represent that this class can be used to open a new window; links each window to another window which they can open.

## Overall Design

### Imports

#### Tkinter

In the analysis 2 GUI libraries were covered (Tkinter and Pygame). After a demo version of each was created my clients decided that they preferred the style of the Tkinter GUI. This is why Tkinter is being used for this project; It is so the Users of this product will be able to easily navigate the program.

#### Sys

Used to take advantage of its “maxsize” function which sets a variable to the largest integer possible.

#### Functools

Used for its “partial” function. This is needed as a tkinter button is allowed to have a command. But the command is not allowed any constraints. To avoid this problem, we can use the partial function. This function allows us to pass arguments with a command in a tkinter button.

#### Copy

Copying 2d arrays in python can be tricky. This is due to the fact when a 2D array is copied in python the individual elements in the array aren’t copied and pasted in a new memory location. Instead a memory reference is created. This memory reference links the new “copied” array to the original array through a memory reference. So, any changes to the new array will also occur in the old array.

To avoid this problem the program uses a function in the library copy called deepcopy. This function actually rewrites the 2d array as a new separate 2D array. This allows us to make changes to the new array without altering the first array.

#### SQLite3

Used to create, access and modify a database which holds all the data about users, admins, classes etc.

#### String

Used to access a string of all characters and letters to then be used to randomly generate a password. It was easier to import string and use its commands then to create an array and fill it with all possible characters

#### Random

Used to randomly generate passwords.

#### Hashlib

Used to hash users’ passwords (+salt) so that if anyone accesses the database they won’t be able to easily find out each users’ passwords.

#### Uuid

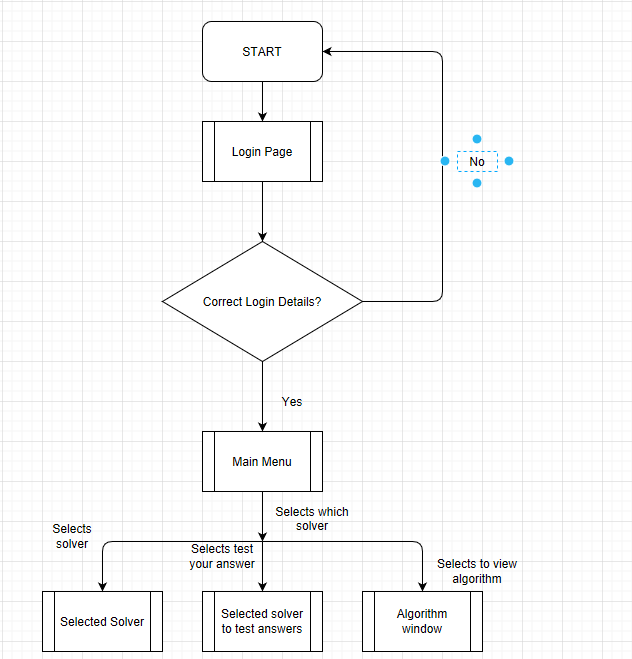
Used to generate a salt to be appended to the password

### First use of Program (Admin and User)

When someone purchases the software, they are given an Admin key and password. They can use this to log onto the programme as an Admin. From there the Admin can create classes and can create students for those classes. When students are created they are assigned temporary passwords which can be given to the students to sign onto their account. On the first log in of a Student account they are prompted to change their password. From there they can access the Main menu for all the solvers and Algorithms.

### How the GUI subsections mix

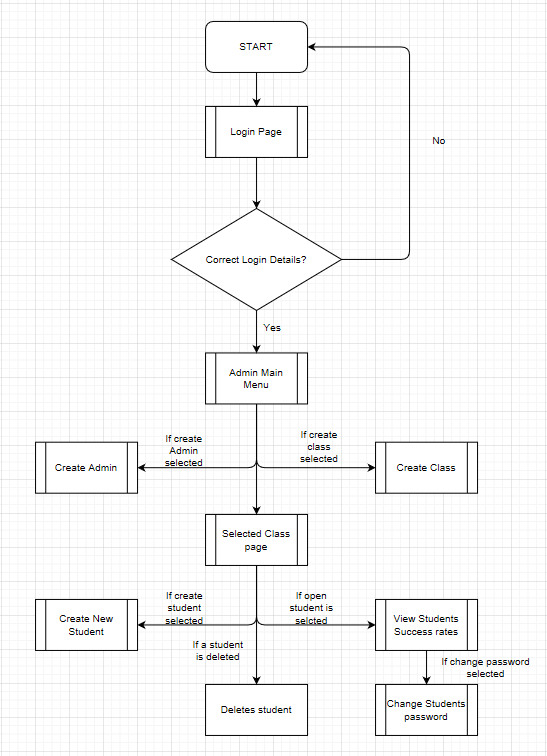
#### Users



As shown in the GUI-Main Menu section the main menu will have buttons linking to all the solvers, answer checkers and algorithms. Whichever Algorithm/answer checker/solver is selected will determine which new window is opened. Any algorithm that is currently Disabled by the Admin will still be shown but the Button will be disabled so will not be usable. Below is what a disabled button looks like (left is disabled, right is enabled).



#### Admins



As shown in the flowchart above. When admins log on they are shown the Admin main menu. If they choose to create a new Admin they are sent to the window to do so. If they choose to create a new class they are sent to the window to do so. If they select to open a class they are sent to the corresponding class’s page. In this page they can choose to create a new student which will send them to the window to do so. They can delete a student which will remove them from the database and they can view a student’s success rates. In the window which views the student’s success rates the admin can select to change that student’s password which will sent them to a new window to do so.

### Database Interactions

#### Logging in

When a user attempts to log in first their username is searched for in the database. If found the password they inputted will have the corresponding salt (stored in the database) appended to it. Then this string is hashed and compared to the hashed password in the database. If the password matches the User logs on. If either the password or username is incorrect a mini window pops up saying “Incorrect username or password”.

#### With admins

Admins will be able to change (per class) which algorithm solvers are available to the students. They will do this by checking a tick box. If checked the students in that class can use the solver. When the admin changes the checkbox state this Boolean value is then stored into the database. This means that user accounts can acknowledge which solvers are available to them.

Admins can also Create new classes and add new users to each class. When a new class is created a new class, entity is made in the class table of the database. This is the same for users but the users are also assigned the class they are in. Admins can also delete classes and Users. This will remove anything related to that User or class in the database.

#### With users

When users use the solver to answer a question the question they asked is stored into the database. The database holds the 10 most recently asked questions for each user. The purpose of this is to ensure users can’t attempt to answer a question in the “test your answers” section which they have had answered for them by the solver. The same thing will happen if a user answers a question correctly. If they answer a question incorrectly the question won’t be stored into the database as the user does not know the answer and can attempt the question again. This has been done to avoid students cheating to get a high success rate.

Success rates for each solver are also stored into the database. The user will not be able to see their success rates but the Admin will be able to view the success rates of each student in their classes. Every time a student attempts to answer a question their success rates in the database will be adjusted accordingly.

## Changes Made during production

Some tweaks had to be made to the SQL queries used in python. This was done to ensure the inputs made by users were acceptable to be inputted into the database; To ensure the Usernames/class names inputted were not empty strings and to ensure passwords were of adequate length (8 characters). To test the password length, I used a decorator (an in-built functionality in python) to test that the inputted password was of valid length before running the procedure that inputs it into the database.

On top of this a variety of try and except statements were used in the Solvers to ensure a valid question was asked by the user. A valid question is a question which will result in an answer for the problem. If no answer can be reached due to the inputs causing an error in the algorithm then it is displayed to the user that the problem they have inputted has no solution

In the design the functionality to add a new admin was left out of the GUI design. This has been corrected and is now a functional part of the Admin Screen.

In the design the functionality for and admin to change a users’ password was left out of the GUI design. This has now been added to the Success Rate Viewer screen for each user.

The linear programming solver tool was missing an important feature which was to identify whether the problem being asked needed to be minimised of maximised. A drop-down menu has now been including to allow the user to identify which the program needs to do.

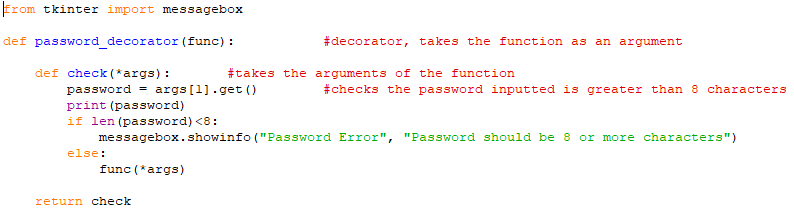
A new window has been included for when the program is first run. This allows the Admin that first uses the program to create their account.

# Technical Solution

|  |  |
| --- | --- |
| 1 | Password Decorator |
| 2 | Password Hasher |
| 3 | Question String |
| 4 | Login Menu |
| 5 | Admin Page |
| 6 | Main Menu |
| 7 | Matrix |
| 8 | Graph |
| 9 | Minimum Spanning Tree (Solver) |
| 10 | Minimum Spanning Tree (GUI) |
| 11 | Critical Path Analysis (Solver) |
| 12 | Critical Path Analysis (GUI) |
| 13 | Route Inspection (Solver) |
| 14 | Route Inspection (GUI) |
| 15 | Linear Programming (Solver) |
| 16 | Linear Programming (GUI) |
| 17 | Travelling Salesman Problem (Solver) |
| 18 | Travelling Salesman Problem (GUI) |

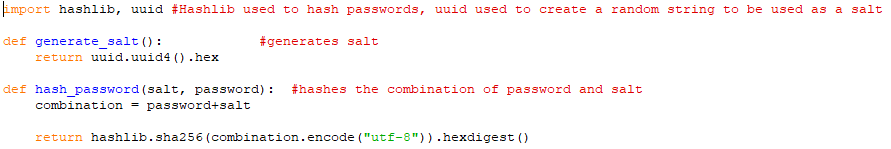
1.

Password decorator (password\_decorator.py) is a file used to test whether an inputted password meets the criteria needed for a password to be deemed secure enough (in this case longer than 8 characters)



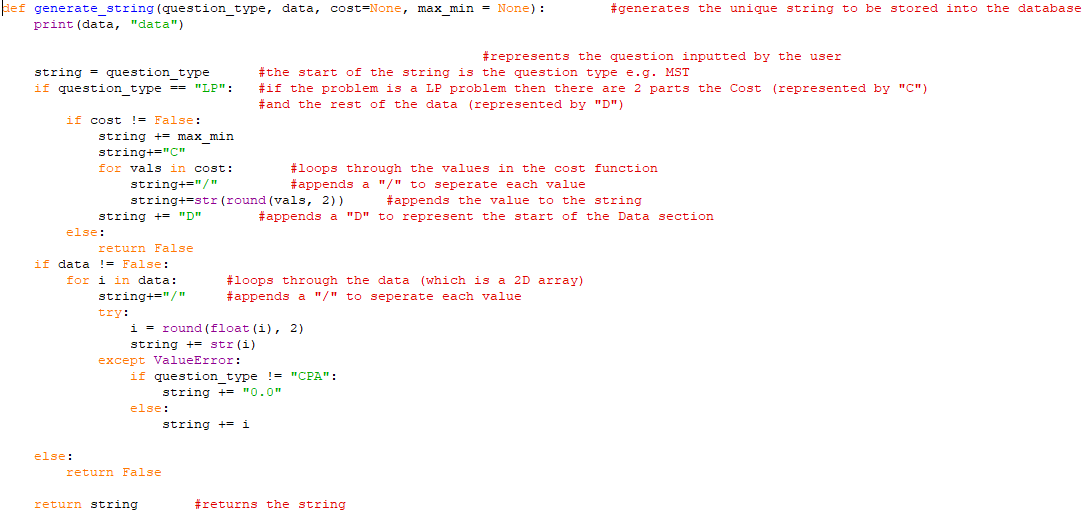
2.

Password hasher file (password\_hasher.py) is used to generate a salt for the password and also hash the password so it is not stored as plaintext into the database



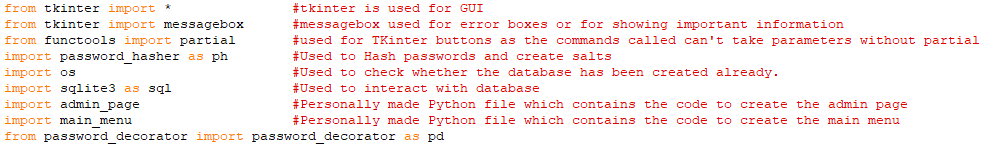
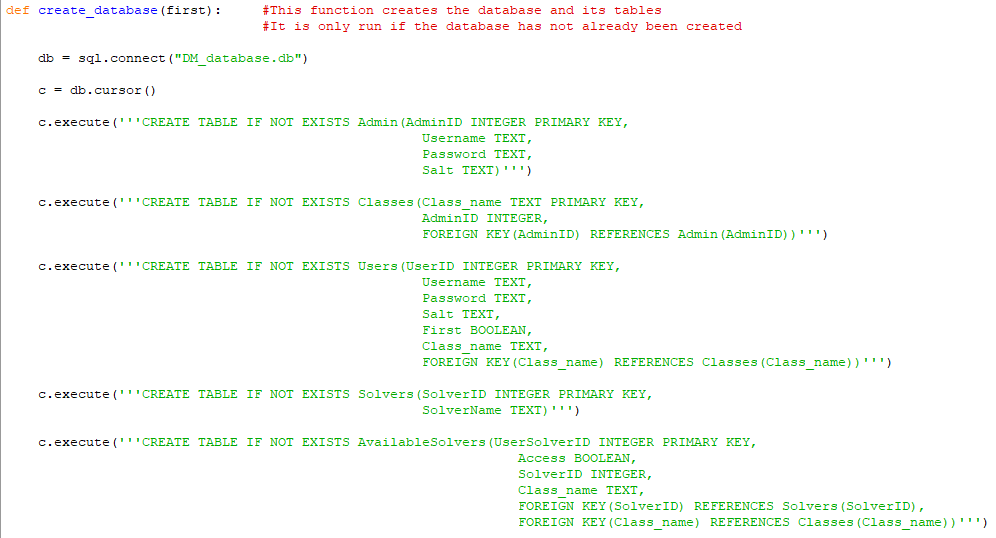
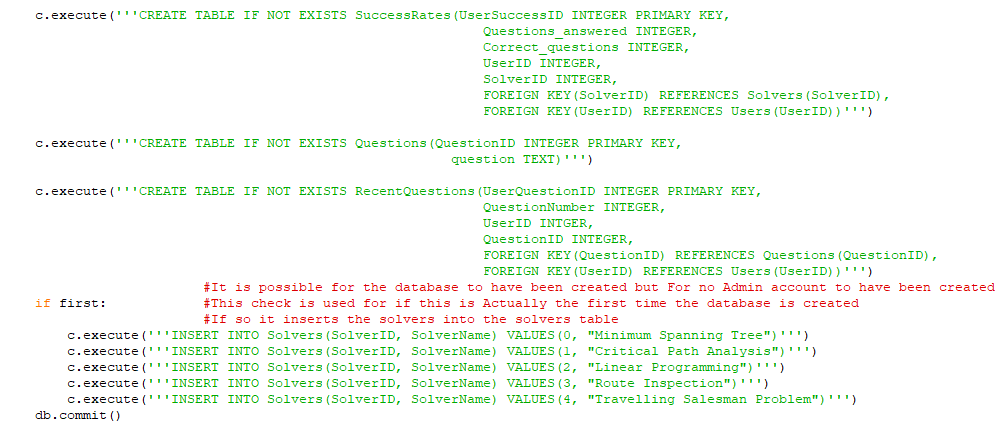
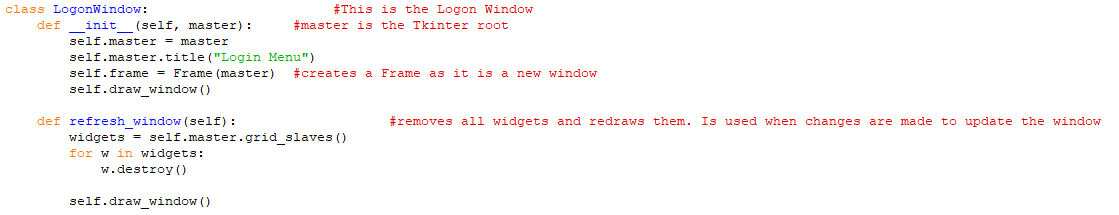
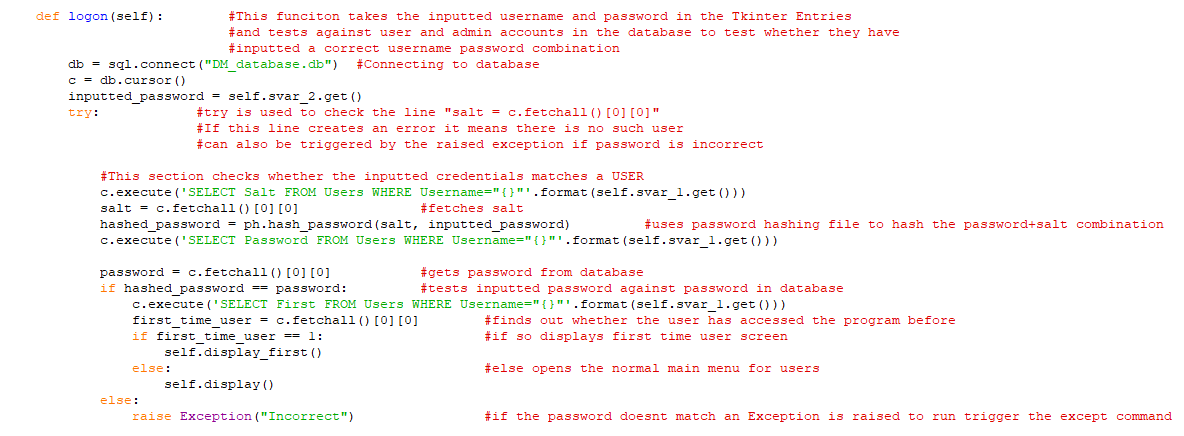
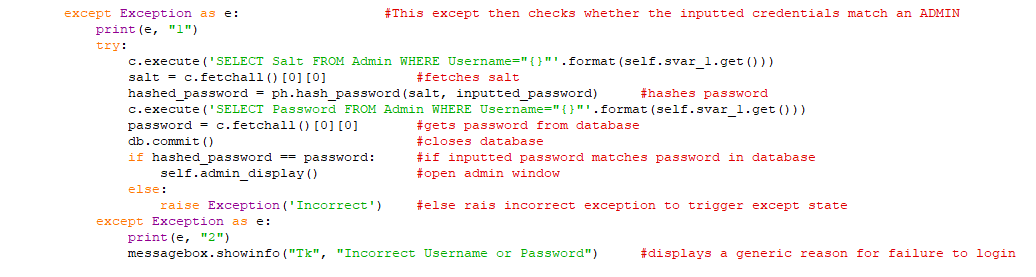
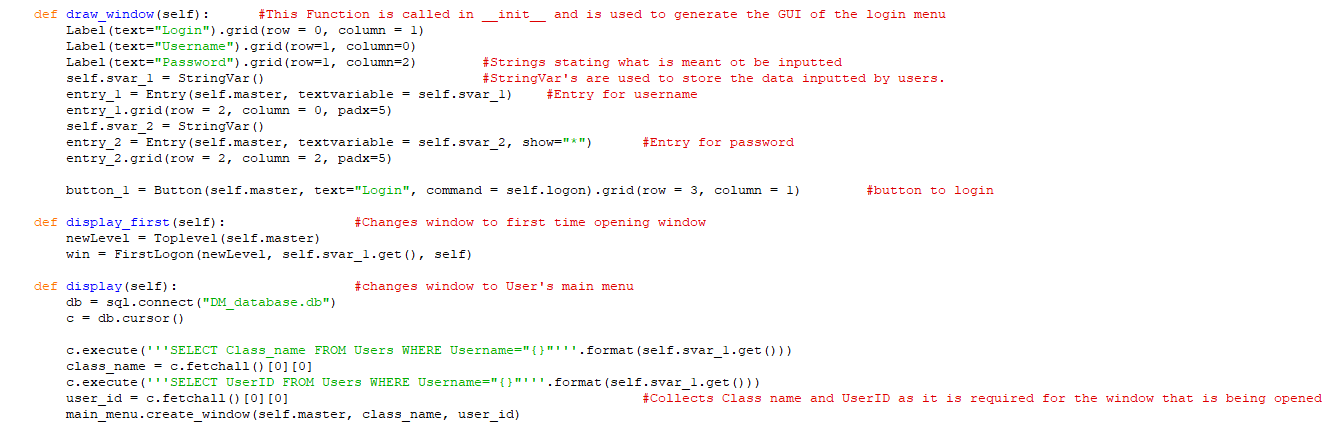
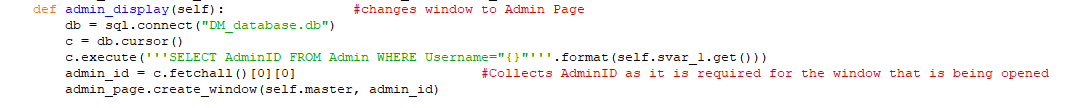
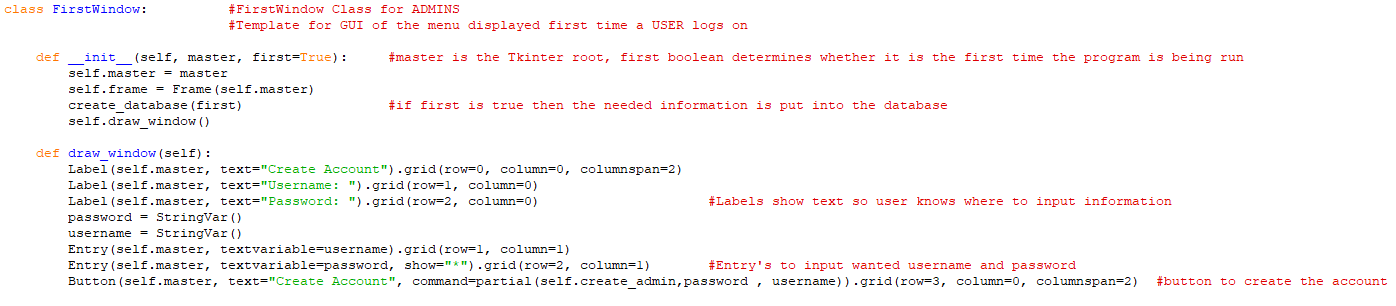
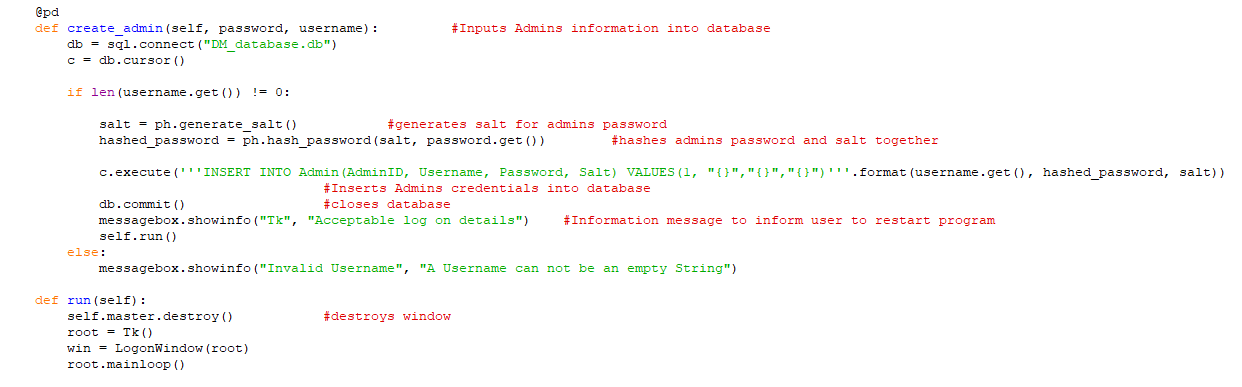
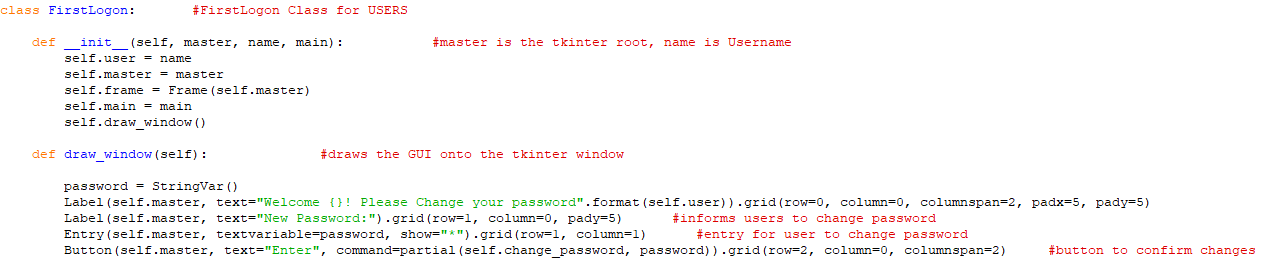
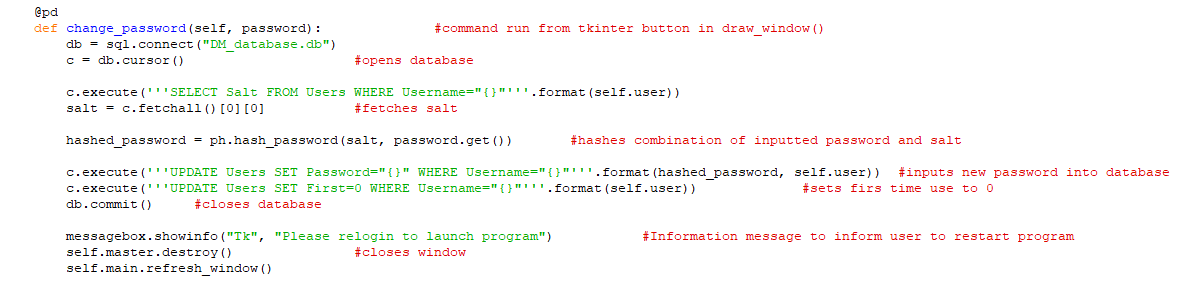
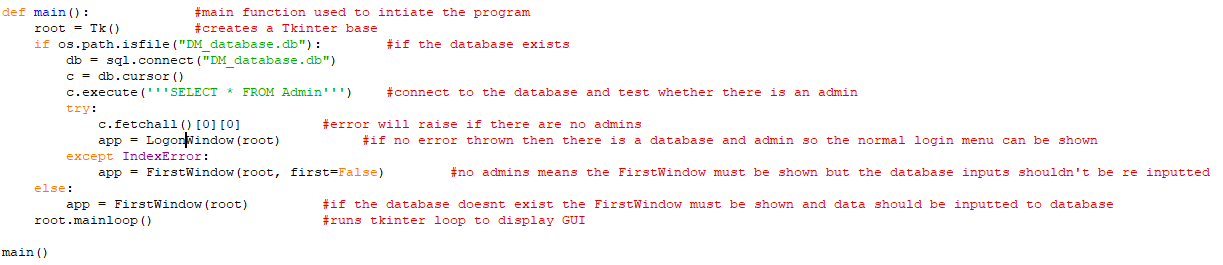
3.

Question string file (question\_string.py) is a file used to generate a unique string to store into the database representing the question that has been asked.



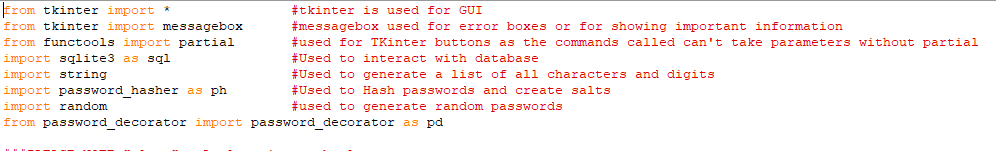
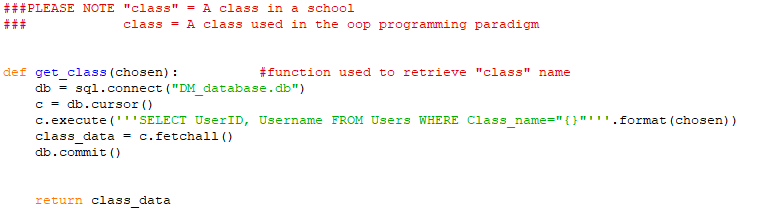
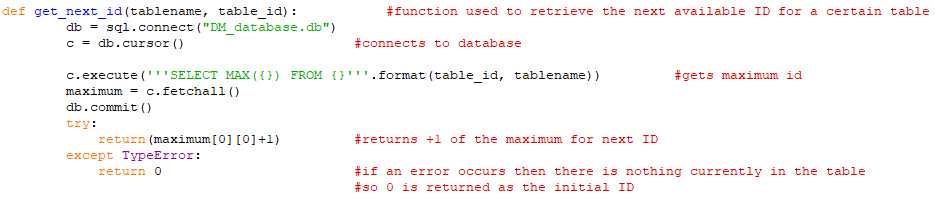
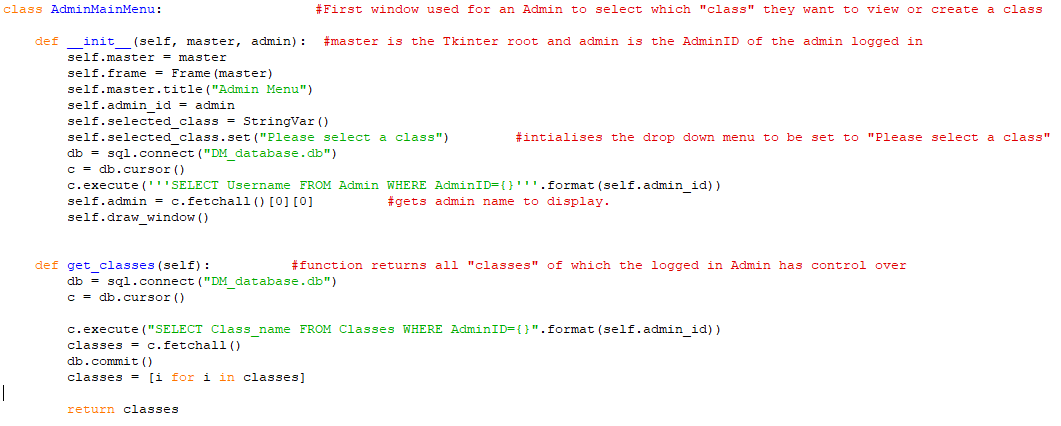
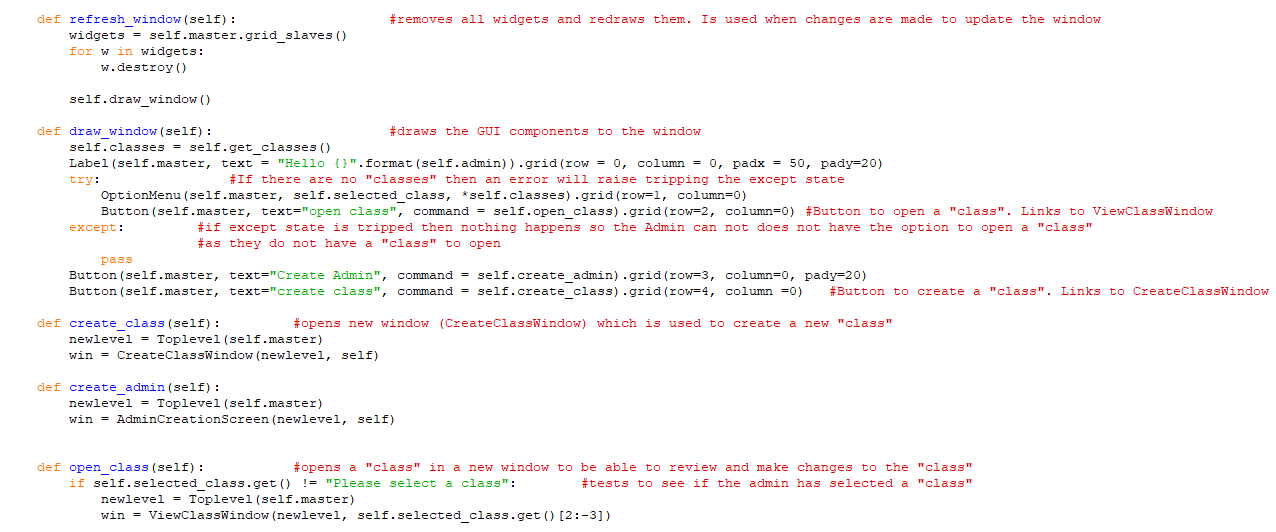
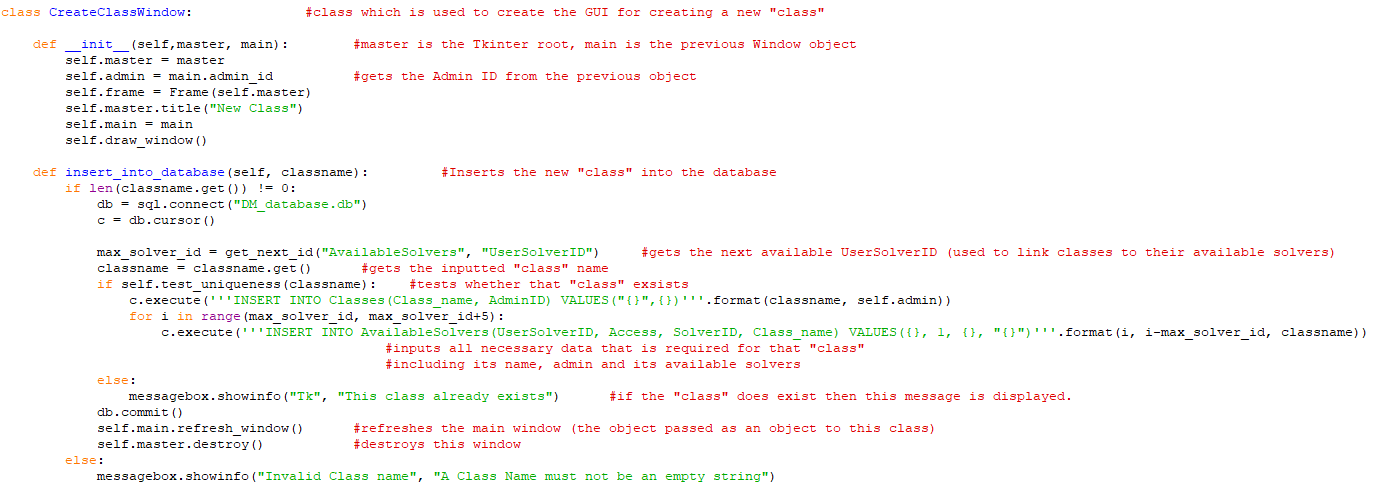
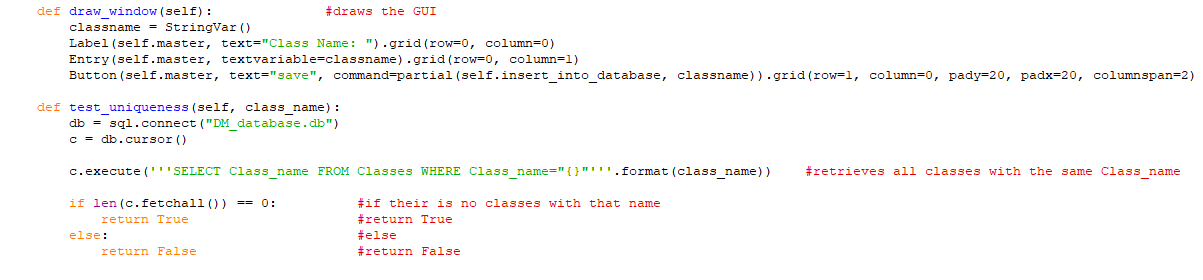
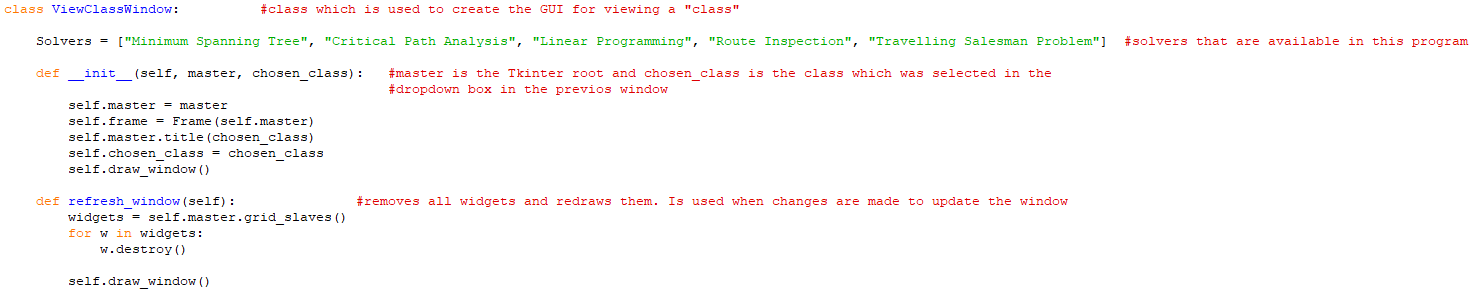
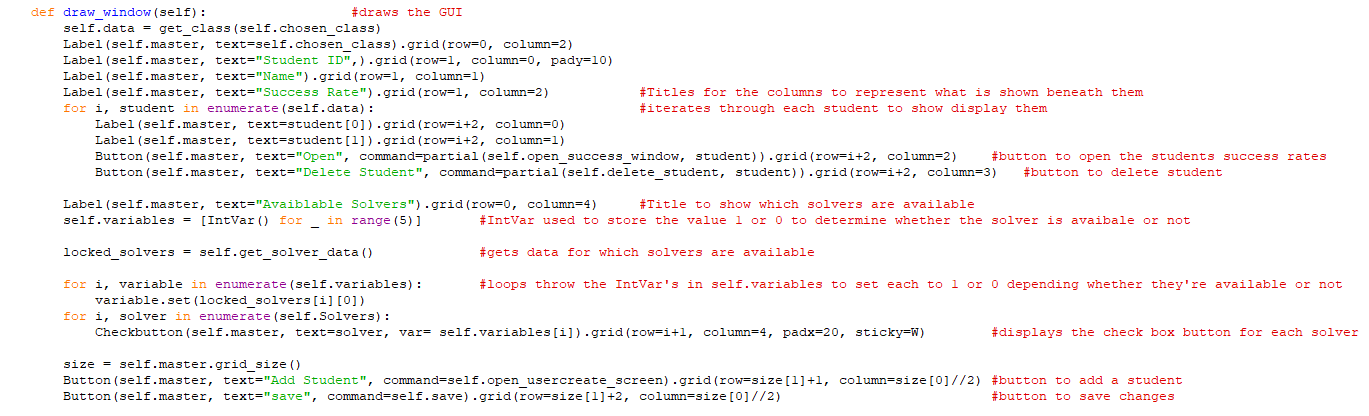
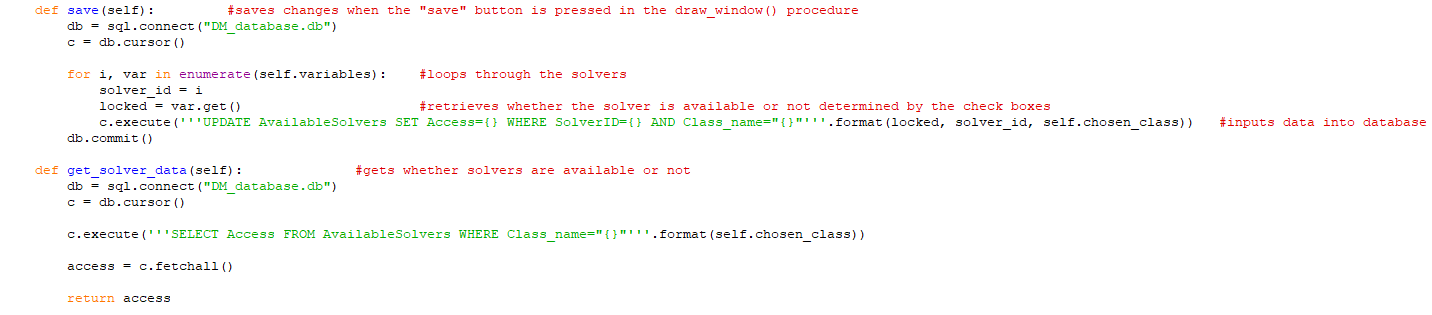
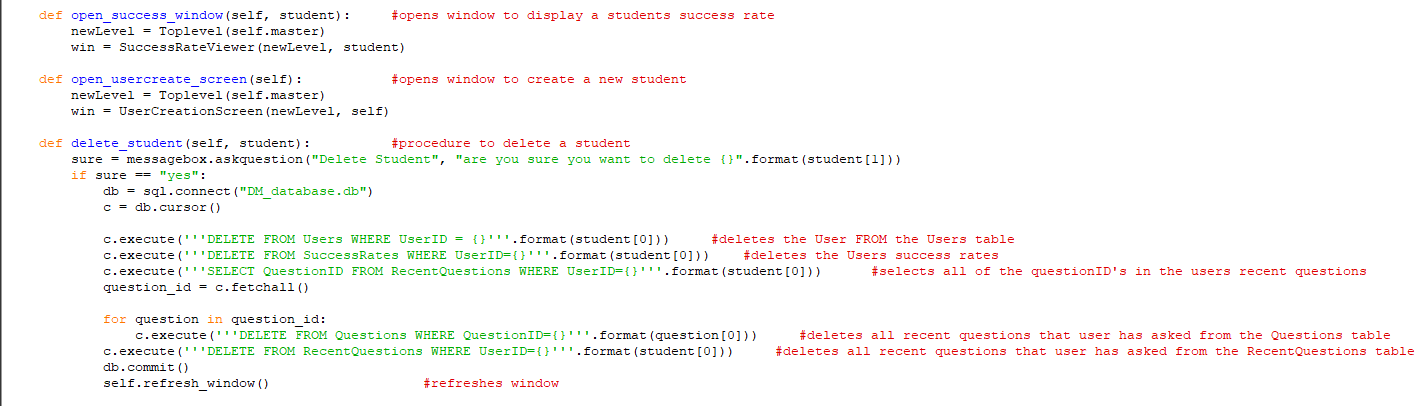
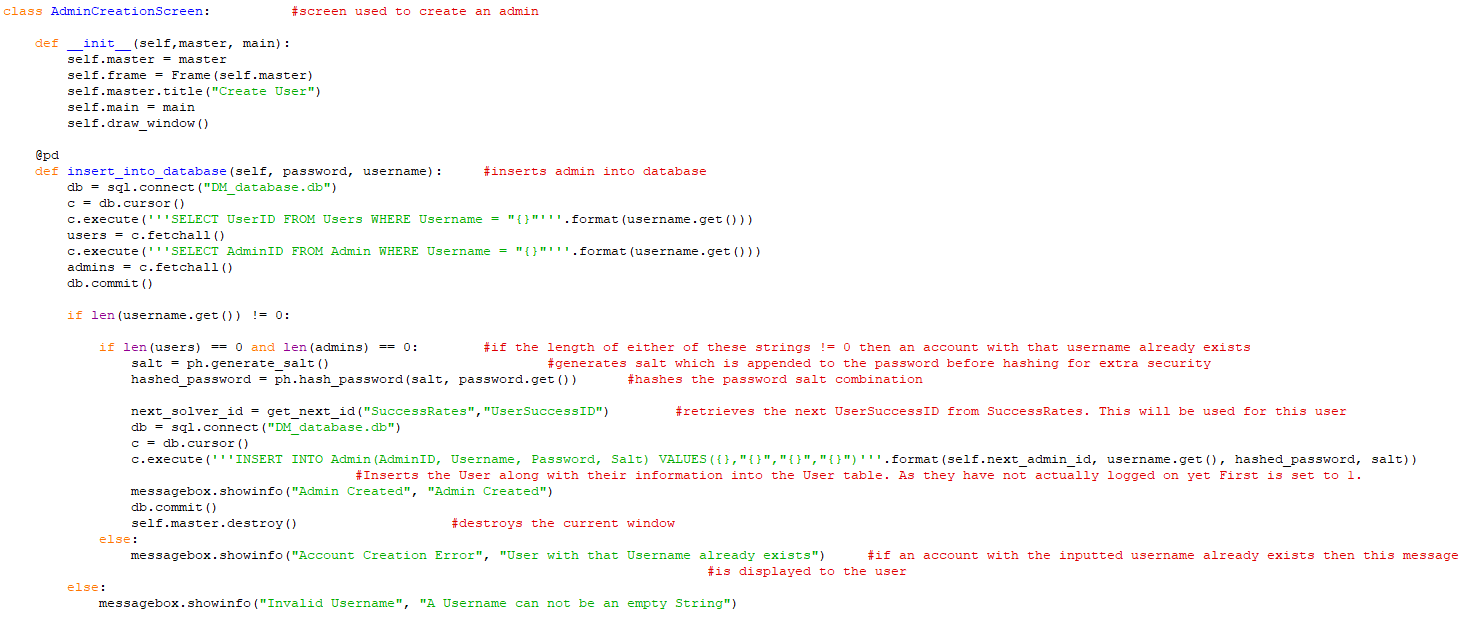
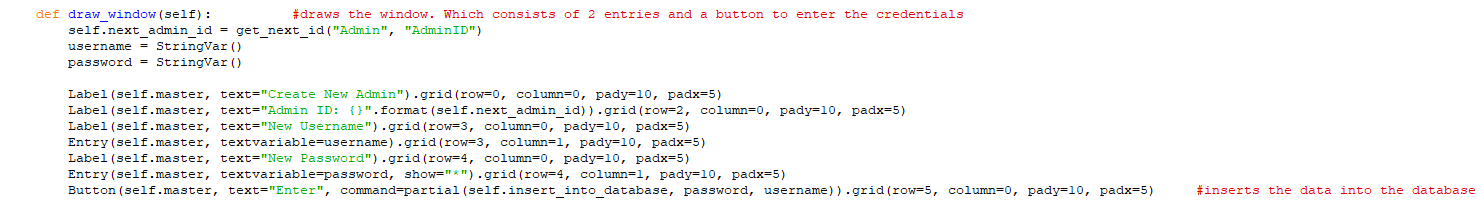
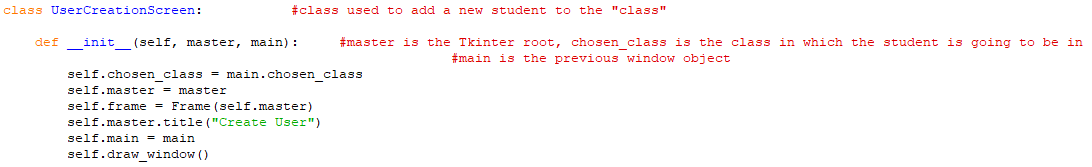
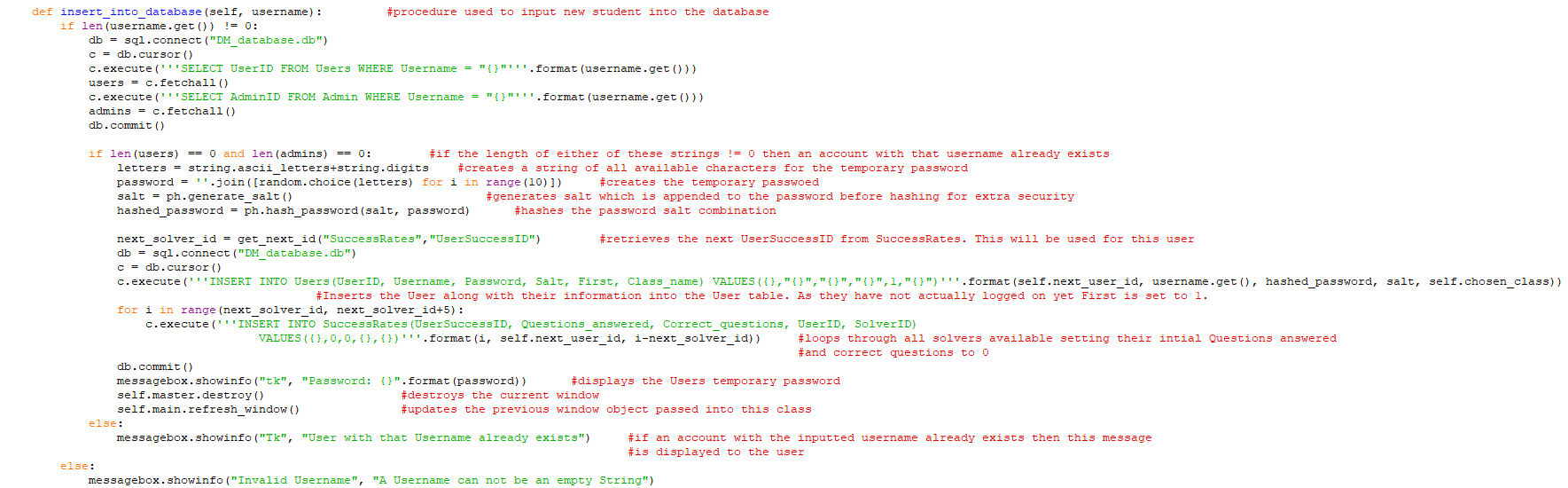
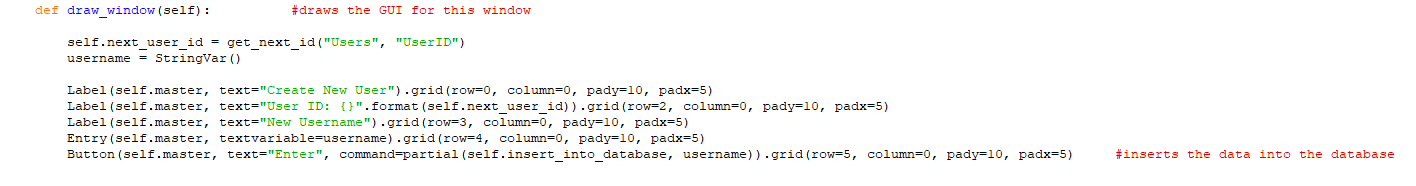
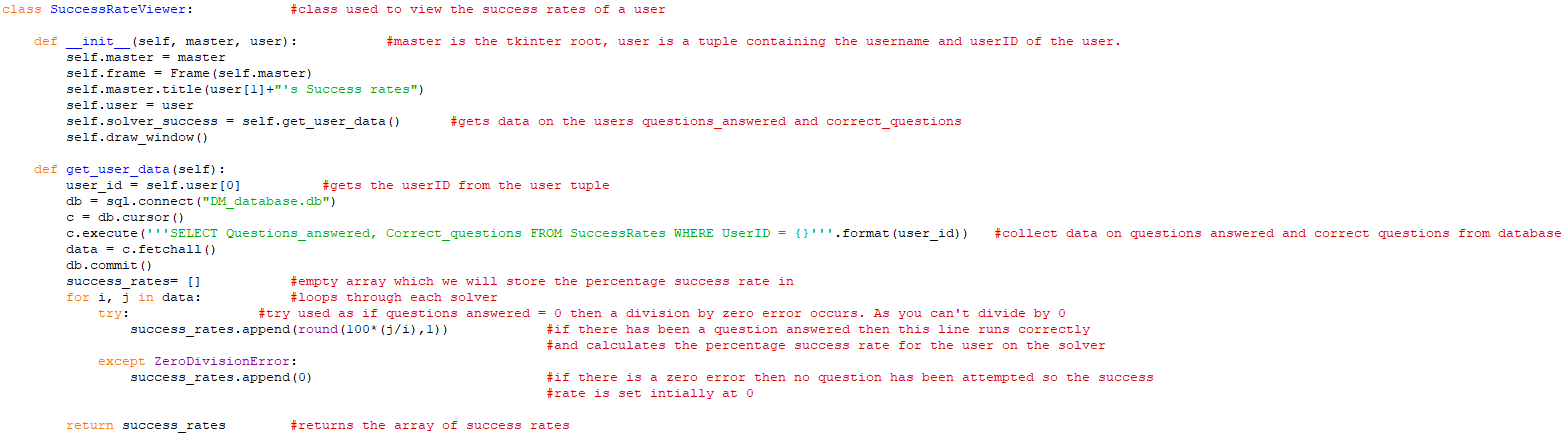
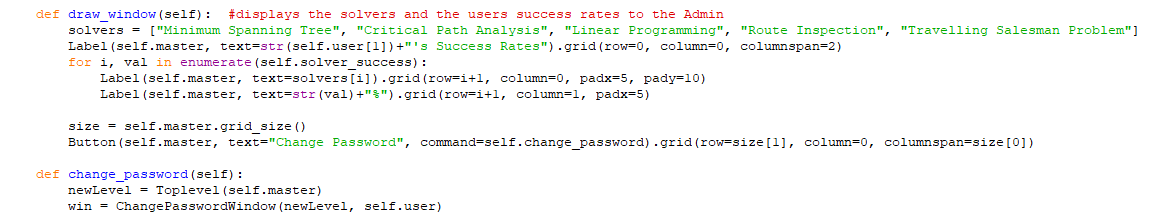
4.

Login menu file (login\_menu.py) comprises of all screens which could be shown when attempting to login to the program. This includes the create account screen for when the program is first run, the login screen for when logging into the program, and for user’s the first login screen which allows them to change their password. This file also contains the procedure used to create the database.

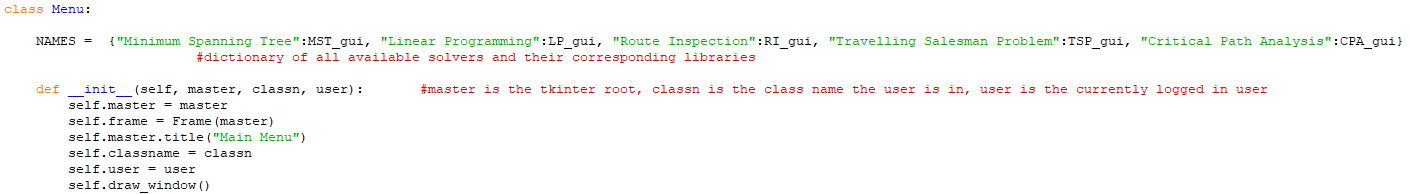
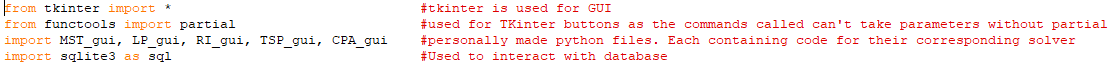
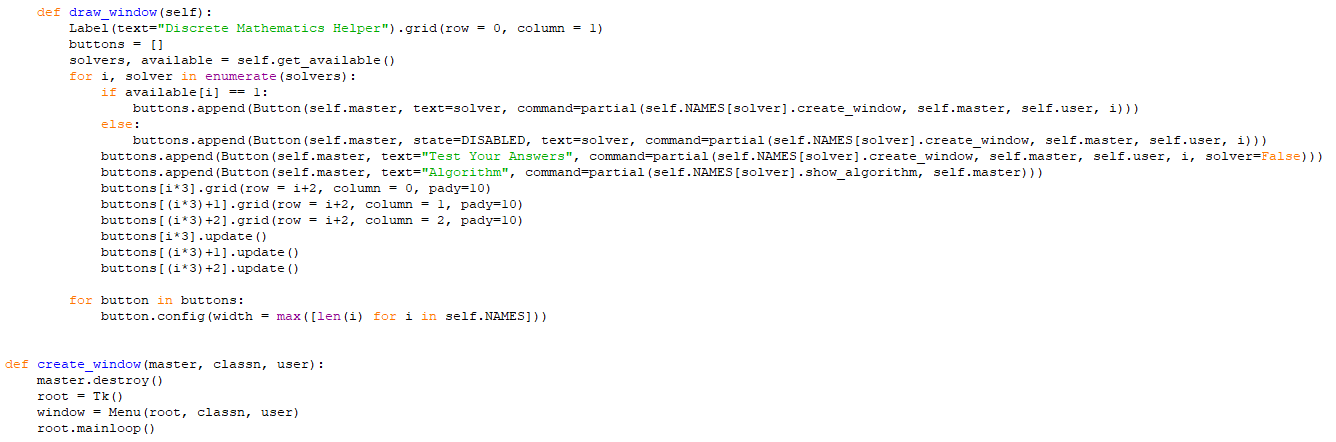
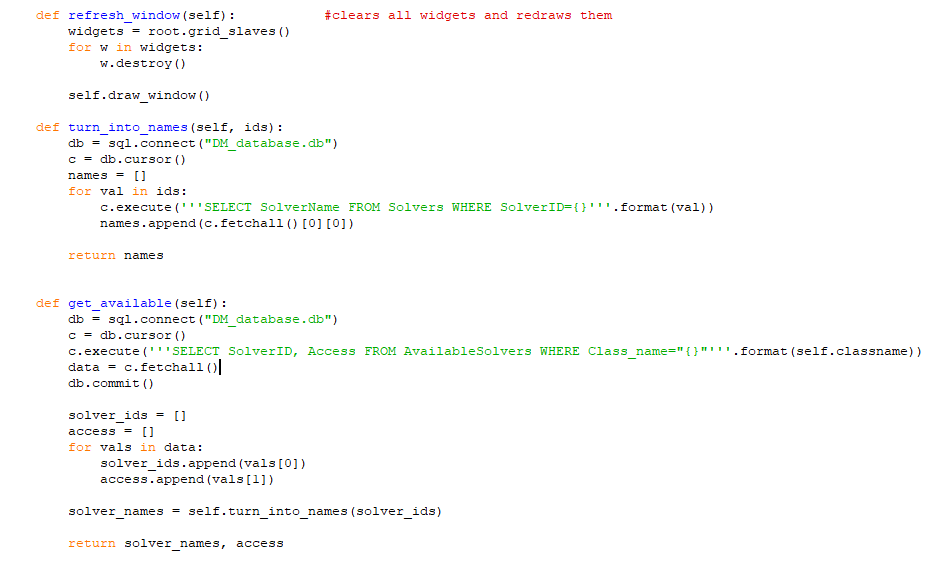
5.

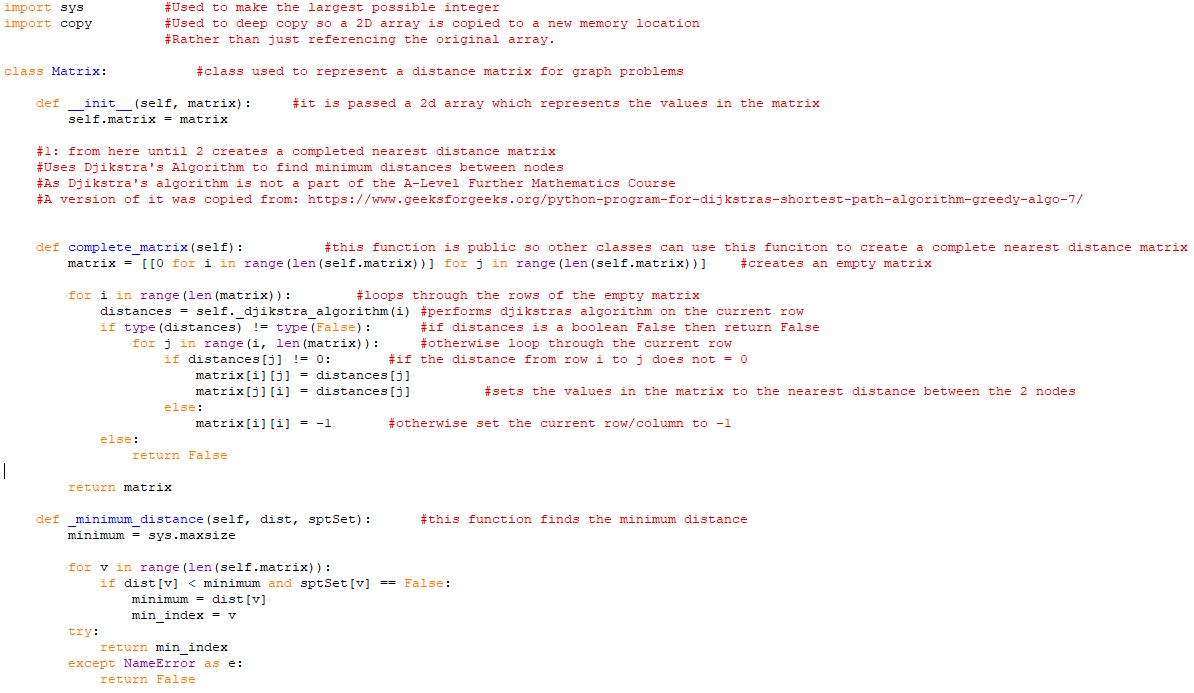
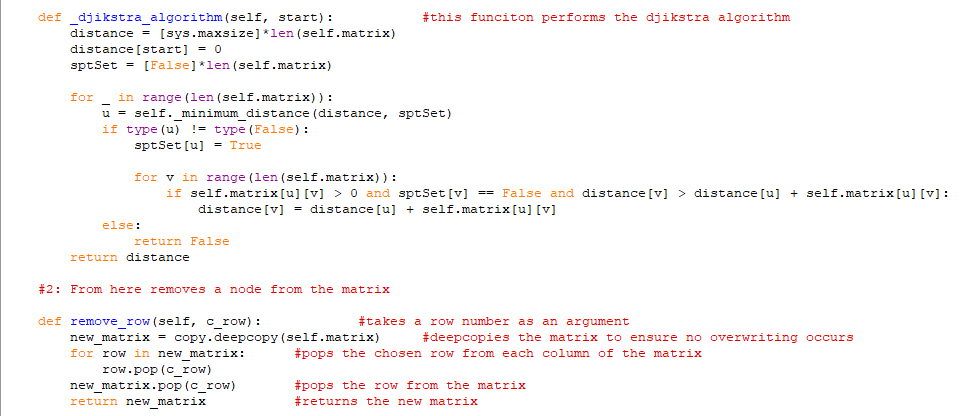
Admin page file (admin\_page.py) consists of all windows that an Admin will have access to when they log in. this includes: Admin main menu used to create admins, create classes and view classes; Admin creation screen which is used to create a new admin; Class creation screen which is used to create a new class; Class viewer screen which is used to view students in a class and change which solvers are available; user creation screen which is used to create a new user; Success rate viewer screen which allows admins to view success rates of individual user’s; User password change screen which is for when a user forgets their password then their admin can reset it for them.

6.

Main menu (main\_menu.py) is the screen in which the user’s will be able to access all of the solvers and other windows available to them.

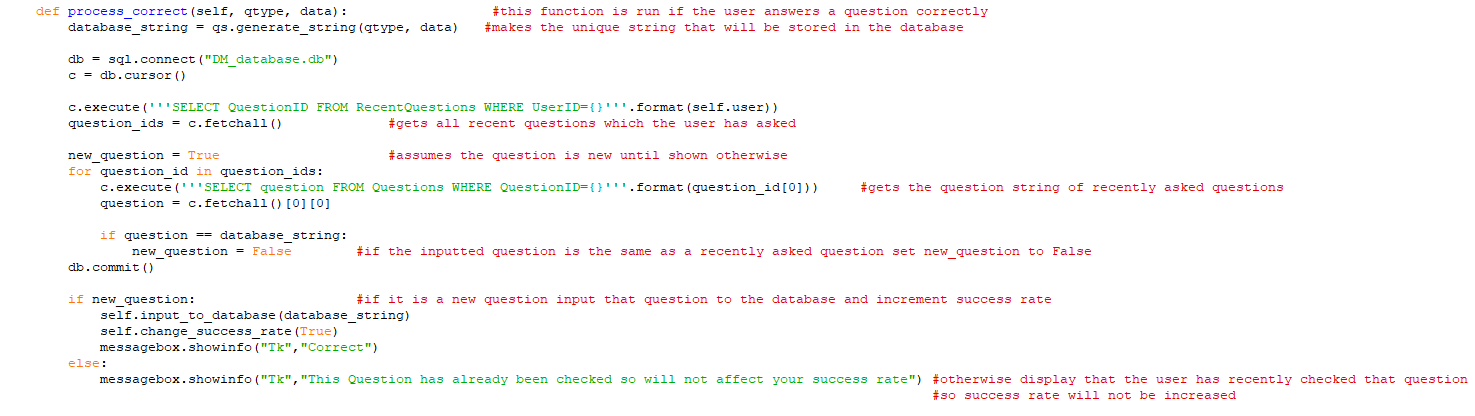
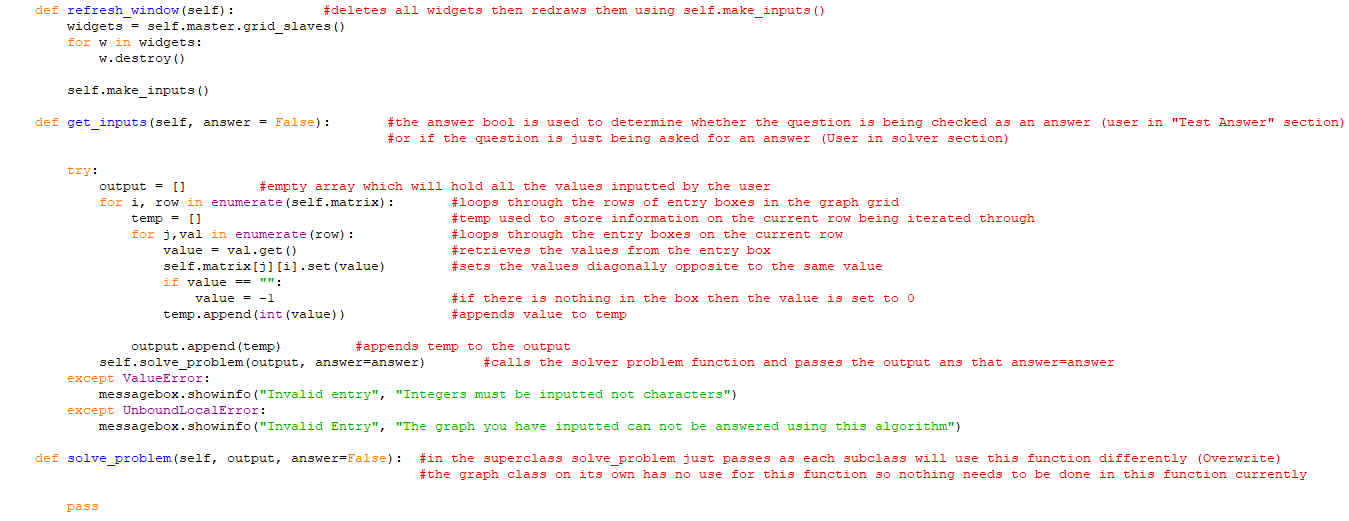
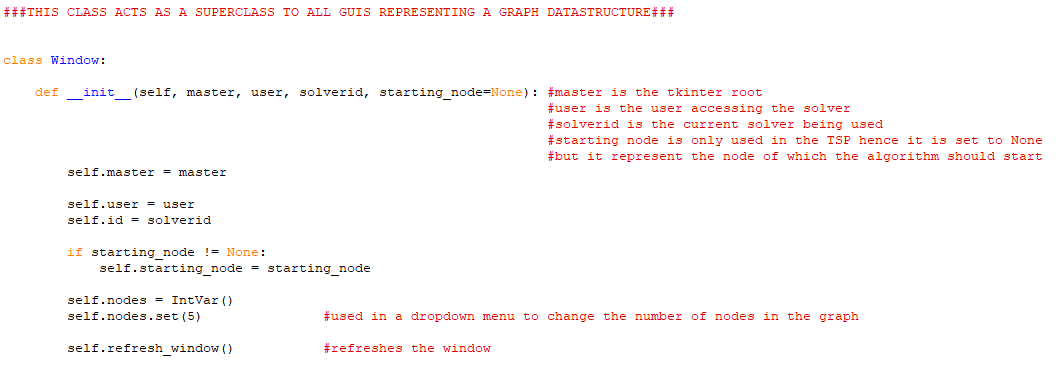
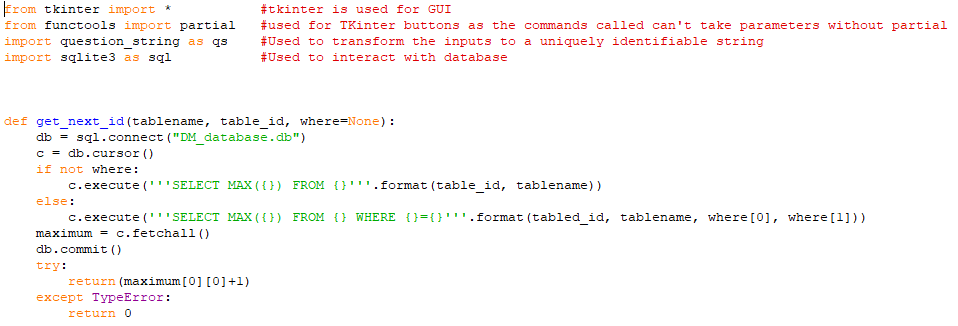
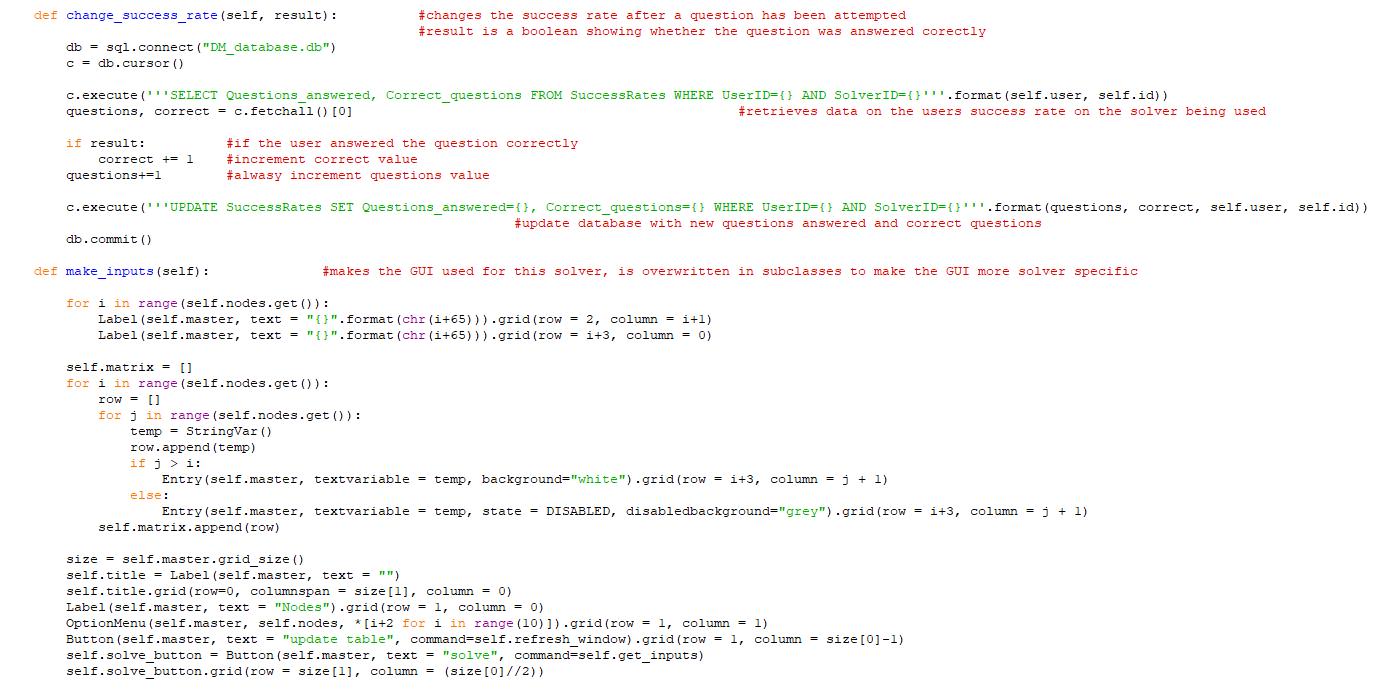
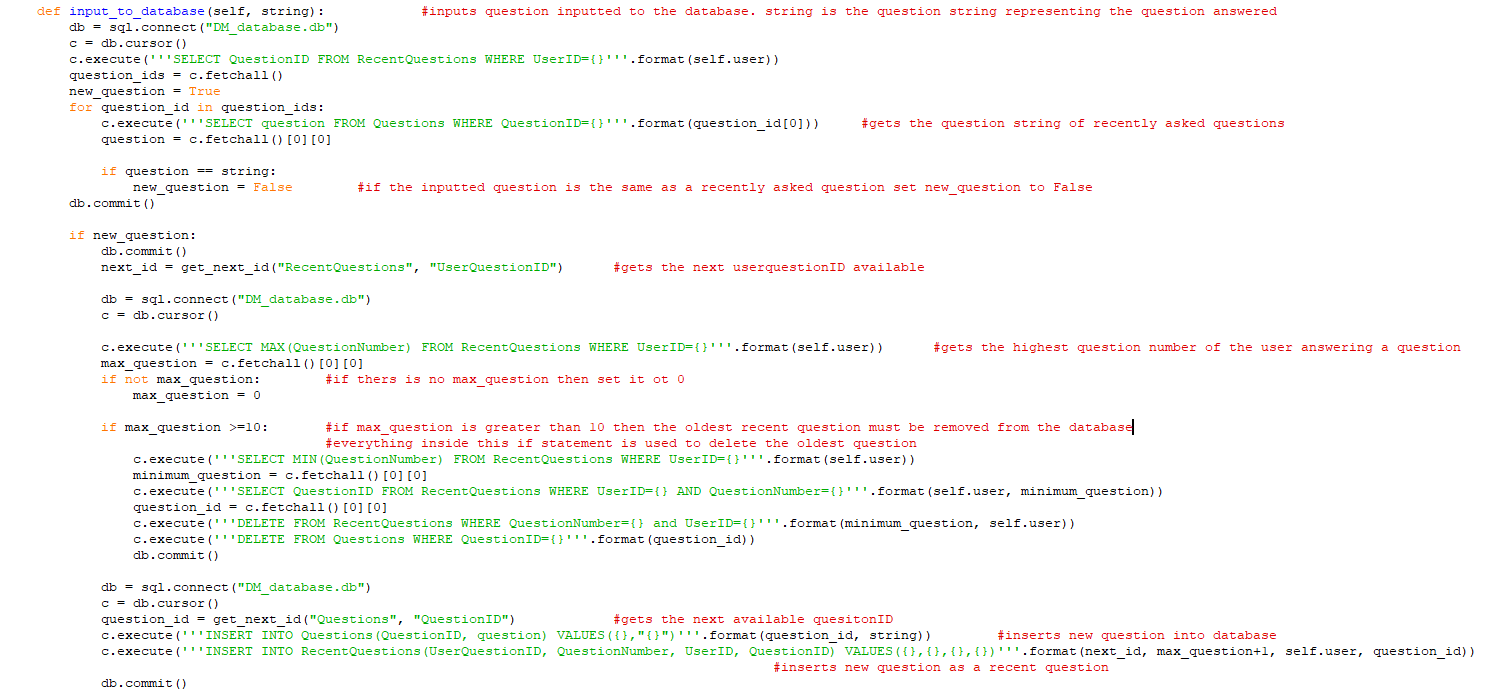
 

7.  
The Matrix file (Matrix.py) is used as a super-class for all graph problems as I use a distance matrix to represent the graphs.   


(Djikstras shortest path algorithm - greedy Algo-7, 2020)

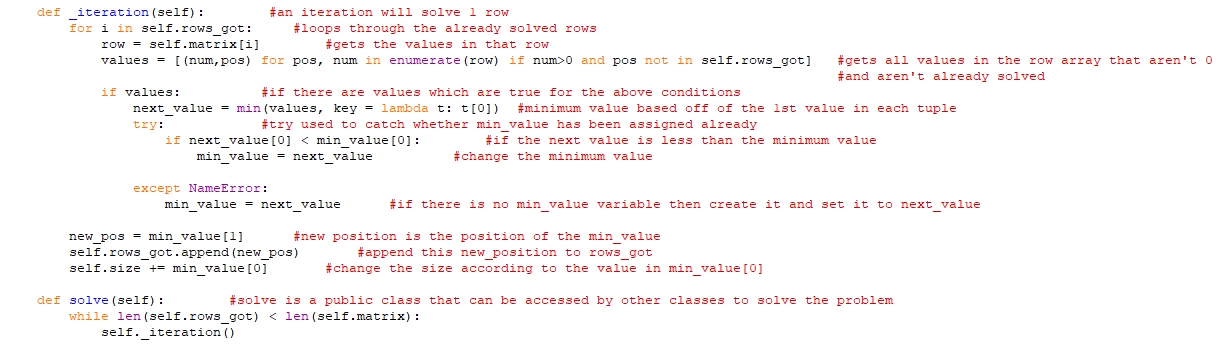
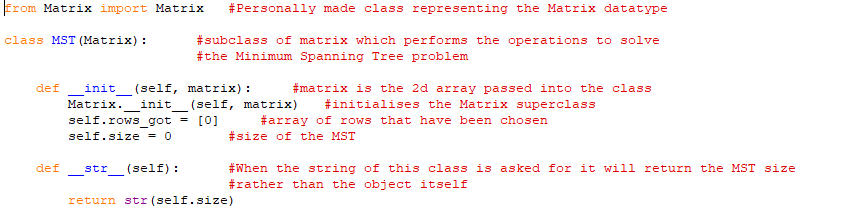
8.

The graph file (graph.py) is used as a superclass to show the GUI for any graph problem solvers. This is because all the graph solvers use a very similar GUI design, so I have created a generic design and have made small adjustments to the individual solvers where necessary

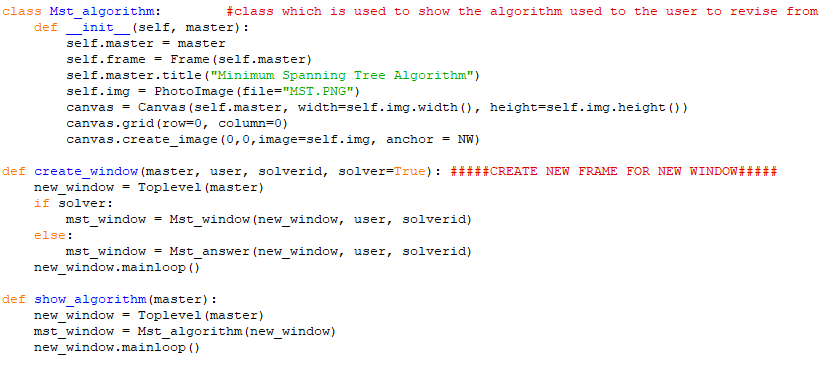
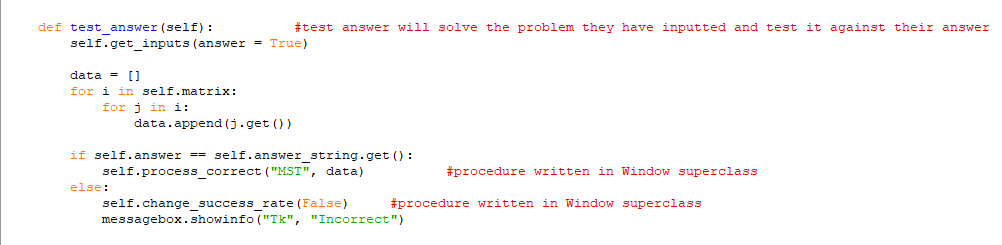
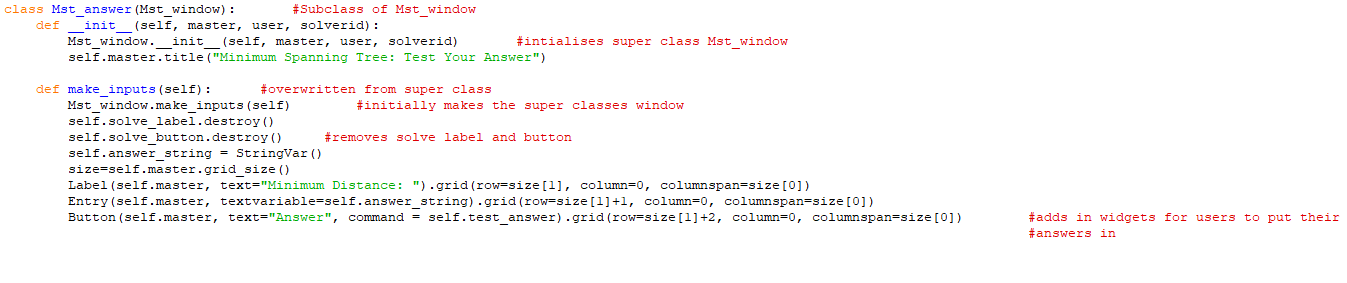
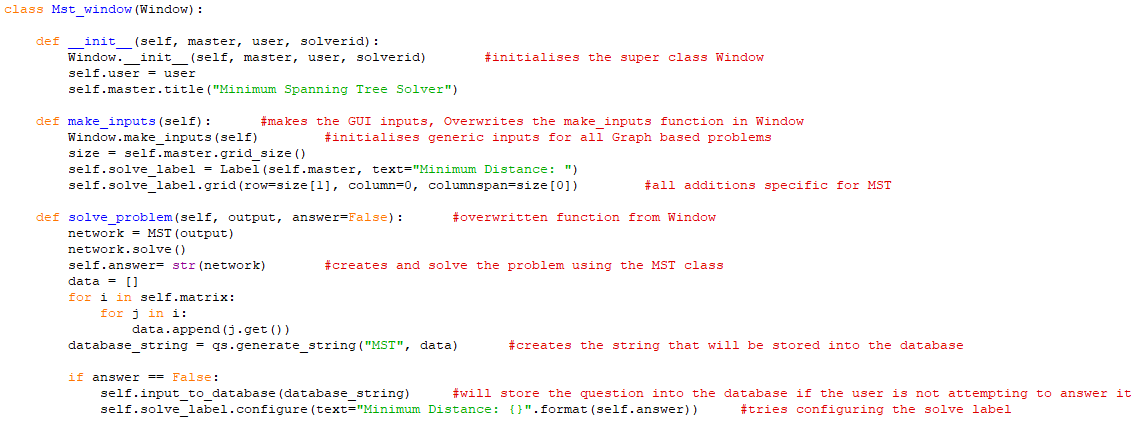
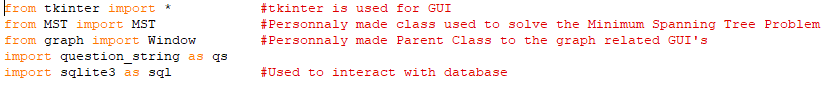
9.

The Minimum Spanning Tree solver (MST.py)



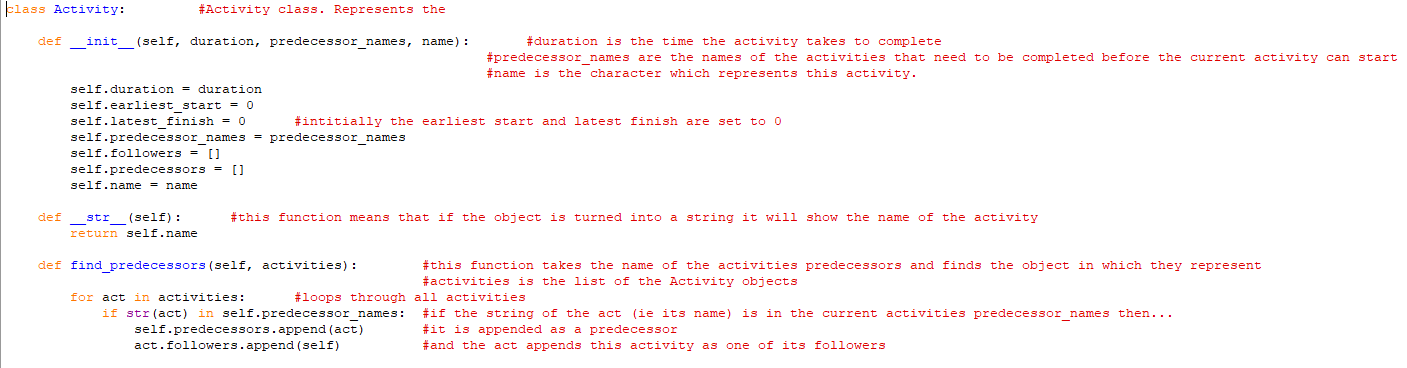
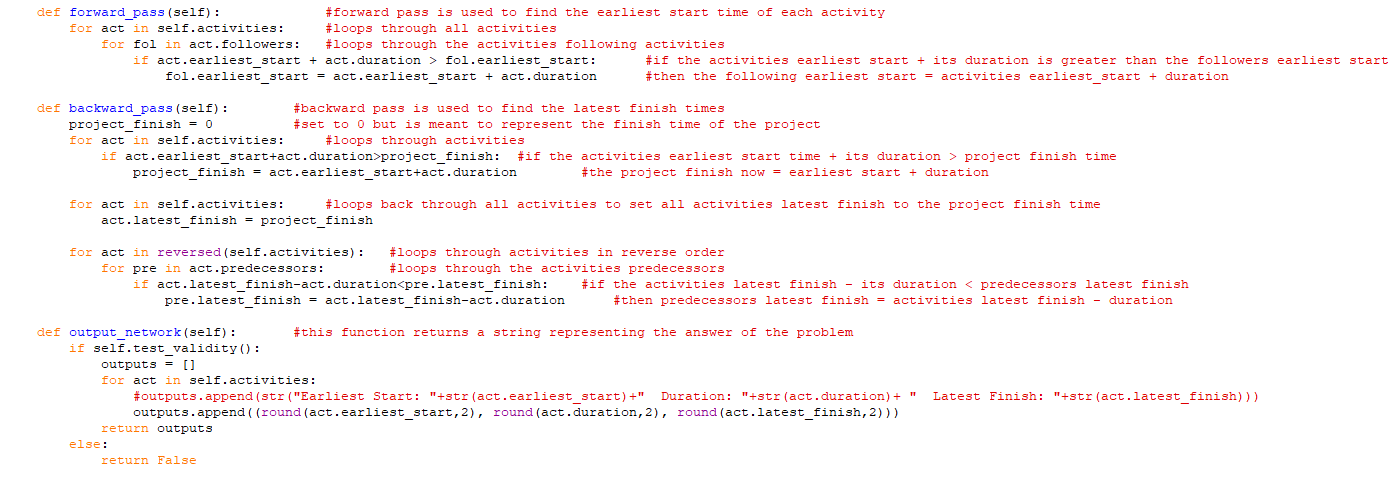
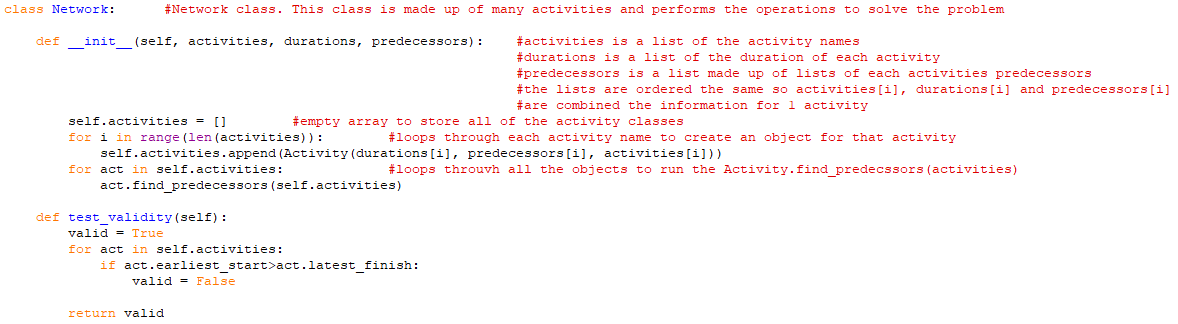
10.

Minimum Spanning Tree (GUI) file (MST\_gui.py) is a subclass of graph which makes small adjustments to the graph GUI design to specify it to solve the Minimum Spanning Tree problem.



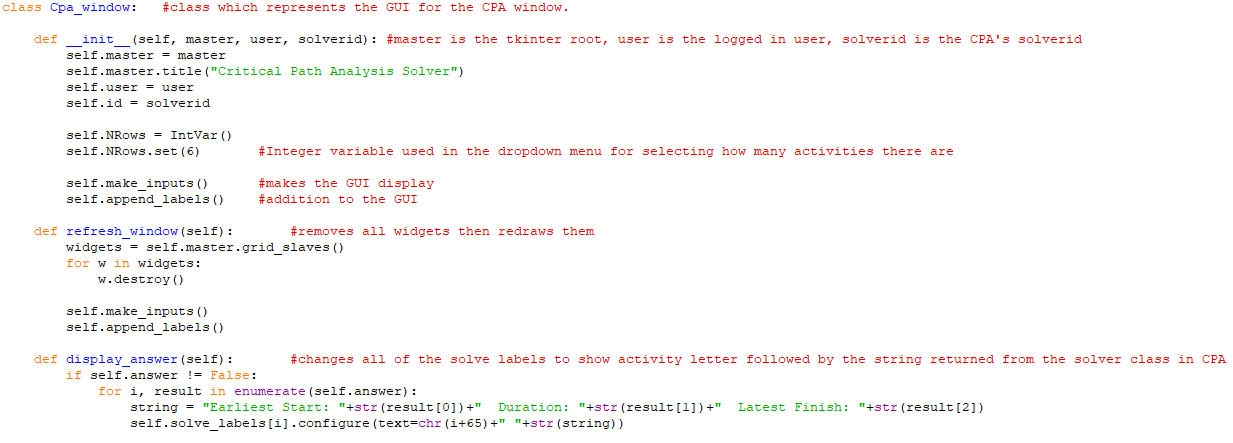
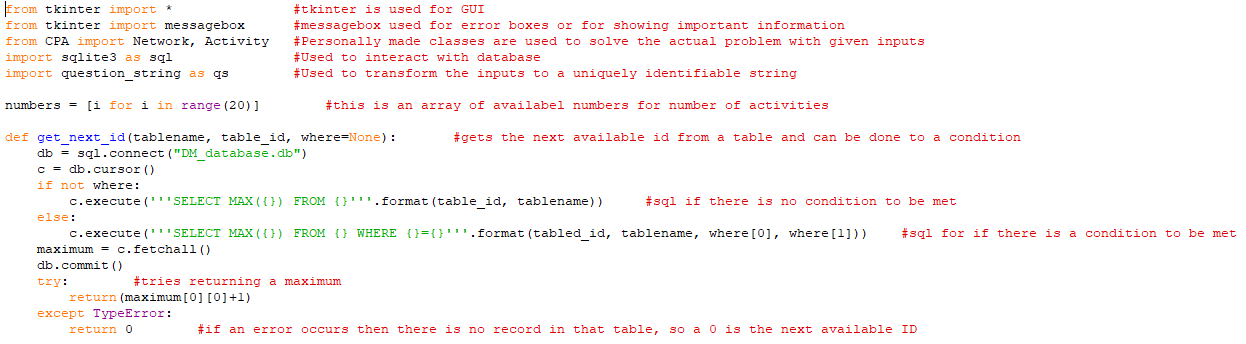
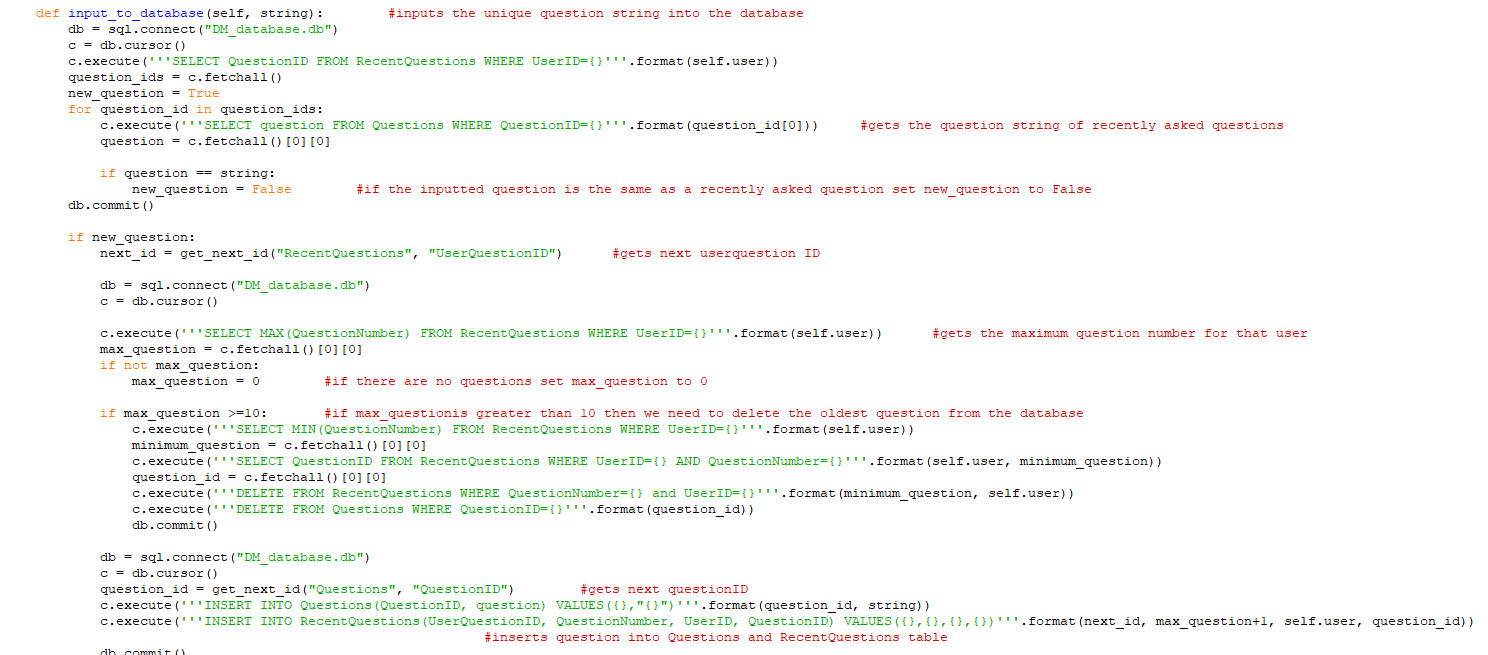
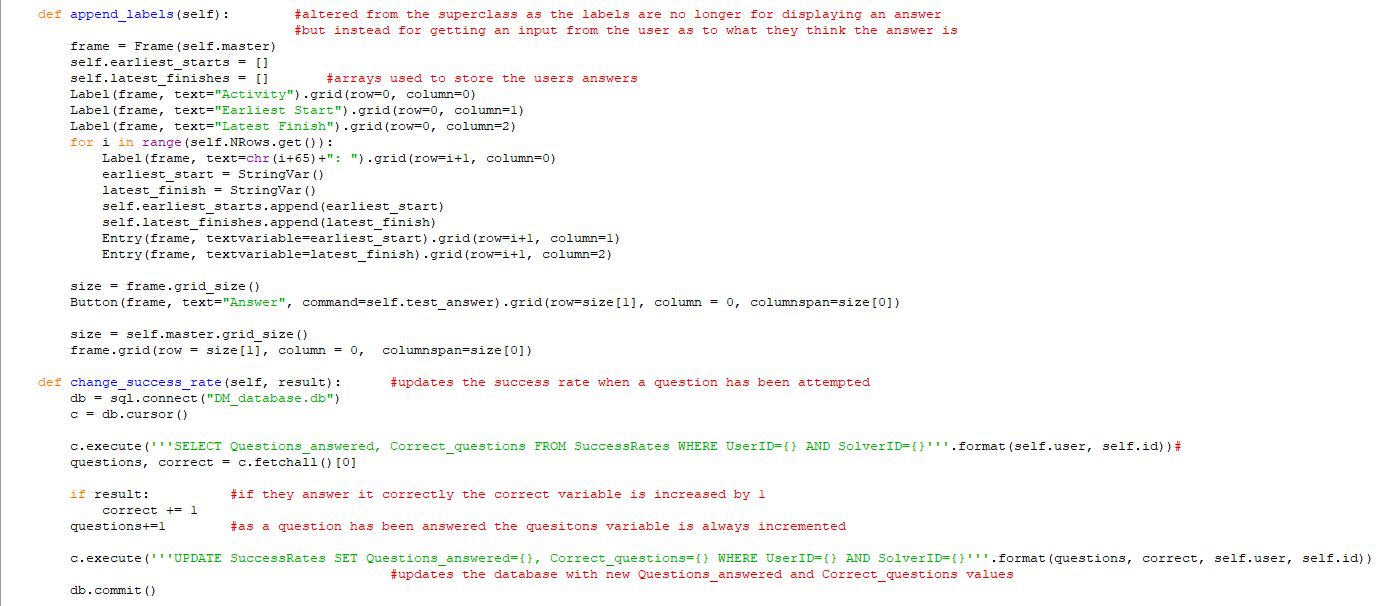
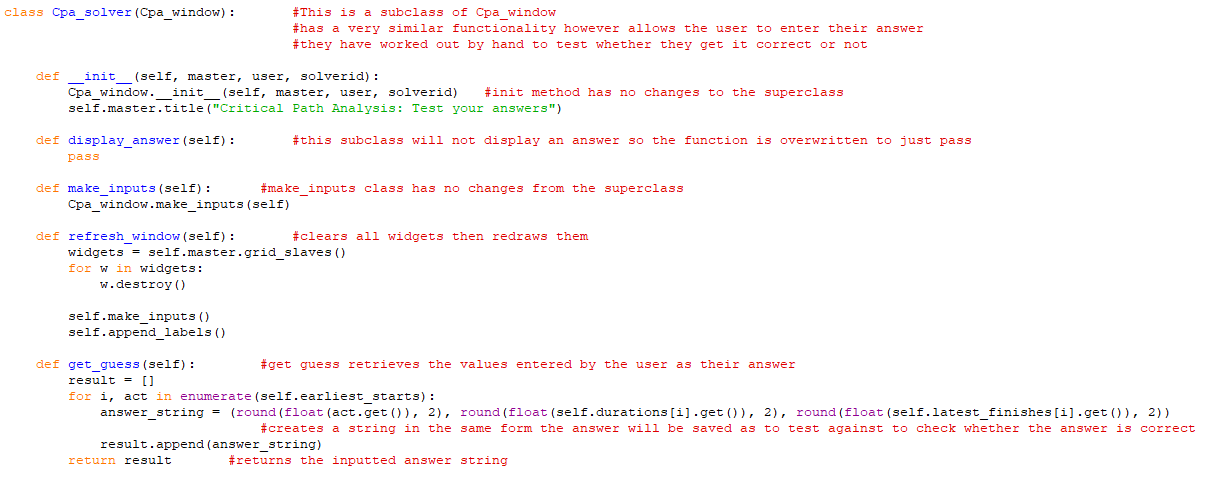
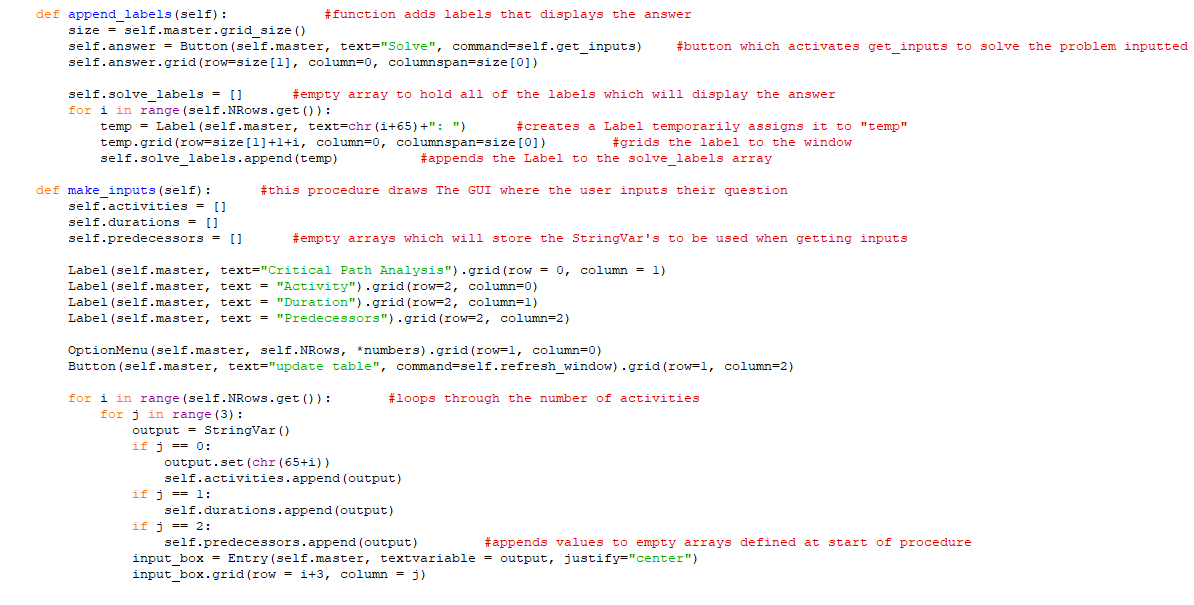
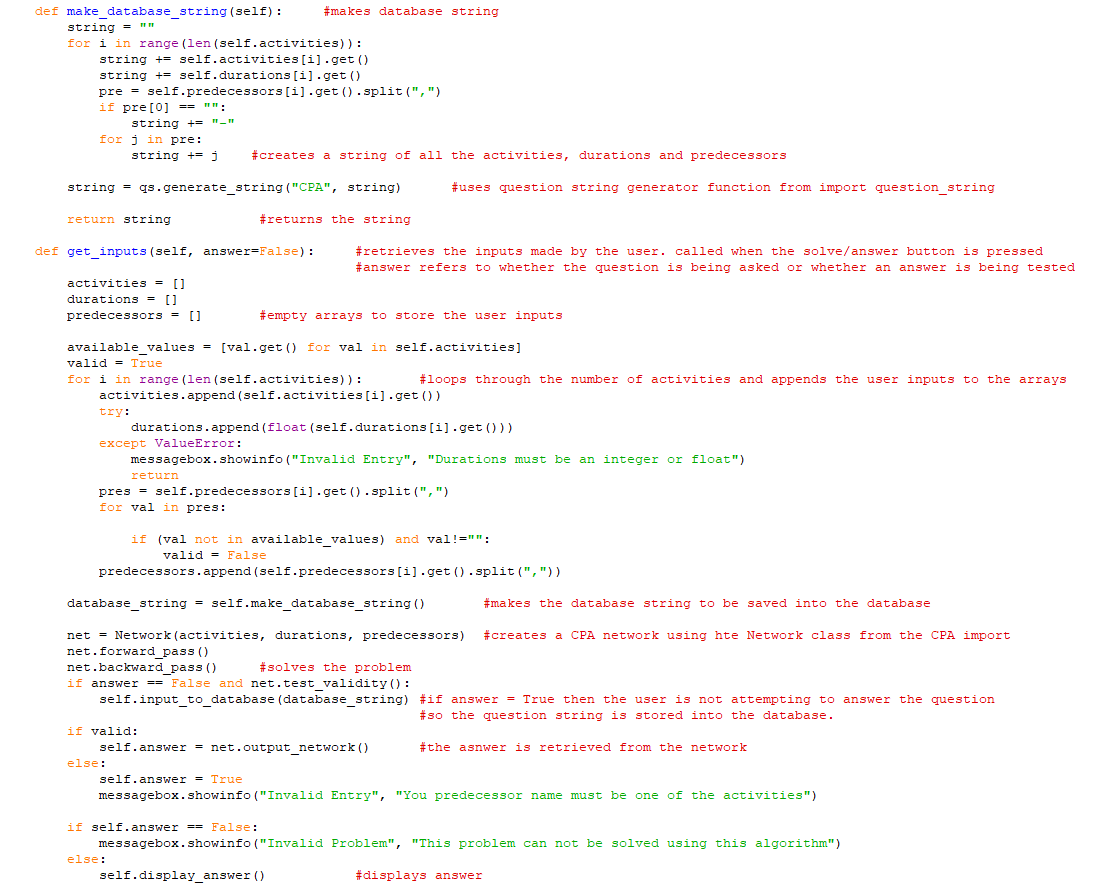
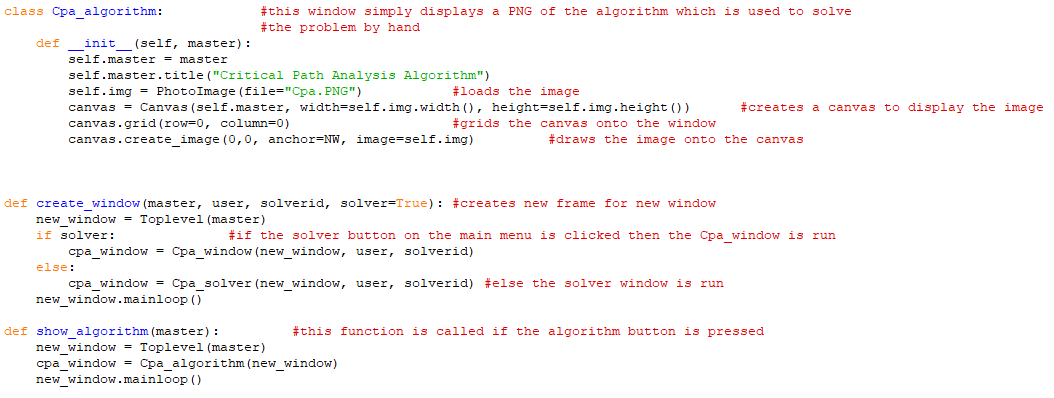
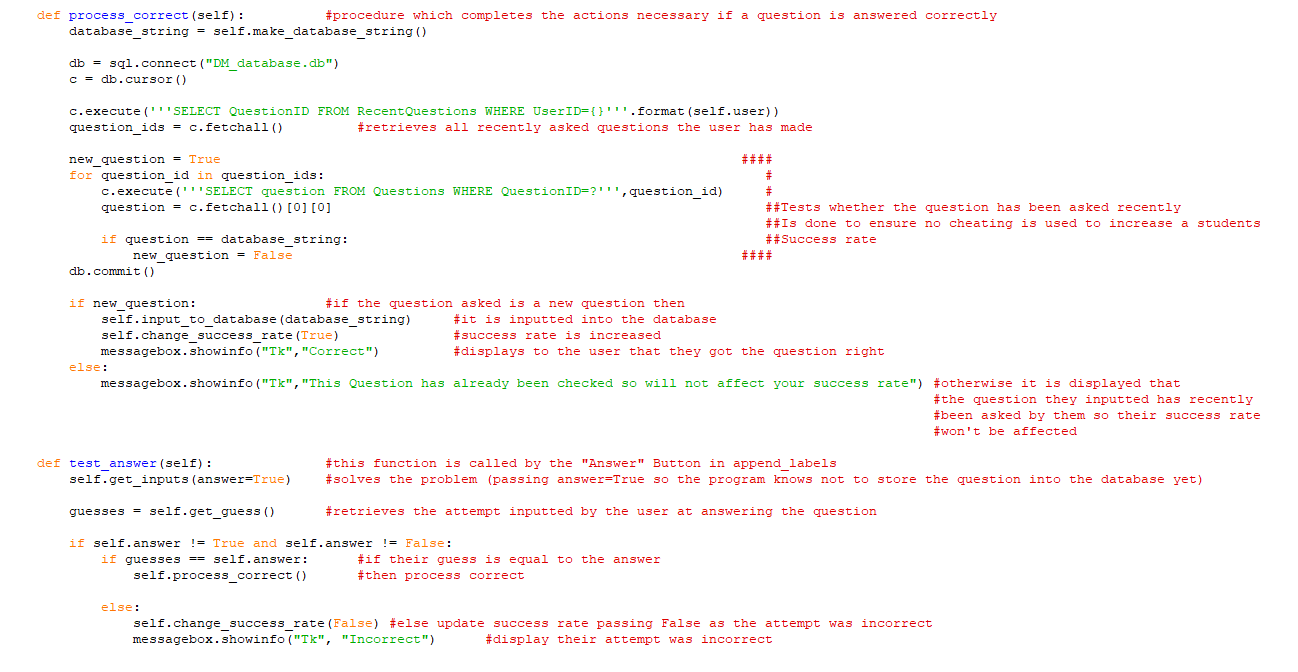
11.

Critical Path Analysis solver file (CPA.py) is used to solve any path analysis question inputted to it.

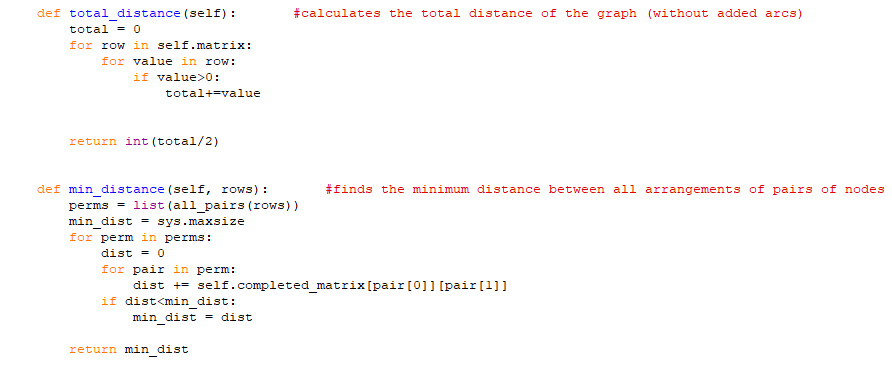
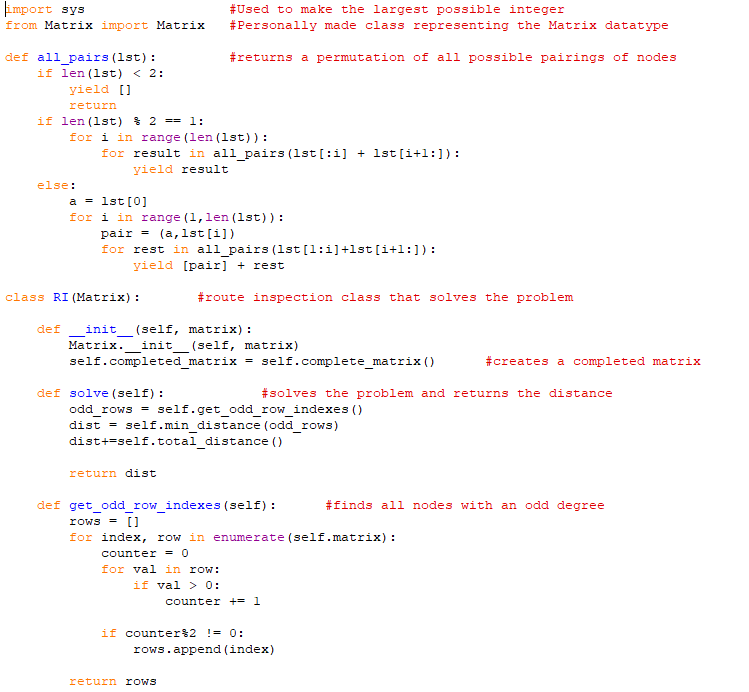
12.

Critical Path Analysis GUI is the screen which is showed to users to input a Critical Path Analysis problem to be solved

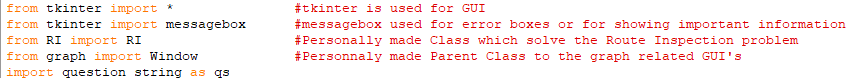
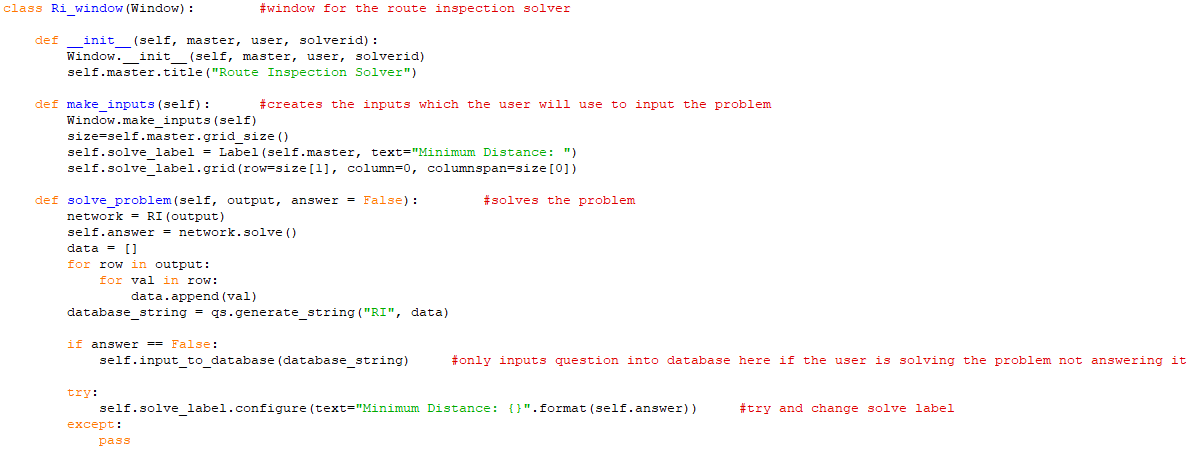
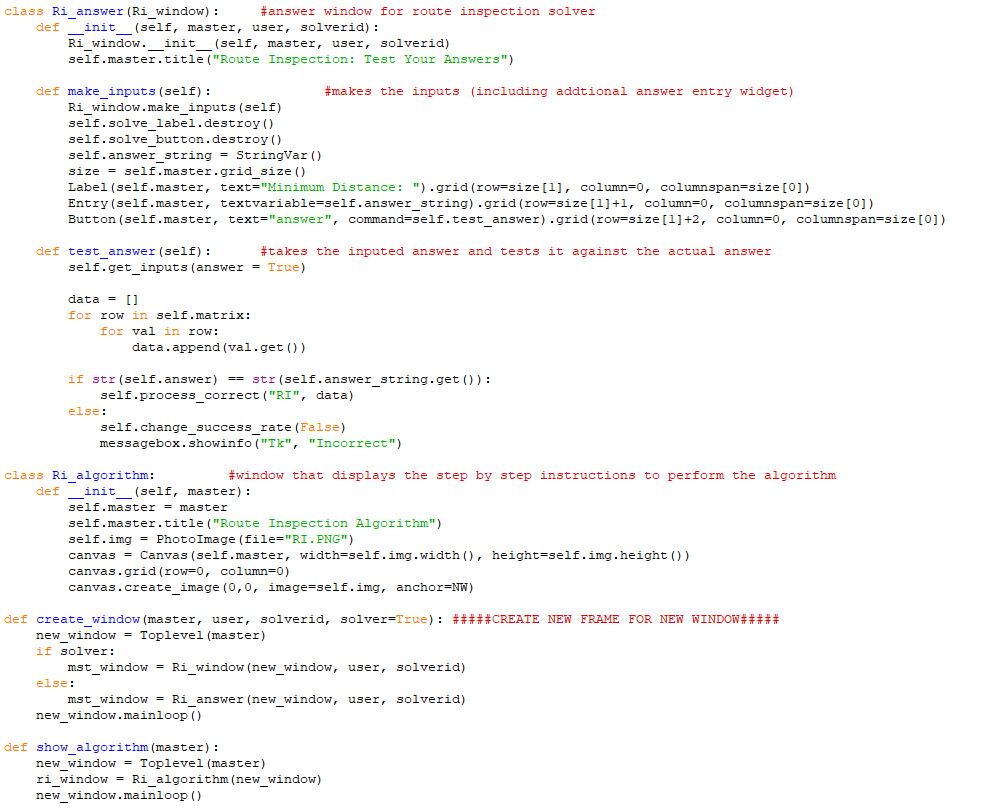
13.

The Route Inspection solver (RI.py) is used to solve route inspection problems inputted by the user.



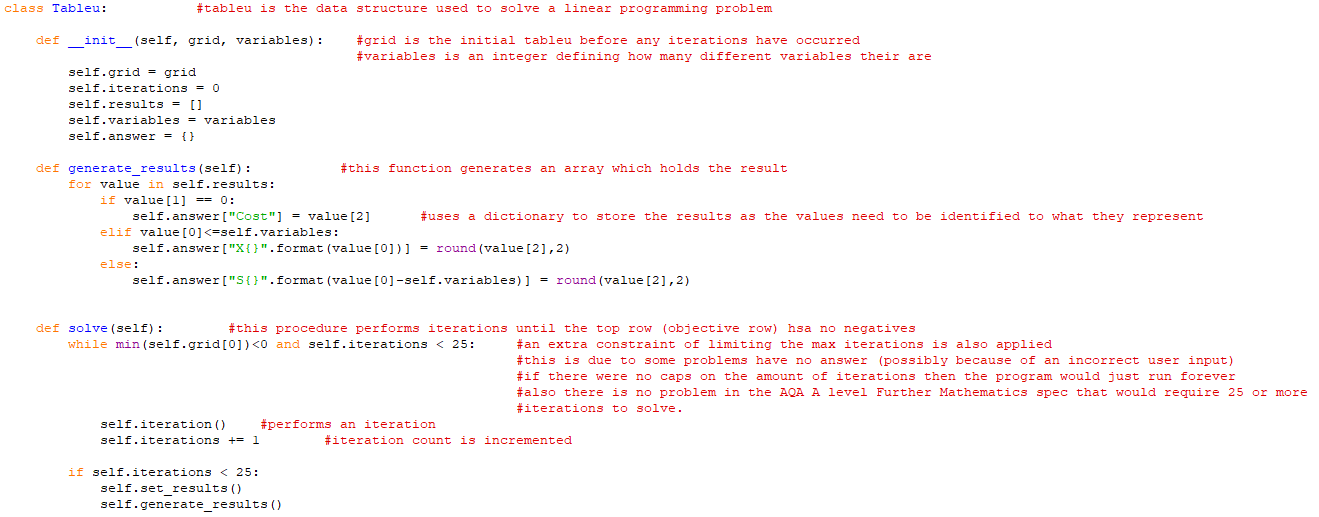
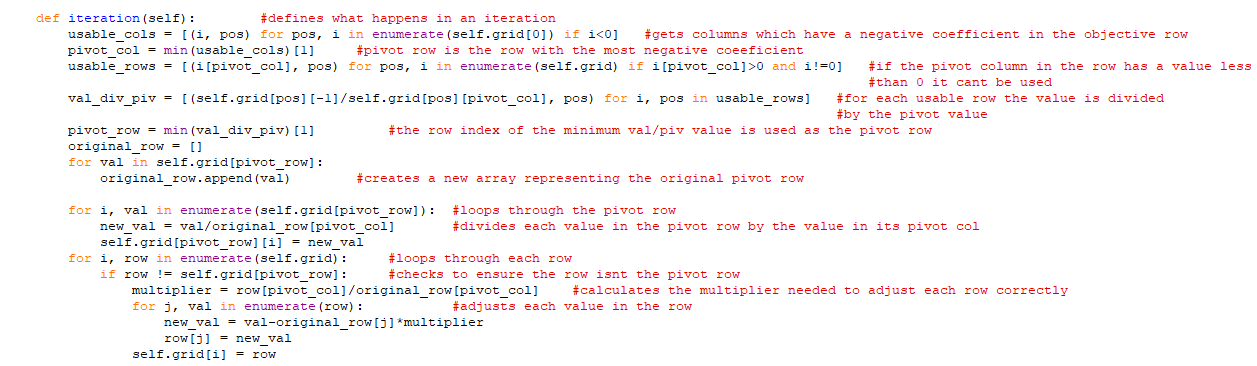
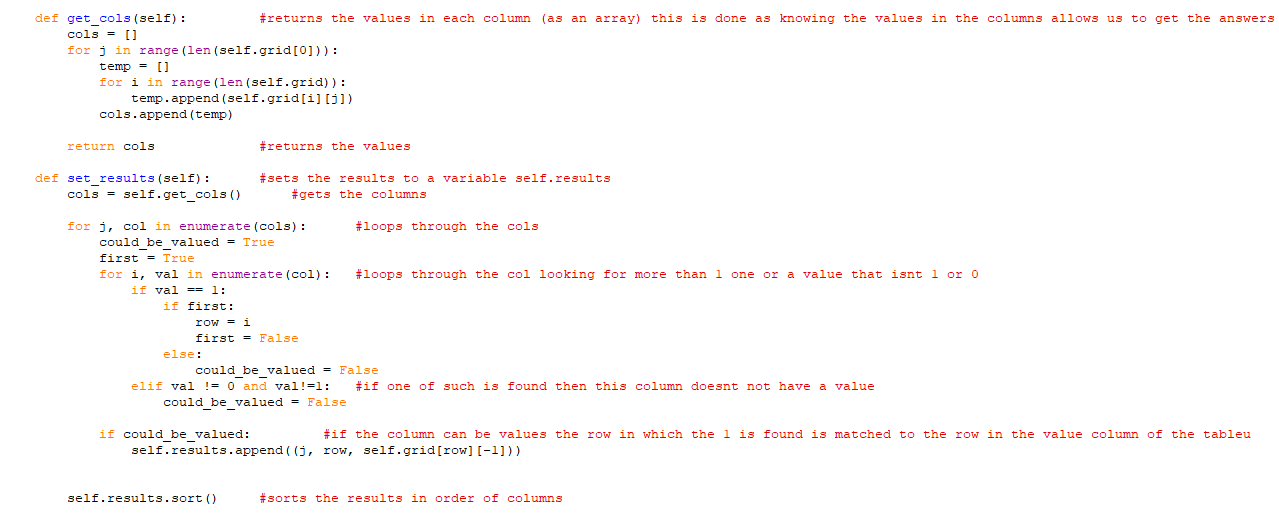
14.

Route Inspection GUI (RI\_gui.py) file is a subclass of the graph GUI screen. Some minor changes are made to specialise it for the Route Inspection problem.

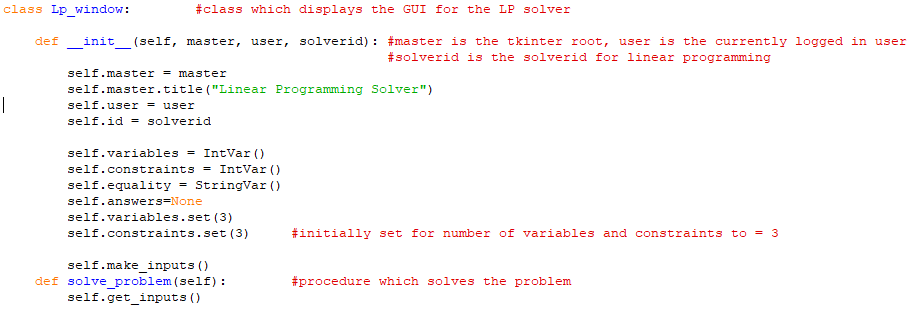
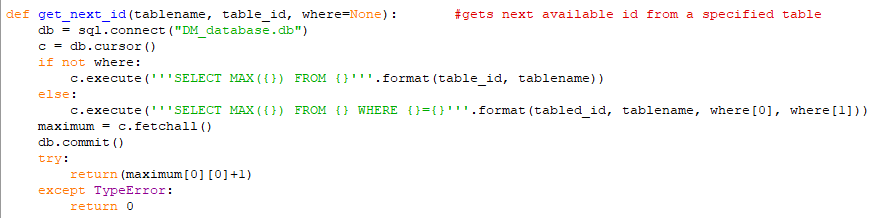
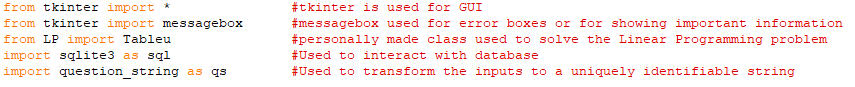
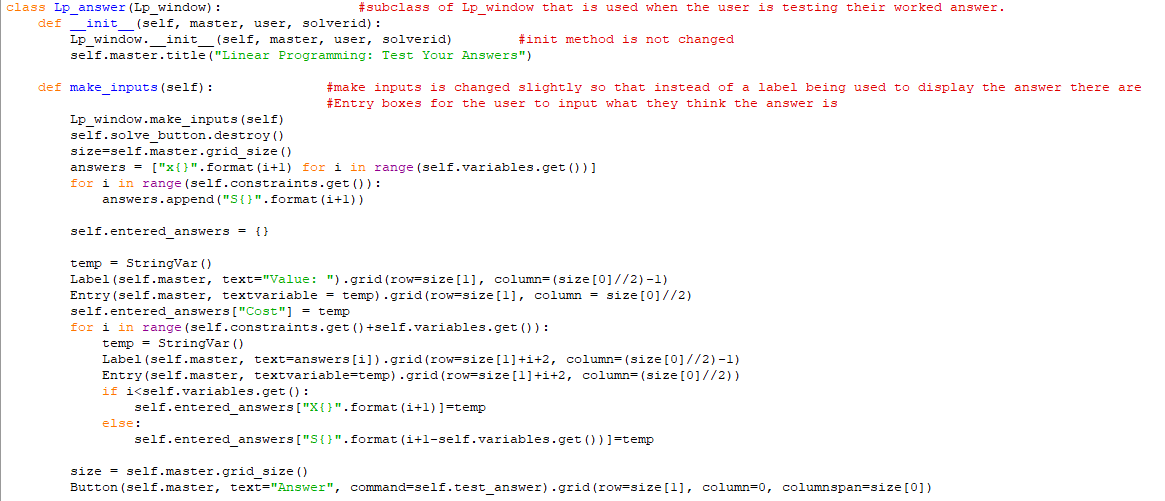
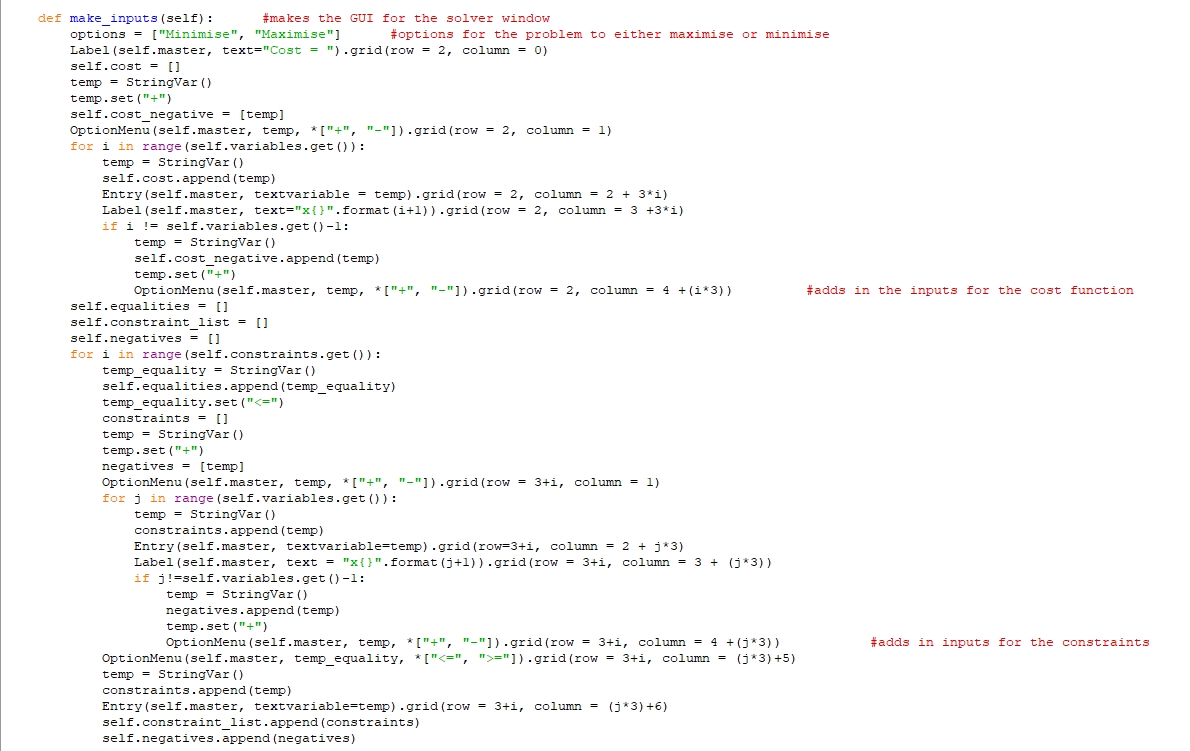
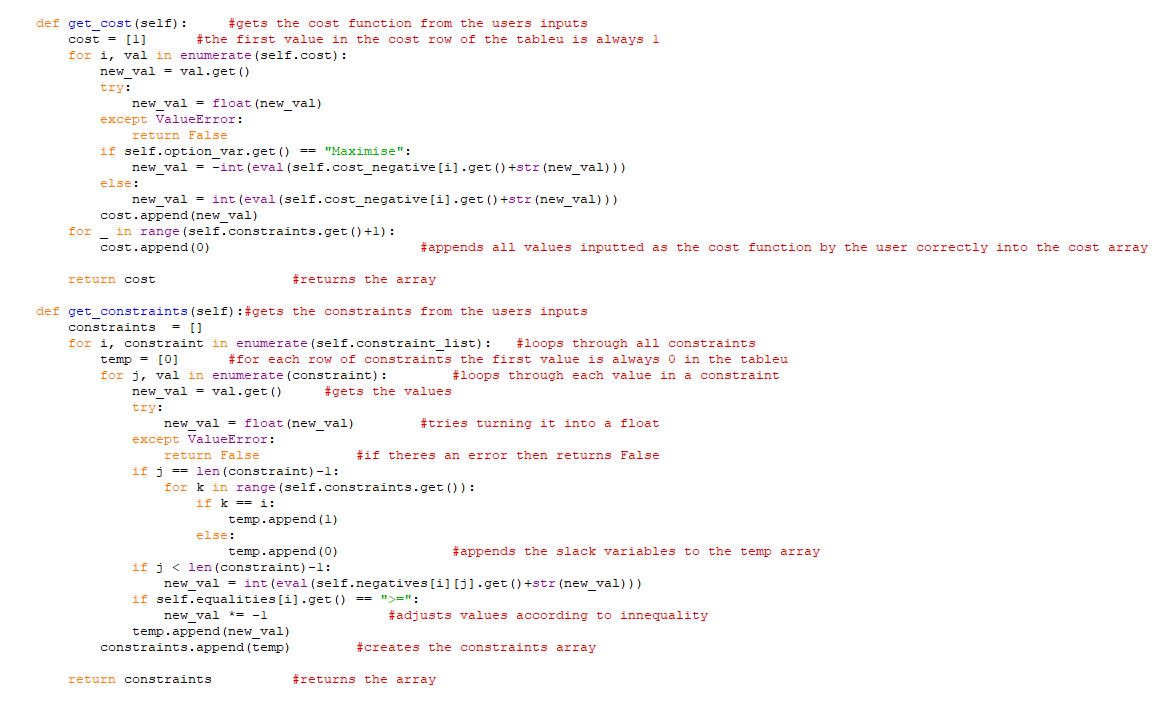
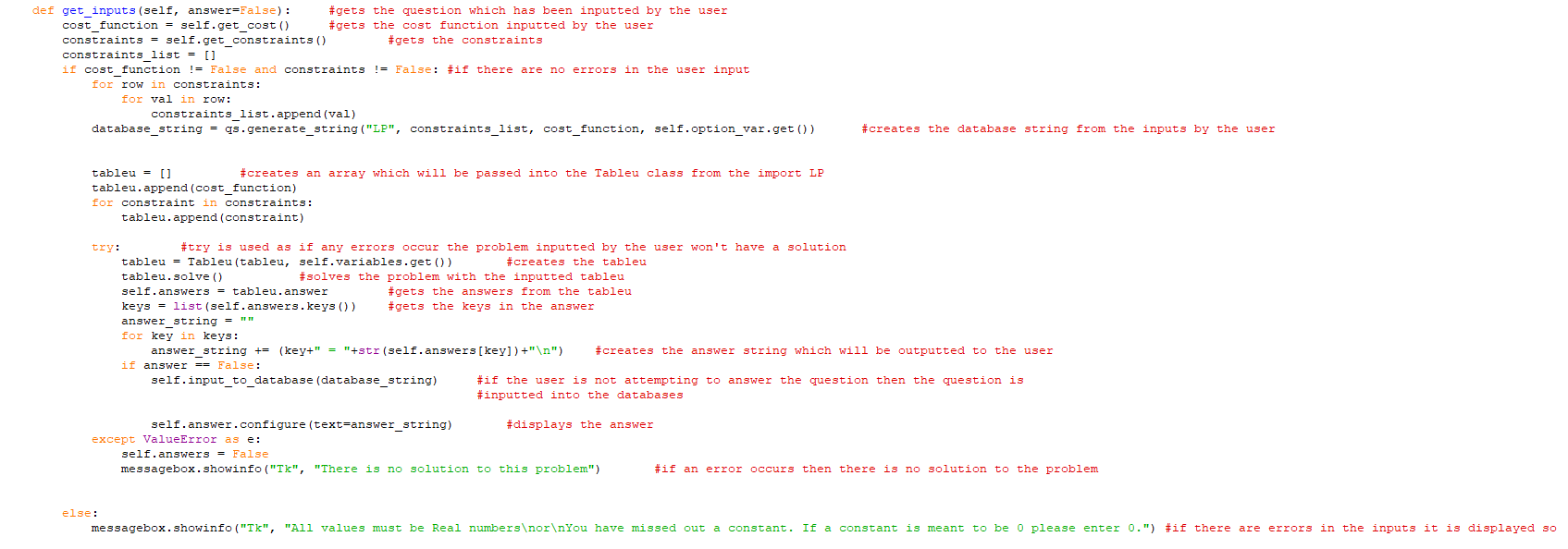
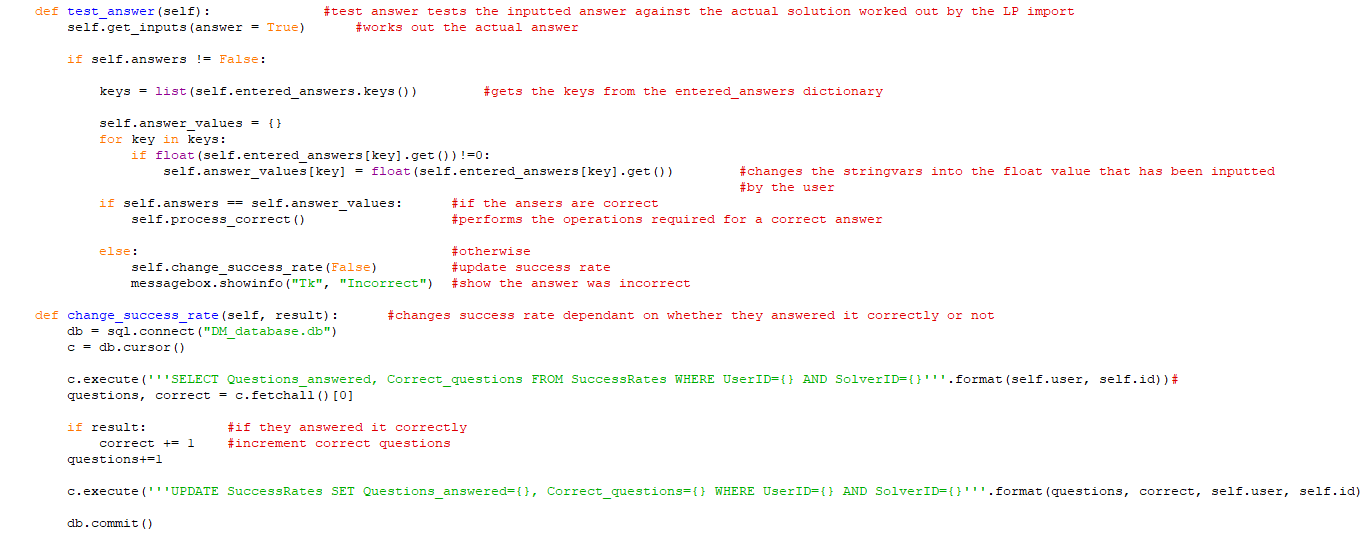
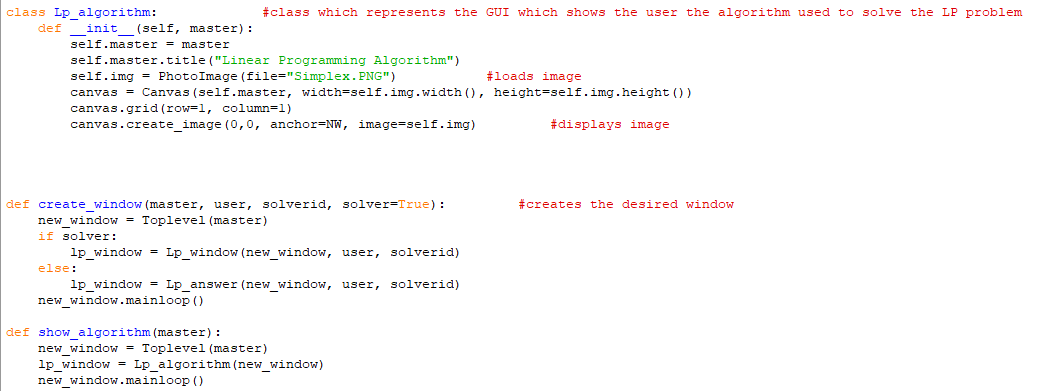
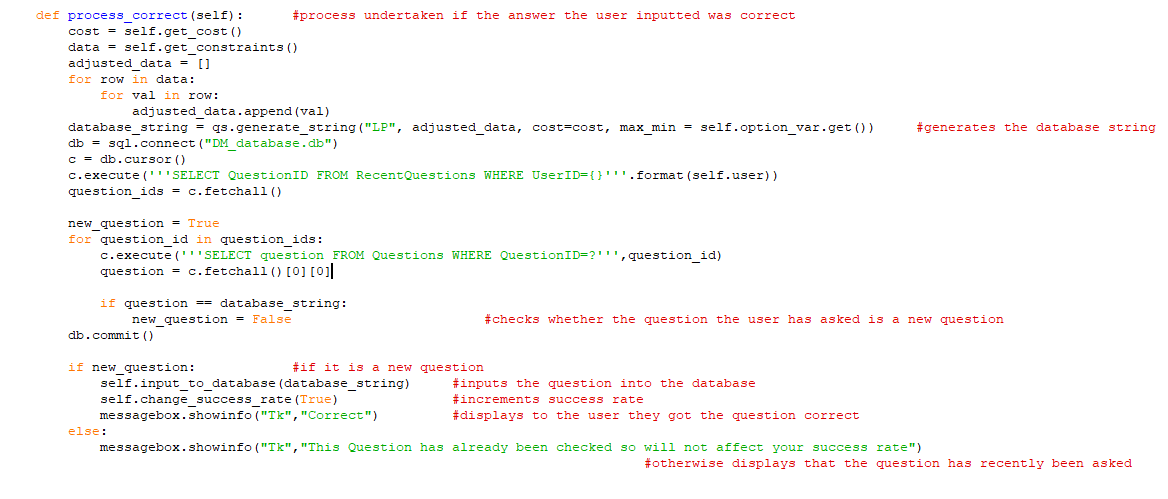
15.

Linear Programming solver (LP.py) is the file which solves linear programming problems inputted by the user.

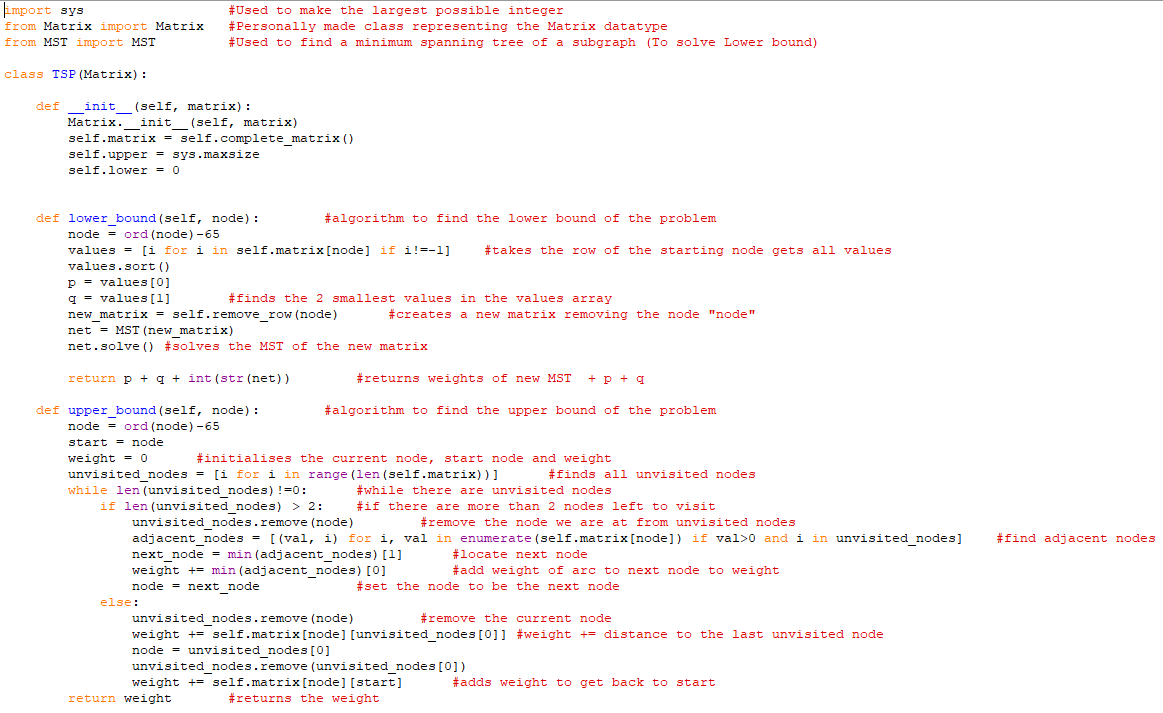
 

16.

Linear Programming GUI (LP\_gui.py) is a GUI for the user to input the Linear Programming problem into

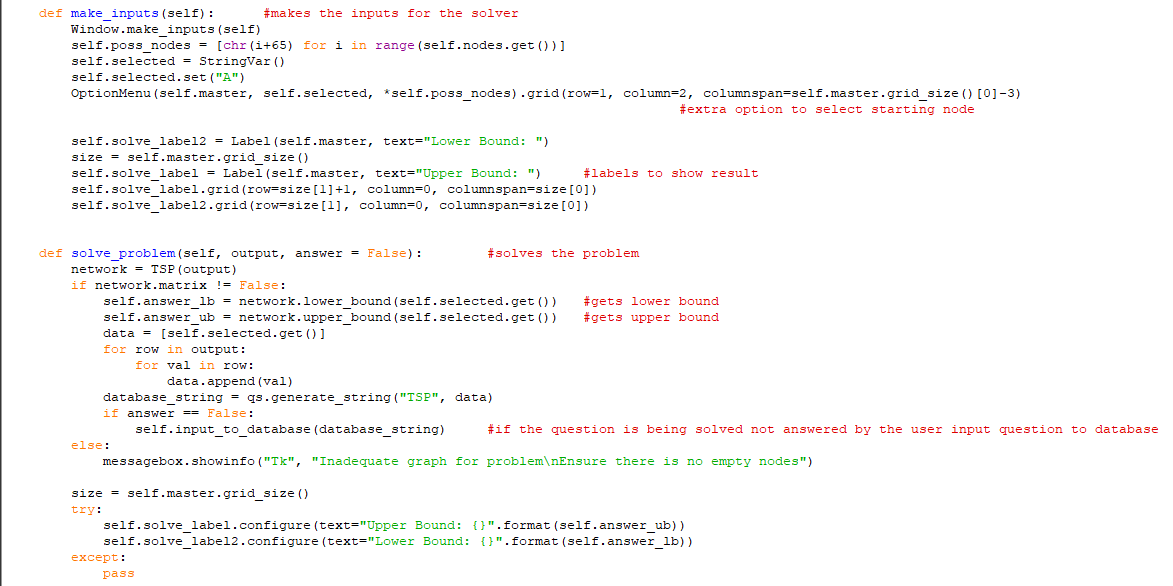
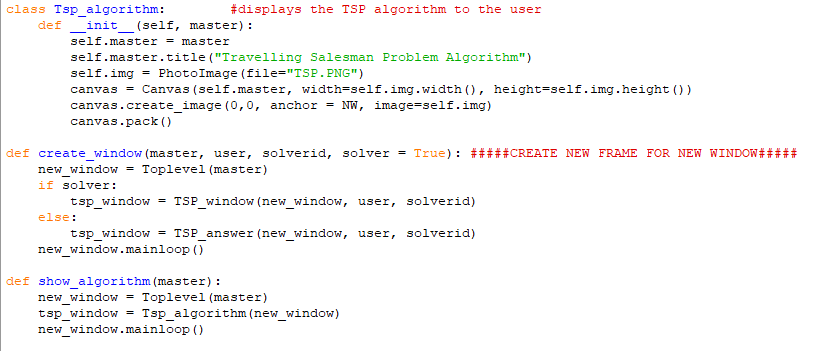
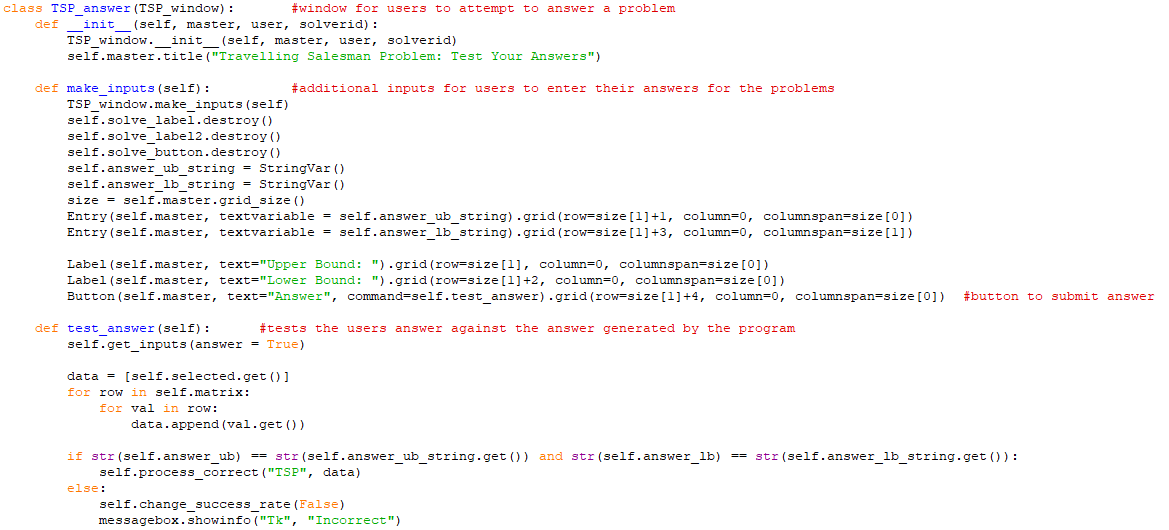
   

17.  
Travelling Salesman Problem solver (TSP.py) is the file used to solve for the upper and lower bound of the travelling salesman problem. This is due to the problem being intractable so a heuristic method must be used to solve it time efficiently



18.

The Travelling Salesman GUI (TSP\_gui.py) file is a subclass of the graph class. Some adjustments are made in this class to make it specific to the Travelling salesman problem.

# Testing

## Testing Plan

1. First run of program
   1. Ensure database is created correctly
   2. Ensure no extra information is added to the database if the program is reopened multiple times before the admin account is created
   3. Ensure the credentials inputted by the admin are stored correctly into the database ensure password is hashed and salt is used
      1. Expected data
         1. 2 strings consisting of any character’s password must be 8 or more characters
      2. Borderline data
         1. Password consisting of exactly 8 characters
      3. Erroneous data
         1. Either of the string being empty should not be allowed
2. First log on for Admin
   1. Ensure the program will only log in with correct credentials
      1. Expected data
         1. The username and password created in step 1 of the test
      2. Borderline Data
         1. Correct Username or Password but not both
      3. Erroneous data
         1. Both password and username incorrect
   2. When logged on should display there are no classes made yet and should be an option for the admin to create a class or create an Admin
3. Admins Creating a new Admin
   1. Ensure correct window is opened
   2. Test for valid passwords
      1. Expected data
         1. Password is 8 characters or longer and a non-empty string is used for the Username
      2. Borderline data
         1. Password is 8 characters long
      3. Erroneous data
         1. Password is less than 8 characters or no string is inputted for the username
4. Admins Creating a new class
   1. Ensure when Create Class button is clicked the correct new window is opened
   2. Enter in a class name and attempt to create the class
      1. Expected data
         1. An unused class name.
      2. Borderline data
         1. A class name that is already in use
      3. Erroneous data
         1. No name inputted
   3. Ensure the data for the new class is inputted to the database and that the new class is now accessible to the admin
5. Admins opening classes to view data
   1. Ensure the correct class has been opened
      1. Expected data
         1. A class being selected in the dropdown menu
      2. Borderline data
         1. No class being selected
   2. Ensure all 5 tick box options are available
   3. Ensure there is an option to add a new student
6. Admins adding a new student
   1. Ensure the correct new window has been opened to create a new user
   2. Enter in a username and attempt to create a user
      1. Expected data
         1. An unused username
      2. Borderline data
         1. A username already being used
      3. Erroneous data
         1. Empty string
   3. Ensure the data has been added to the database and that the user is accessible to the admin
   4. Ensure a password is given for the new user
7. Students log on for the first time
   1. Shows that given password from previous test will allow the user to log in
   2. Change password
      1. Expected data
         1. A string of at least 8 characters string consisting of any characters
      2. Erroneous data
         1. An empty string
   3. Ensure data is updated in database
   4. Ensure changed password works for the user
8. Students using Main Menu
   1. Test each button works and shows the correct window
9. Students using a solver (same test for each solver)
   1. Data input
      1. Expected data
         1. Integers or empty cells (depending on problem)
      2. Borderline data
         1. Data that is of the correct type (integers/floats) but the inputted data will not give a solution (proved by handwritten solution)
         2. For CPA Link Activities to activities to future activities.
      3. Erroneous data
         1. Characters instead of integers
   2. Ensure answer given by the inputted problem is correct. Testing it against a correct answer
   3. Ensure question solved is inputted to the database
10. Students attempting a question (same test for each solver)
    1. Testing answer
       1. Expected data
          1. Correct answer
       2. Borderline data
          1. Incorrect answer
    2. Test whether the question has recently been asked by answering a recently asked question correctly
11. Admins checking user’s success rates
    1. Ensure correct success rates are shown by working out what they should be by attempted questions in previous test
12. Admins Deleting students
    1. Ensure an “are you sure” message appears
    2. Ensure the student disappears from the GUI
    3. Ensure student disappears from the database
13. Admins changing available solvers
    1. Change tick boxes and click the save button. Ensure the changes are made to the database
14. Admins changing Users password
    1. Testing data
       1. Expected data
          1. 8 character or longer string
       2. Borderline data
          1. String of 8 characters length
       3. Erroneous data
          1. String less than 8 characters length
15. Check how students are affected by test 13 and 14
    1. Log in as a user using old password followed by new password (made in test 14)
    2. Show how the corresponding solvers changed in test 13 are now unavailable to the user

## Link to testing video

# Evaluation

## Final Questionnaire

Before answering the questionnaire, the Users/Clients have been allowed to test the program and use it to answer questions.

For both clients and users:

1. Did the solvers answer the questions correctly?
2. Were any of the solvers hard to understand/use? If so, explain why.
3. Were the algorithms available to view useful in solving problems?
4. Any other comments on the product.
5. What would you change about the product?

For just Clients/Admins:

1. Was the admin menu and subsequent windows easy to use and navigate?
2. Was it easy to create classes?
3. Was it easy to create students?
4. Was it easy to view user’s success rates?
5. Is being able to see user’s success rates useful?
6. Were there any problems you encountered in using the program?
7. Any other comments on the Admin side of the product.

For just Users:

1. Do you believe this program would be useful in studying for your Further Mathematics A-Level course?

## Questionnaire results and review

### Helen Morgan

1. Did the solvers answer the questions correctly?  
   Yes.
2. Were any of the solvers hard to understand/use?  
   No, the solvers were laid out in an easily identifiable way to how Further Mathematicians would solve the problem.
3. Were the algorithms available to view useful in solving problems  
   Yes, the algorithms supplied were sufficient to work through to solve the problems.
4. Any other comments on the product?  
   Include Colour to make the product have a nicer look. Also increase the size of everything to make it easier to use.
5. What would you change about the product?  
   Same as above and also include a way to reset an Admin’s password.

Admin specific questions

1. Was the admin menu and subsequent windows easy to use and navigate?  
   Yes, I was able to navigate through all the windows without any help.
2. Was it easy to create classes?  
   Yes.
3. Was it easy to create students?  
   Yes.
4. Was it easy to view user’s success rates?  
   Yes.
5. Is being able to see user’s success rates useful?  
   Yes, it allows me to know where my students are struggling and can then give them specific tasks to help them improve in that area.
6. Were there any problems you encountered using the program?  
   No.
7. Any other comments on the Admin Side of the product?  
   Again, would be nice to include a method of admins to reset their password in case they forget it.

### Karen Tunnicliff

1. Did the solvers answer the questions correctly?  
   Yes.
2. Were any of the solvers hard to understand/use?  
   No, the solvers were laid out in an easily identifiable way to how Further Mathematicians would solve the problem.
3. Were the algorithms available to view useful in solving problems  
   Yes, the algorithms supplied were sufficient to work through to solve the problems.
4. Any other comments on the product?  
   Include Colour to make the product have a nicer look. Also increase the size of everything to make it easier to use.
5. What would you change about the product?  
   Same as above and also include a way to reset an Admin’s password.

Admin specific questions

1. Was the admin menu and subsequent windows easy to use and navigate?  
   Yes, I was able to navigate through all the windows without any help.
2. Was it easy to create classes?  
   Yes.
3. Was it easy to create students?  
   Yes.
4. Was it easy to view user’s success rates?  
   Yes.
5. Is being able to see user’s success rates useful?  
   Yes, it allows me to know where my students are struggling and can then give them specific tasks to help them improve in that area.
6. Were there any problems you encountered using the program?  
   No.
7. Any other comments on the Admin Side of the product?  
   Again, would be nice to include a method of admins to reset their password in case they forget it.

### User 1

1. Did the solvers answer the questions correctly?  
   yes, the solvers did successfully answer all questions I asked successfully
2. Were any of the solvers hard to understand/use?  
   All but the linear programming solver. It wasn’t obvious where the button was to change the program from a minimise to maximise problem. Also, as most questions are maximise it should initially be on maximise.
3. Were the algorithms available to view useful in solving problems  
   Yes, I was able to follow the algorithms and they worked in solving the problem.
4. Any other comments on the product?  
   No
5. What would you change about the product?  
   Include colour

User Specific Questions

1. Do you believe this program would be useful in studying for your Further Mathematics A-Level course?  
   Definitely. Often, the textbook we use does not have the correct answers. This allows us to check our answers and it be correct.

### User 2

1. Did the solvers answer the questions correctly?  
   Yes.
2. Were any of the solvers hard to understand/use?  
   No, I was able to understand all of the solvers
3. Were the algorithms available to view useful in solving problems  
   Yes.
4. Any other comments on the product?  
   Could include more problems. Discrete mathematics has a lot more topics than included here. It would be nice for it to be able to do everything. Maybe even expand into the Pure, Mechanics and Stats part of the course.
5. What would you change about the product?  
   Include more solvers/problems.

User Specific Questions

1. Do you believe this program would be useful in studying for your Further Mathematics A-Level course?

Definitely. Even though it is limited in what it can help you with, the parts it does have solvers for is extremely helpful in checking answers and learning how the algorithms work

## Commenting on criteria in objective order stated in Analysis

1. **Correctly solve all 5 problems:**The testing video shows each solver correctly answering a question correctly. This however wasn’t enough for me to claim I have successfully made a program that will solve the problems. Because of this I gave my program to a few users and got them to test a couple questions each for me. All questions the users asked got a correct answer given by the program. Therefore, this objective has successfully been achieved by this program. I have a few problems with how I implemented the algorithms in my code. One problem comes with the TSP Upper Bound algorithm. I have followed the algorithm stated in the AQA A Level Further Mathematics textbook. However, this algorithm doesn’t state what to do if 2 available nodes have the same minimum weight on the arc to them. So, the algorithm goes to whichever has the smallest index (e.g. A comes before B) this means the Upper Bound calculated may not be the smallest upper bound from that node. There are ways around this; I could have used a recursive algorithm to check multiple routes. However due to time restraints I did not have time to do this.
2. **Using Tkinter to create a GUI for the program:**I used Tkinter for the entirety of the programs User Interface. I believe the GUI I have created is rather intuitive, especially for an A-Level Further Mathematician. All needed windows were created. These Include: login Menu, Admin Menu, Class creation screen, Admin creation screen, Class editor screen, User creation screen, User viewer screen, A Main Menu, Individual windows for each problem (opened when selected from main menu). My clients and intended users both agreed that the GUI was easy to use, apart from the minimise/maximise dropdown menu in the Linear Programming solver. They also stated they thought the inclusion of colour to the GUI may have been beneficial.
3. **A logon system for Users and Admins:**I successfully created a working log in system for this program. On first run of the system the Client can create an Admin account of which its username, password (encrypted and appended with a salt) and other details will be stored into the database. When logging in the Admin will put in these credentials, SQL statements are used to check the Username and password are the same as what is stored in the database and if so logs them in to their account. The program also checks that all usernames are unique and that no username can be an empty string. When an Admin creates a User, they are given the new users password to give to the User so they can access the program. The user is then able to log in with this temporary password and change it to a password they will remember. A weakness with the logon system is there is no way for an Admin to reset their password if they forget it. My clients brought this to my attention and mentioned how it would be a good bonus feature so if Admins do forger their passwords all of the data they should have access to isn’t lost. This could be implemented using a password recovery via email. This would, however, need a mail server to be used for emails to be sent to users of the program.
4. **Database:**I am pleased with the structure of the database even with the problems covered in the Documented Design. Even though the database does not by word follow the rules to be in 3rd Normal form I believe it acts as if it is normalised. The normalisation I have done has reduced the amount of redundant data being used. From the clients and user’s perspective the database works as required. It saves the state of the solvers being locked or unlocked; it saves the users success rates at each individual solver; It stores the users log on details.
5. **Locking Solvers:**This functionality of the program was implemented in an attempt to stop the clients’ users from cheating on homework etc. An Admin can block certain solvers from being used for specific classes. When showing this functionality to the client they thought it was a useful aspect. Another comment they had however was they thought it would be useful to block solvers from individual users not just entire classes. This would be achievable by moving the checkboxes into the Screen where the user’s success rates are seen. On top of this the database would need editing so the AvailableSolvers table would have the UserID as a field instead of Class\_name. Some adjustments would also need to be made to the SQL queries used to update and check the AvailableSolvers table.
6. **Algorithms available for users to look at and learn:**This section of the program is very useful in studying for Exams/tests. Users can see, learn and remember the algorithms used in the solvers to use in their exams. Client’s believe that the ability for the users to view the algorithms is very helpful in allowing them to learn the algorithms for their exams. However, they also mentioned how they would like it to be locked as well as the solver when they decide to block the specified solver. Personal adjustments I have considered after producing the product is including worked examples of each problem as student’s may not understand the written algorithms easily.
7. **Answer mode:**I believe this part of the program works well and my Clients also believe that the functionality of this section is extremely useful in learning where certain users need help with revision. Problems I have with the Answer mode is not with how it is used and seen by Users and Admins but in how the data is processed in the code. I find it displeasing that there isn’t a set way in which answers are formatted in the code. I.e. In the Linear Programming section answers are stored as a dictionary and in Critical Path Analysis the answers are stored as a long string. This problem could be solved by working on a unique way in storing answers similar to what was done in creating database strings. However, due to time constraints I was unable to do this.
8. **Ability for admins to change Users passwords:**The purpose of this functionality is for if a user forgets their password their corresponding Admin can change it to a password they will remember. My clients believed that this was useful as it is common for students to forget their passwords.
9. **Travelling Salesman Problem**I have covered some of my opinions on the travelling salesman problem in point 1. I have some minor problems with it however it does do as it should. The Travelling salesman problem is known as an intractable problem so no brute force algorithms can solve the problem efficiently enough to be used on a large scale. So instead Heuristic approaches are used. Heuristic approaches are ones that give a reasonable answer, but it may not be the shortest possible route. Therefore, I deduce that the algorithm I have used does give a reasonable answer therefore does act as a heuristic algorithm for the problem. On top of this the algorithm also reaches the same answer in which questions in practice papers and textbooks give.

# References

*12.3 AQA A Level Further Maths Year1 + Year 2, Kerboodle*. (2019, September 19). Retrieved from Kerboodle: https://www.kerboodle.com/api/courses/42061/interactives/244639.html

*12.4 AQA A Level Further Maths Year 1 + Year 2, Kerboodle*. (2019, September 19). Retrieved from Kerboodle: https://www.kerboodle.com/api/courses/42061/interactives/244639.html

*12.5 AQA A Level Further Maths Year 1 + Year 2, Kerboodle*. (2019, September 19). Retrieved from Kerboodle: https://www.kerboodle.com/api/courses/42061/interactives/244639.html

*12.6 AQA A Level Further Maths Year 1 + Year 2, Kerboodle*. (2019, September 24). Retrieved from Kerboodle: https://www.kerboodle.com/api/courses/42061/interactives/244639.html

*13.1 AQA A Level Further Maths Year 1 + Year 2, Kerboodle*. (2019, September 19). Retrieved from Kerboodle: https://www.kerboodle.com/api/courses/42061/interactives/244639.html

*30.1 AQA A Level Further Maths Year 1 + Year 2, Kerboodle*. (2019, September 16). Retrieved from Kerboodle: https://www.kerboodle.com/api/courses/42061/interactives/244639.html

*Critical path, Total float, Free float, Independent float calculator*. (2019, September 11). Retrieved from AtoZmath.com: https://cbom.atozmath.com/CBOM/PertCPM.aspx

*Djikstras shortest path algorithm - greedy Algo-7*. (2020, March 10). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

Fitzmaurice, J. (2019, September 12). Prims Algorithm Example. *Prims Algorithm Example*. Bedford, Bedfordshire, England: Joshua Fitzmaurice.

*Simplex method calculator*. (2019, September 10). Retrieved from AtoZmath.com: https://cbom.atozmath.com/CBOM/Simplex.aspx?q=sm

*sqlDBM*. (2019, November 1). Retrieved from sqlDBM: https://app.sqldbm.com/MySQL/Draft/