System Maintenance

**System overview**

This system was designed in Visual Basic. Net 2010 express on computers at both Yeovil College and my home which had either Windows XP or Windows 7 installed.

Modular structure

Main\_ Menu

Kruskal’s algorithm

Timed\_ Run

Run

Teacher’s section

Quit

Teacher\_Login

Times\_ Achieved

Quit to Main Menu

Update\_Kruskal

Teacher\_Menu

Forename\_Entry

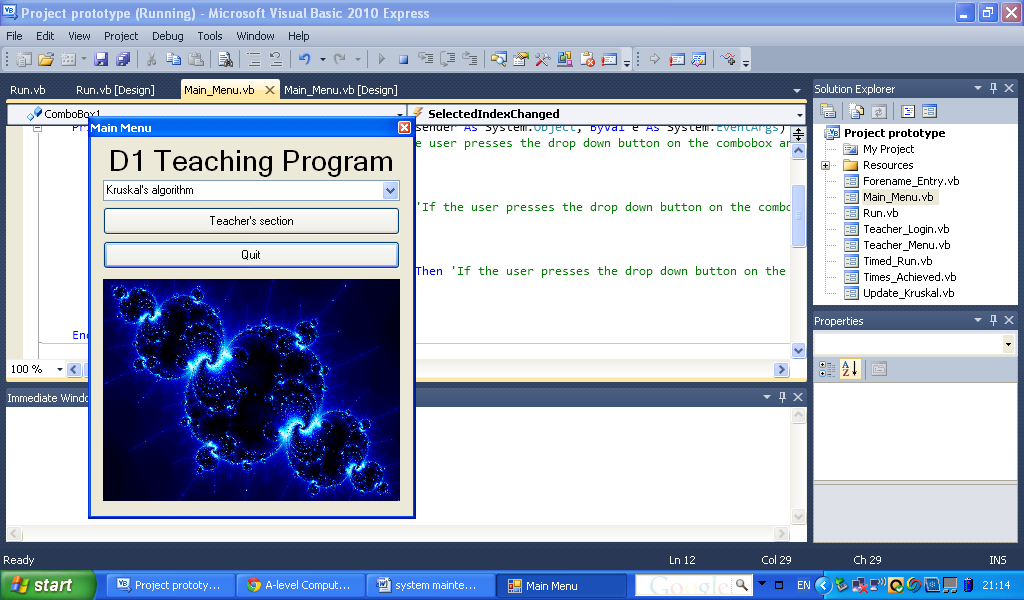
Forms

All of the message boxes displayed by the program have “D1 teaching program” as a header. This and the other aspects of my program all work as they are explained below; which was proved by my system testing section.

Main\_Menu

Description:

This form is the Main Menu and it is the page that every user will see immediately once the program has been opened. It is designed to be easy on the eye, easy to understand and compact.

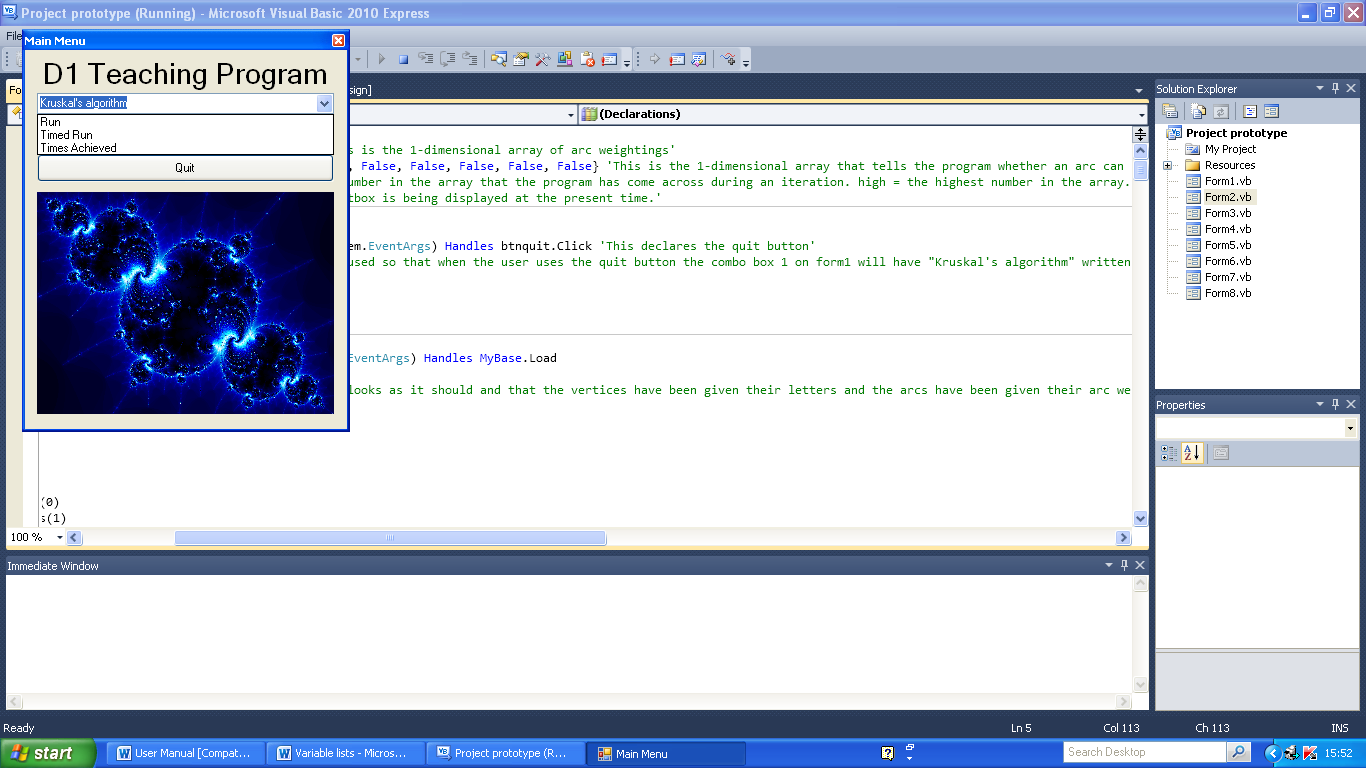
Screenshot:

This combo box displays three choices to select when the drop down menu button is pressed.

This is the “Teacher’s section” button which takes users to the Teacher\_Login form when pressed.

This quits the program when pressed.

This picture box contains a fractal which is used in Maths, however in my program it is just used to make my program look more exciting and appealing.

Combo box:

When this line is pressed the user is taken to the Run form.

When this line is pressed the user is taken to the Timed\_Run form.

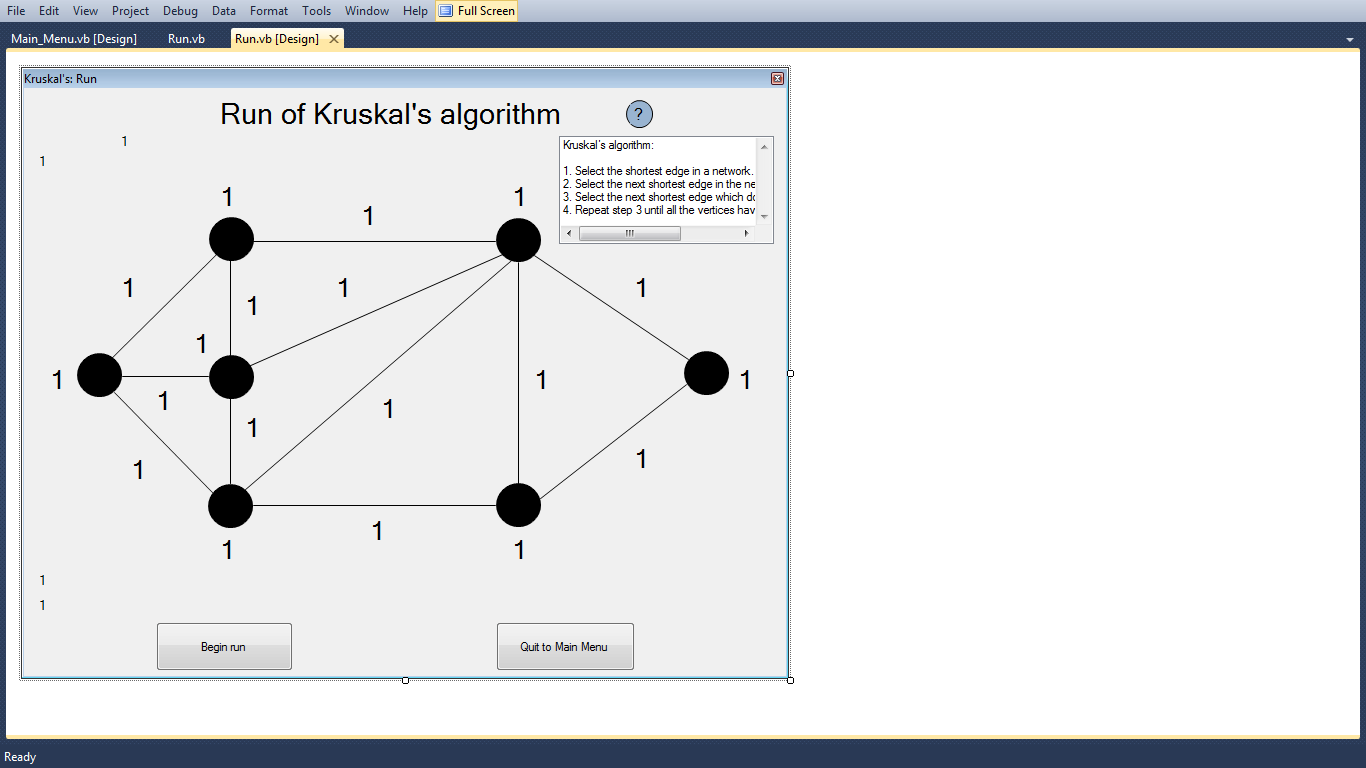
When this line is pressed the user is taken to the Times\_Achieved form.

Run

Description:

This form is shown when the user selects the Run option in the Kruskal’s algorithm combo box. It performs a run of the algorithm on a set weighted network where the same stages are performed and the same minimum connector is produced each time.

Screenshot:



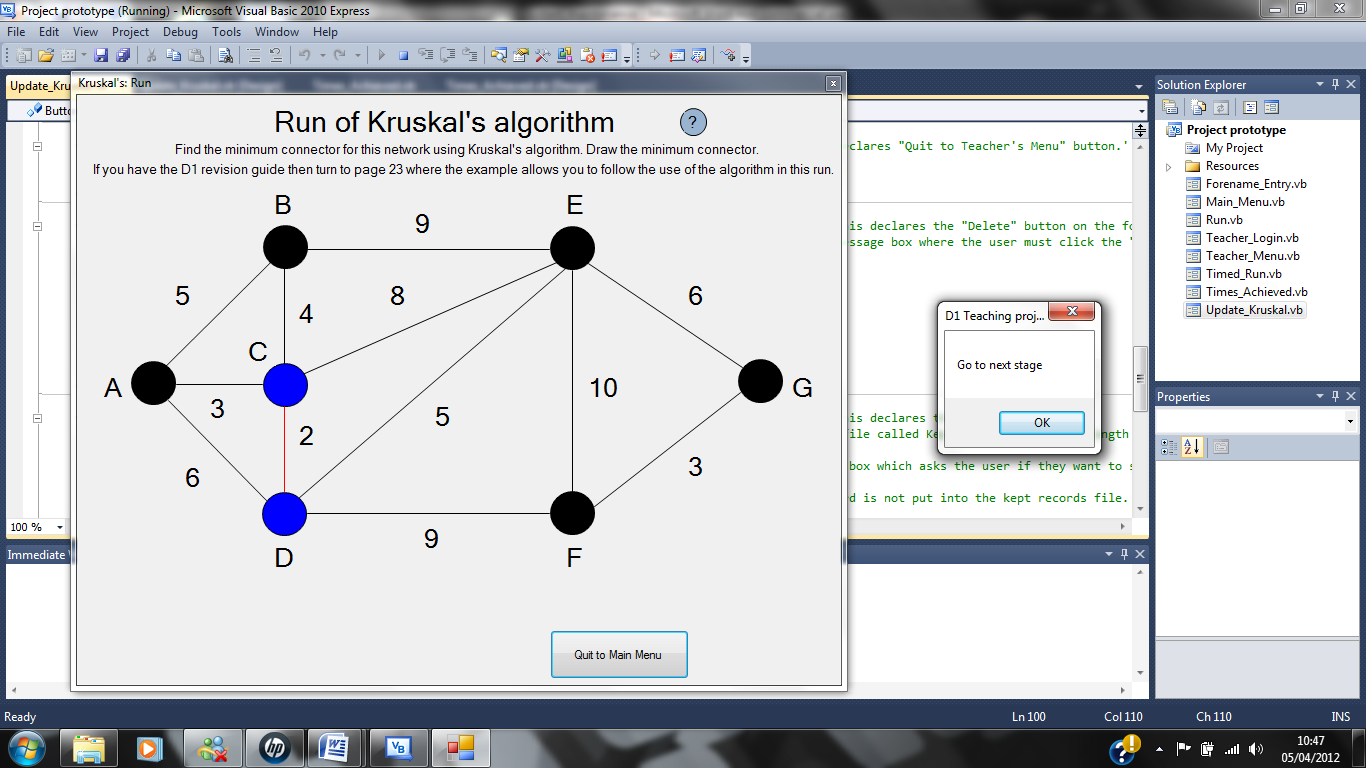
If the list box is not currently displayed and this button is pressed then the list box is displayed. If the list box is currently displayed then the opposite happens. The list box displays the step by step method of Kruskal’s algorithm.

This is the set network. The set weightings are displayed in their labels once the form is loaded.

I have put 1’s in the labels so that I can see them; this makes the form easier to design.

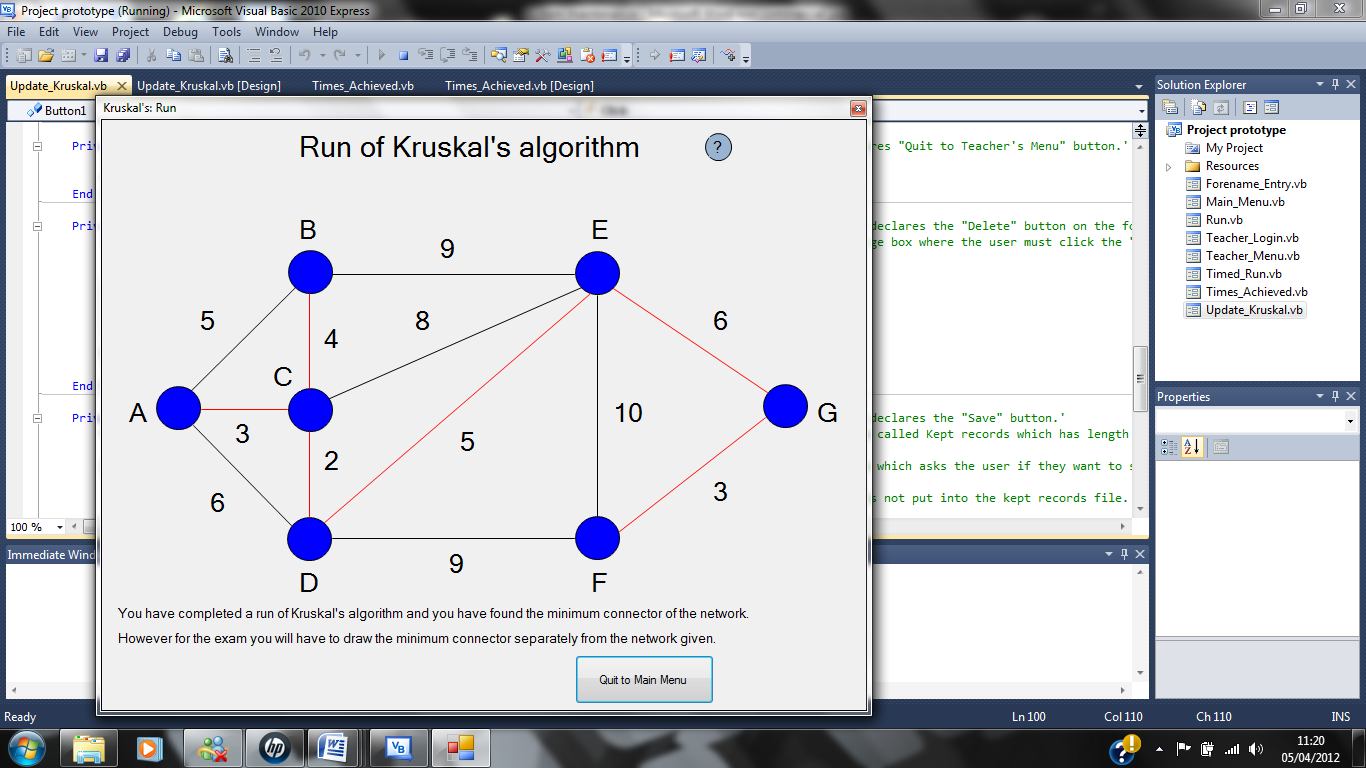
When this button is pressed the Main\_Menu form is displayed.

When this button is pressed the run of the algorithm is started and the arcs begin to be coloured. This button disappears after being pressed.

Message boxes/messages:

This message box is displayed after the user presses the “Begin run” button and after every subsequent stage of the algorithm.

This button needs to be pressed by the user to move onto the next stage.

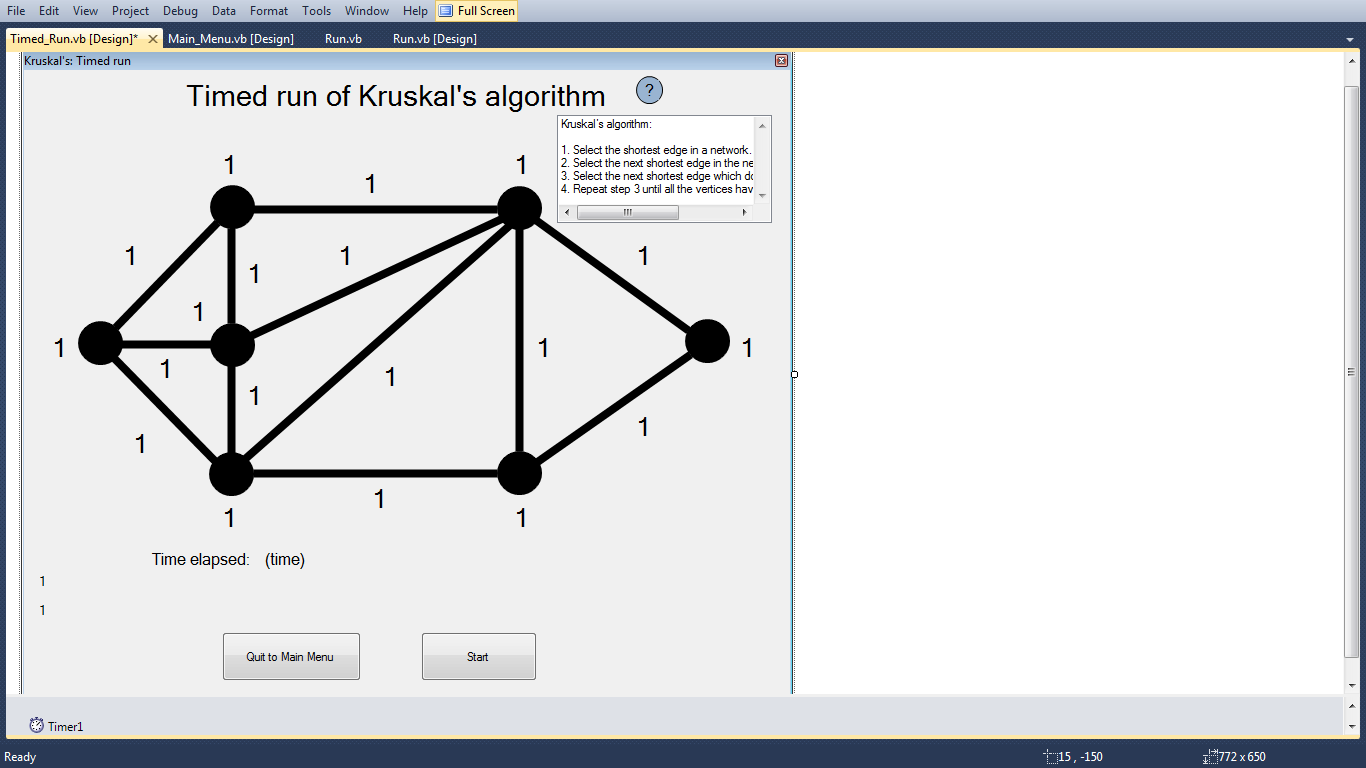


This is displayed when the user has completed a run of Kruskal’s algorithm. It tells the user that they have completed the run and what they will really have to do in the exam.

Timed\_Run

Description:

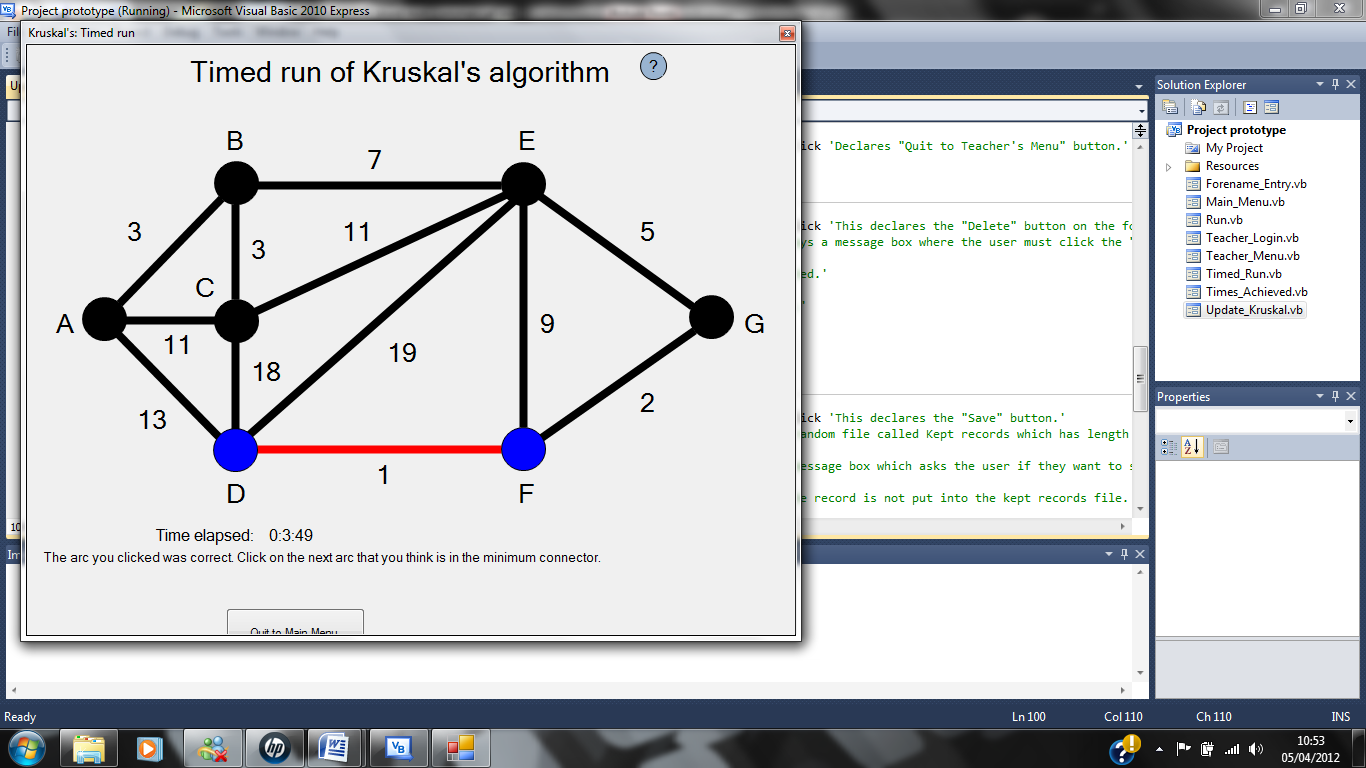
This form is displayed when the user clicks on the Timed Run option on the combo box, like the Run form it uses the same network but the arc weights are randomised and the arcs have been made thicker so that they can be clicked on easier. Also the same help button is used on this form as in the Run form. The program performs the algorithm once the “Start” button has been pressed where it then works out which arc needs to be selected at each stage. If the arc clicked by the user at each stage is the same as these then the arcs will turn red indicating that they are in the minimum connector and the user has clicked on the correct arc. The time that has elapsed since the user pressed the “Start” button is shown on the form to the user; the stopwatch stops once the minimum connector is shown on screen. Once the complete minimum connector is shown on the screen a message box will be displayed that tells the user that they have completed the timed run; after the user has pressed the “OK” button on the message box the Forename\_Entry form is displayed to the user. The user fails a timed run if they don’t complete it in under 10 minutes because if you were to do this in an exam then you would more likely not have time to do the rest of the questions. The user is notified when their time limit is up using a message box; which states that they have failed the timed run. After they press “OK” on the message box the Main\_Menu form is displayed to the user.

Screenshot:

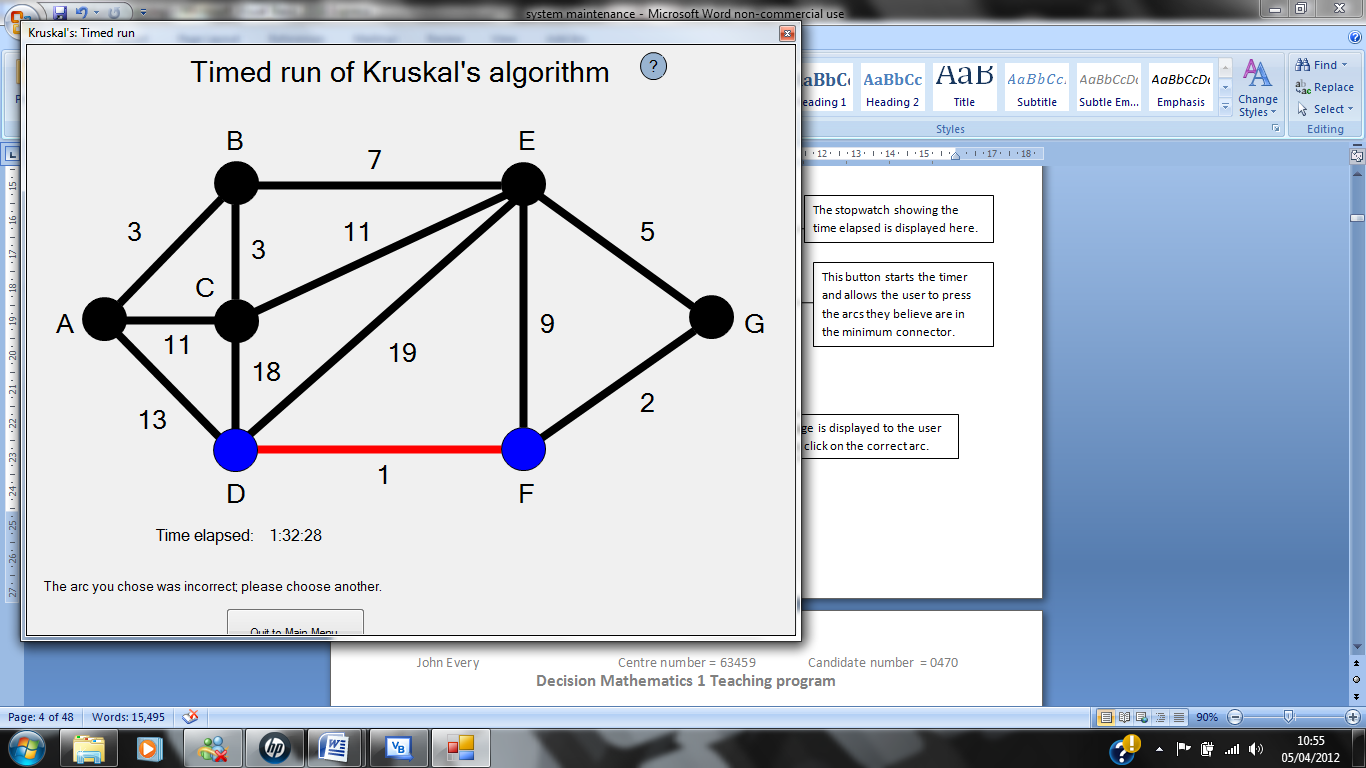
The stopwatch showing the time elapsed is displayed here.

This button starts the timer and allows the user to press the arcs they believe are in the minimum connector.

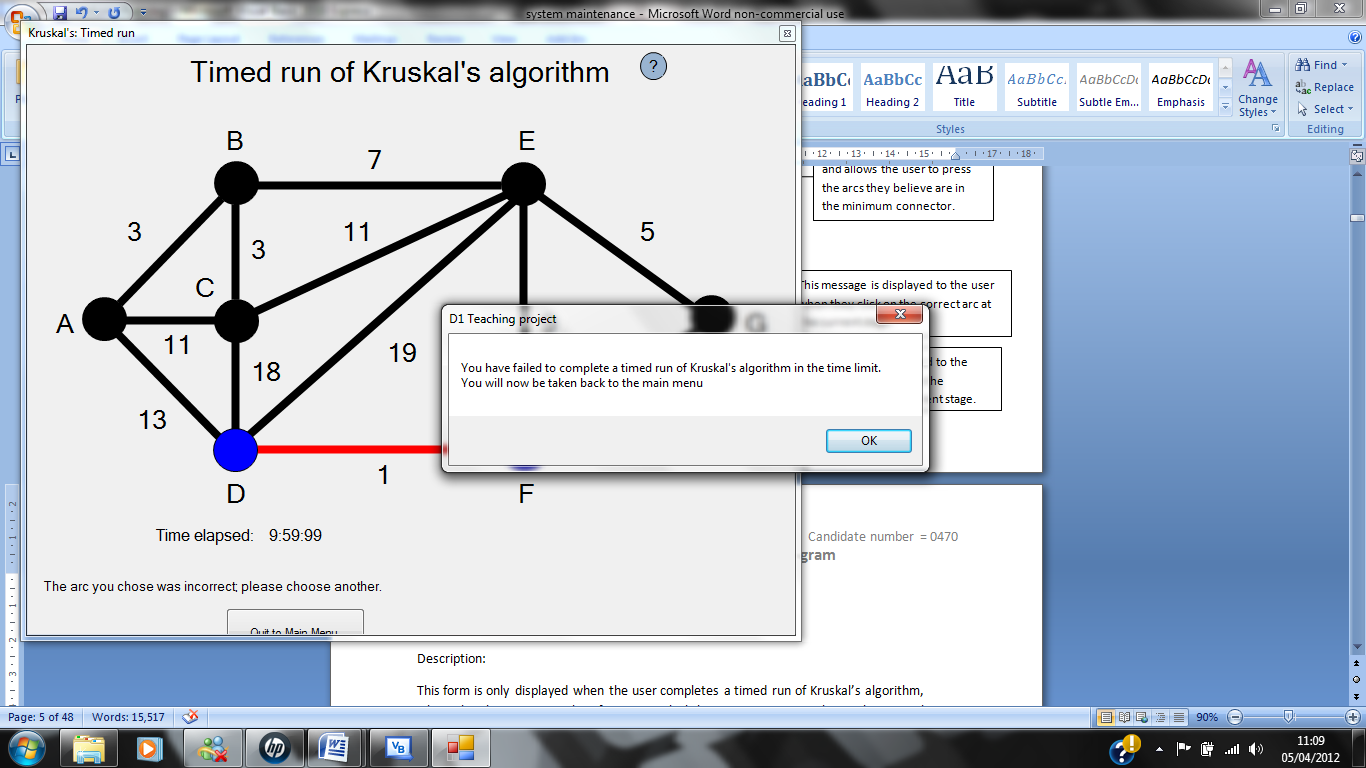
This button closes this form and displays the Main\_Menu form when pressed.

Message boxes/ messages:

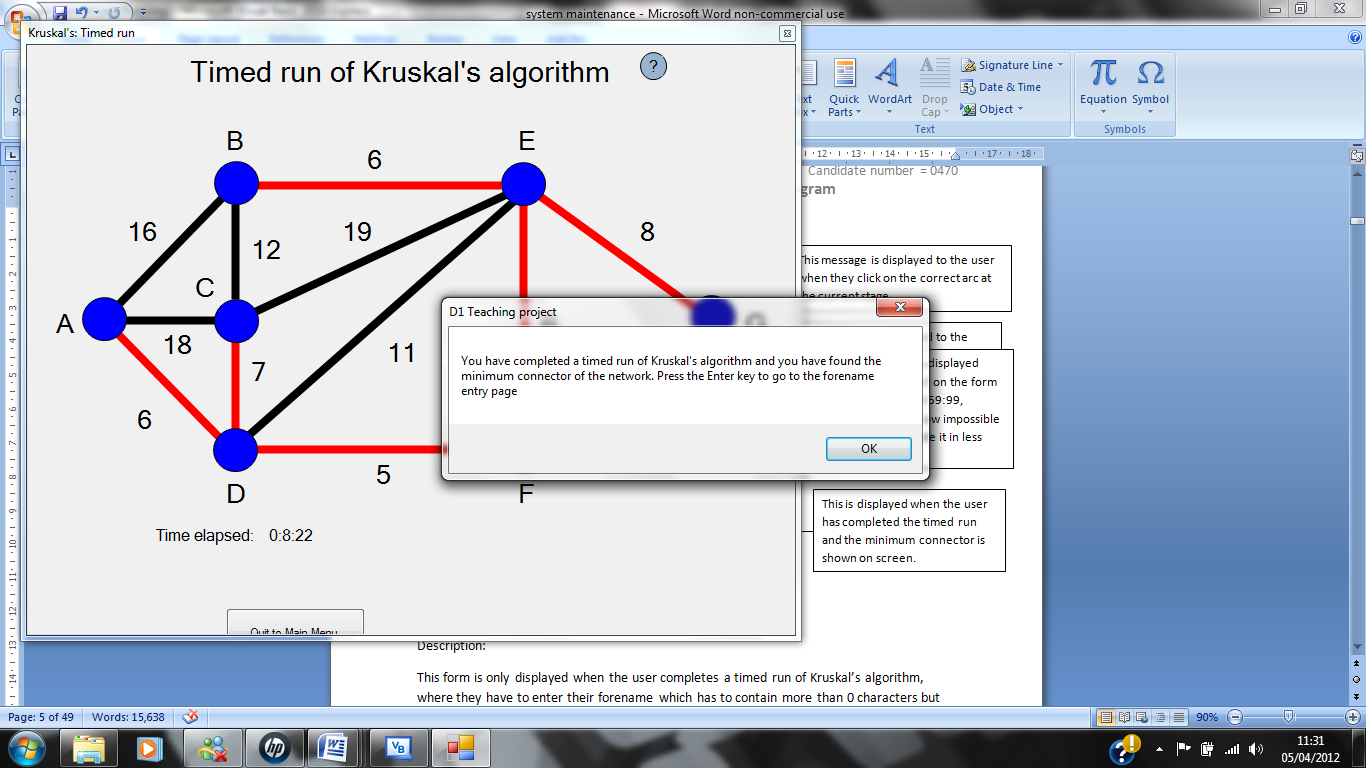
This message is displayed to the user when they click on the correct arc at the current stage.



This message is displayed to the user when they click on the incorrect arc at the current stage.



This message box is displayed when the stopwatch on the form displays a time of 9:59:99, meaning that it is now impossible for them to complete it in less than 10 minutes.



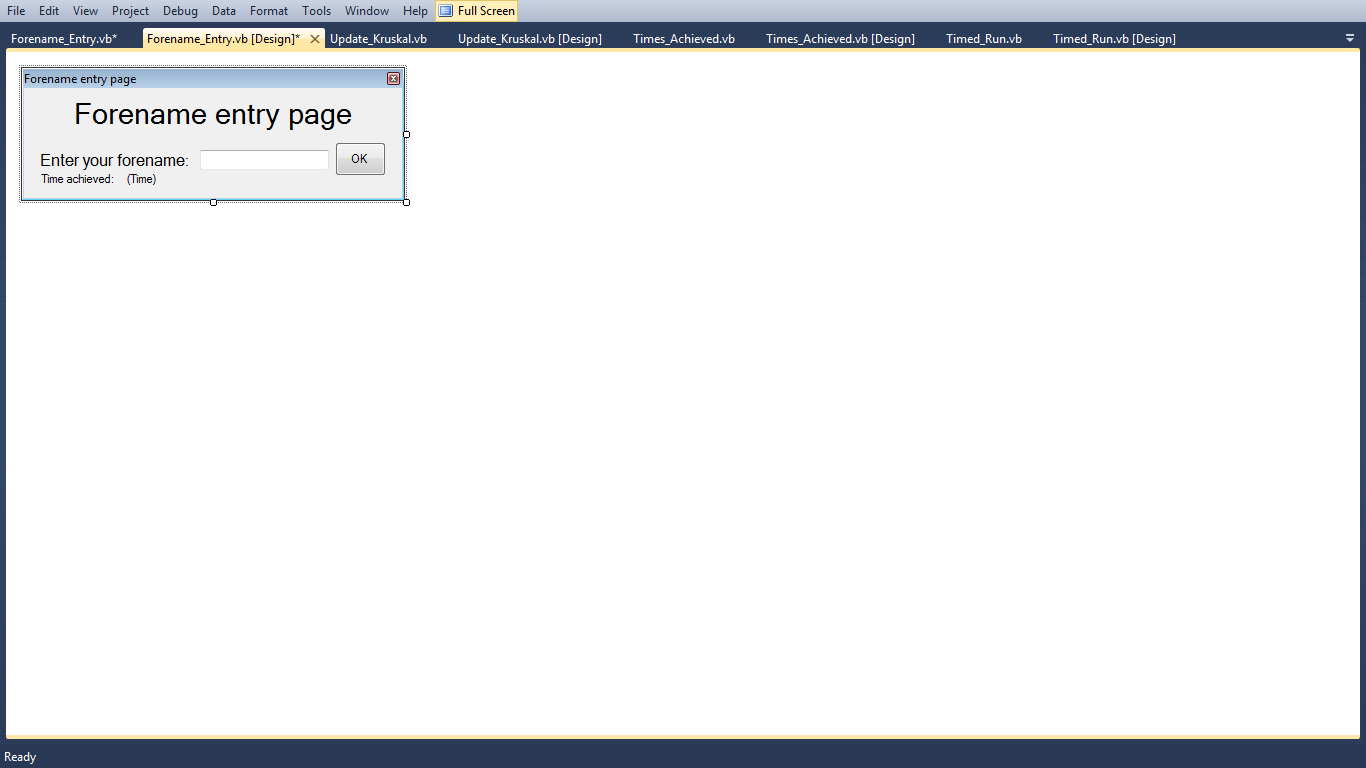
This is displayed when the user has completed the timed run and the minimum connector is shown on screen.

Forename\_Entry

Description:

This form is only displayed when the user completes a timed run of Kruskal’s algorithm, where they have to enter their forename which has to contain more than 0 characters but no more than 50 characters. This is so that other users can see how they compare with their friends and people they know. I have also made it accept 50 characters or less because I want to make it so that the user can enter their forename and/or their surname and/or their middle name. Once the user enters a name into the text box they must press the “OK” button on the form; if the name entered is accepted then a message box is displayed which tells the user their time achieved and forename have been saved to the file. After the “OK” button on the message box is pressed this form is closed and the Main\_Menu form is displayed. However when the name entered is not accepted a message box is displayed which tells the user why and the name in the textbox is deleted so that the user can enter another name; this is done after the “OK” button has been pressed on the message box. Also when the number of records in the file reaches 100 no further records can be saved by users until some have been deleted by a teacher using the Update\_Kruskal form. This is to stop users from saving too many records and so that the file has a limit to the number of records it can hold. If the number of records in the file is 100 and a user tries to save another the program will output a message box that tells the user that the maximum number of records has been reached and that some records must be deleted to continue to save records. The user is then taken back to the Main\_Menu form along with subsequent users until the problem is solved.

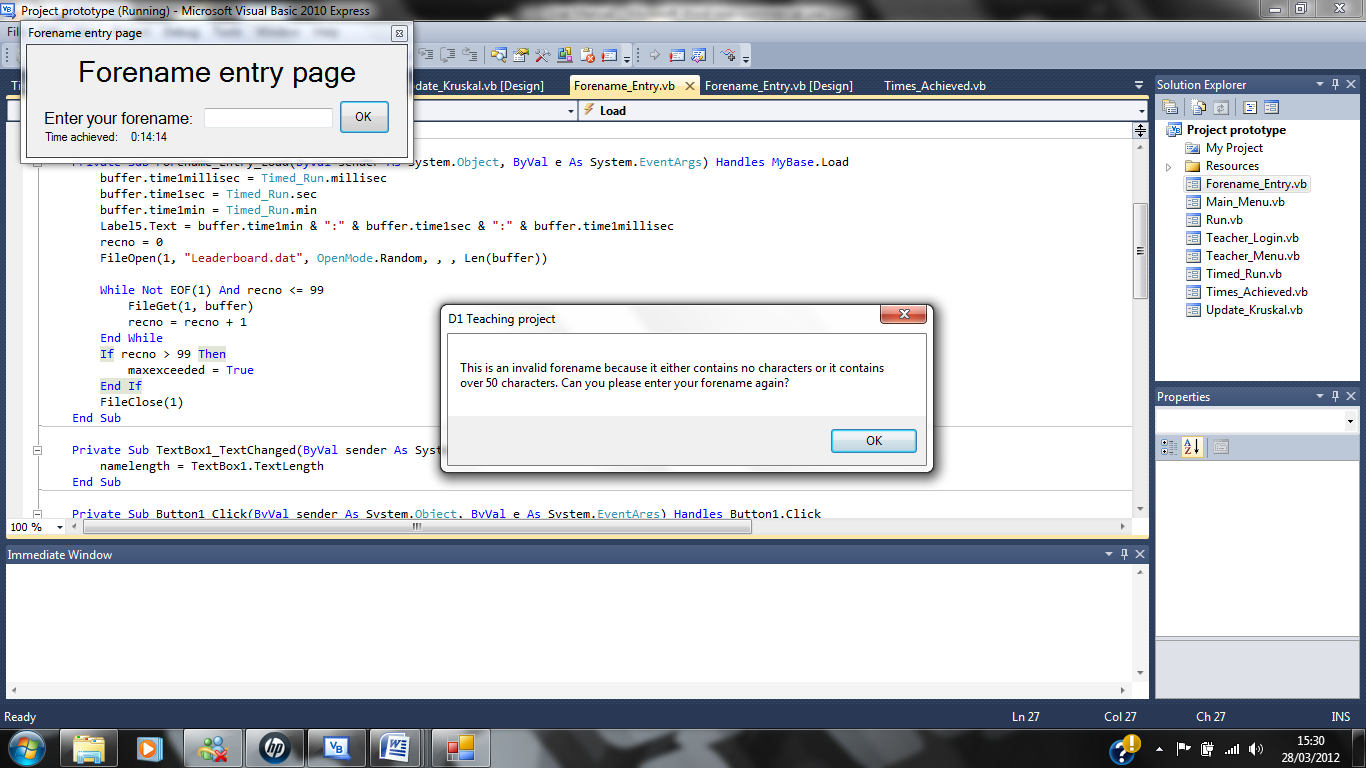
Screenshot:



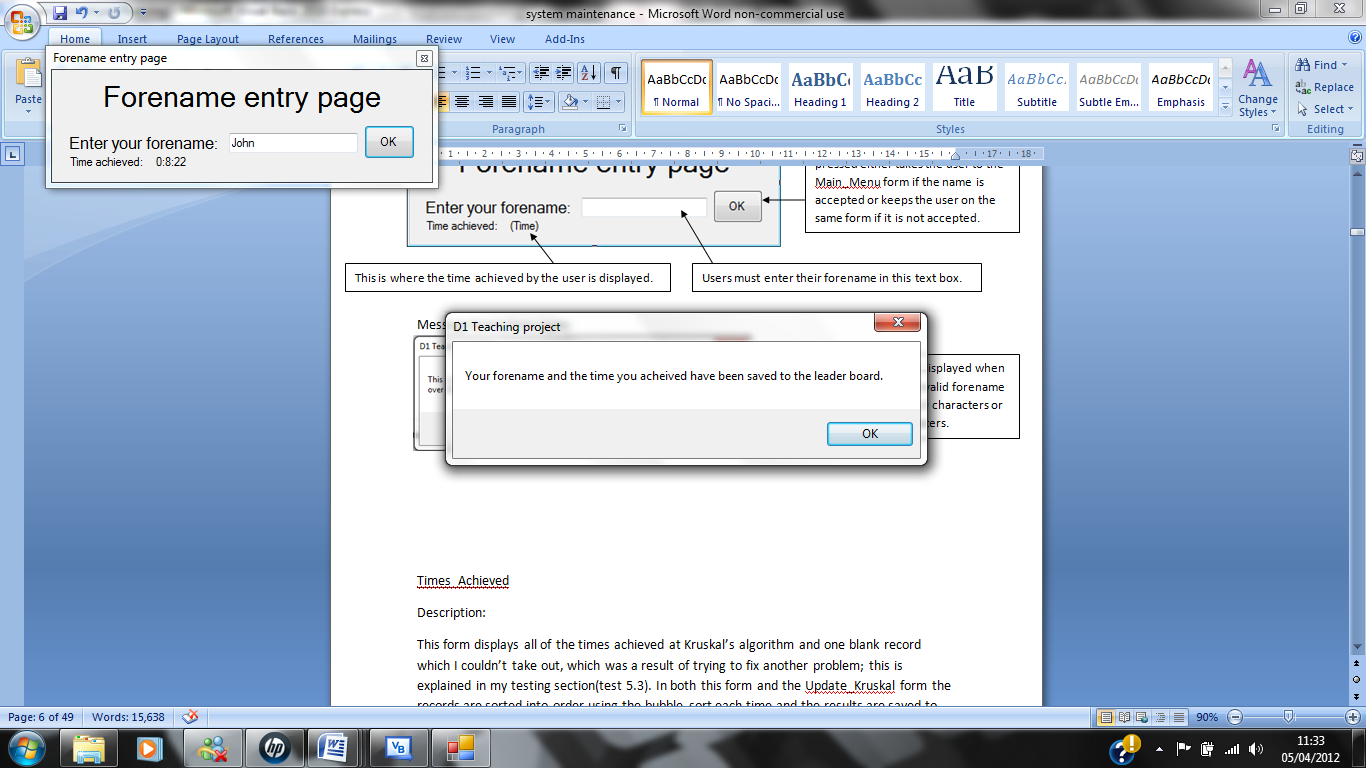
This is the “OK” button which when pressed either takes the user to the Main\_Menu form if the name is accepted or keeps the user on the same form if it is not accepted.

Users must enter their forename in this text box.

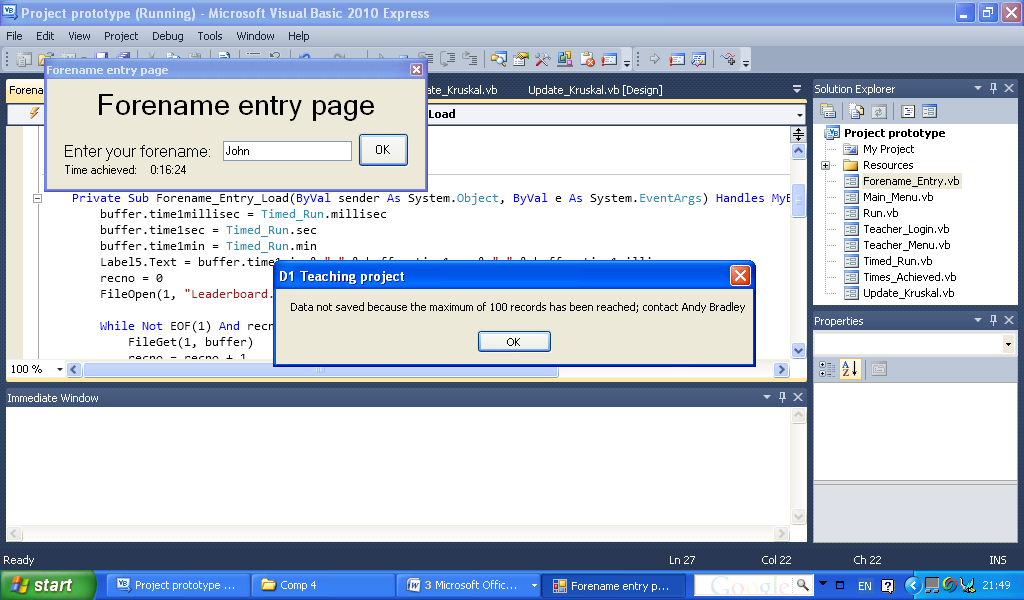
This is where the time achieved by the user is displayed.

Message boxes/messages:

This message box is displayed when the user enters an invalid forename that either contains 0 characters or more than 50 characters; after pressing the “OK” button.



This message box is displayed when the user’s forename entered has been accepted and the forename has been saved to the file; after pressing the “OK” button.



This message box is displayed when the number of records in the file is 100 and the user is trying to save another record.

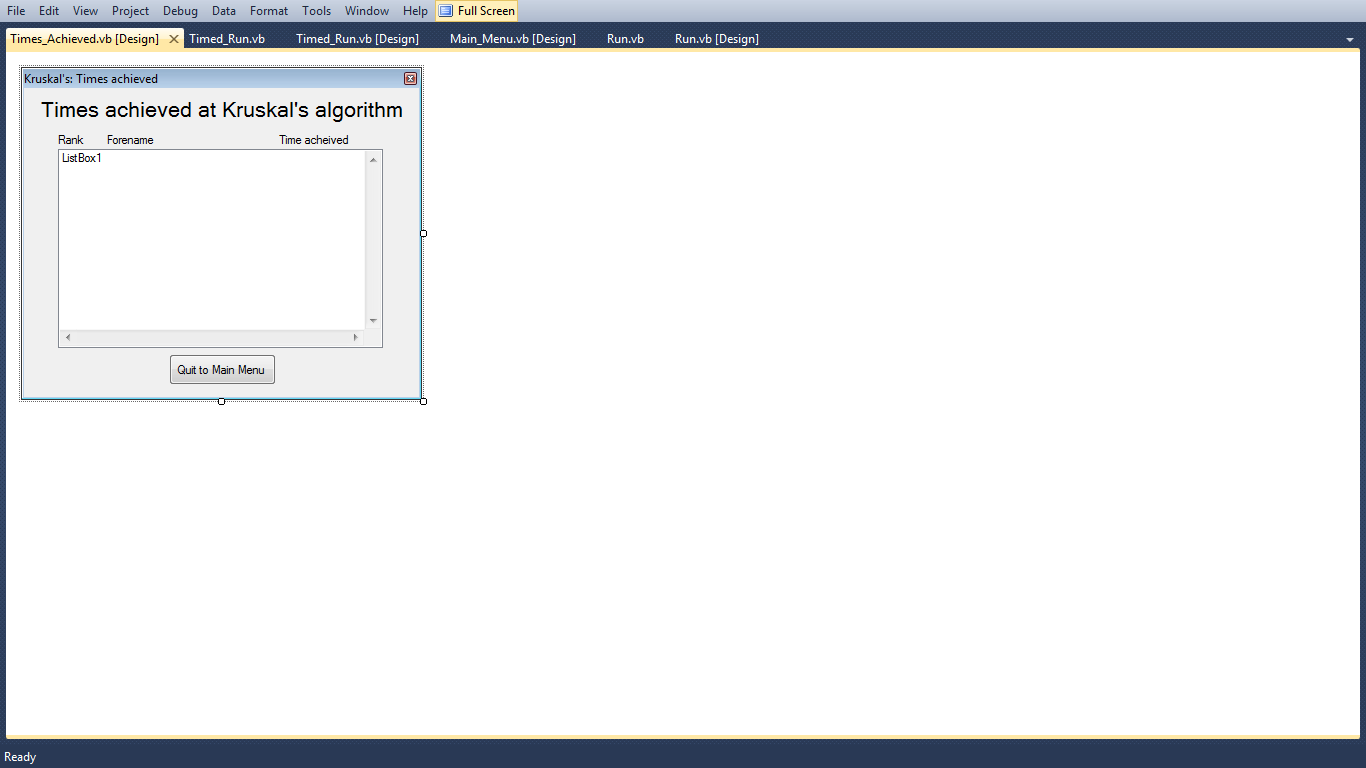
Times\_Achieved

Description:

This form displays all of the times achieved at Kruskal’s algorithm and one blank record which I couldn’t take out, which was a result of trying to fix another problem; this is explained in my testing section(test 5.3). In both this form and the Update\_Kruskal form the records are sorted into order using the bubble sort each time and the results are saved to the file without the ranks; which are generated each time they are loaded. The results of the bubble sort are also displayed in the list boxes on both of these forms.

Screenshot:

Times achieved by users are displayed in this list box and are displayed in line where one is beneath another. Except for those with long forenames whose time achieved is not displayed in line with the others. This is due to a problem with my program; this is explained in my testing section (test 2.1).



This form can still be loaded even if the number of records in the file is 100, so that users can still see the times that were achieved by other users.

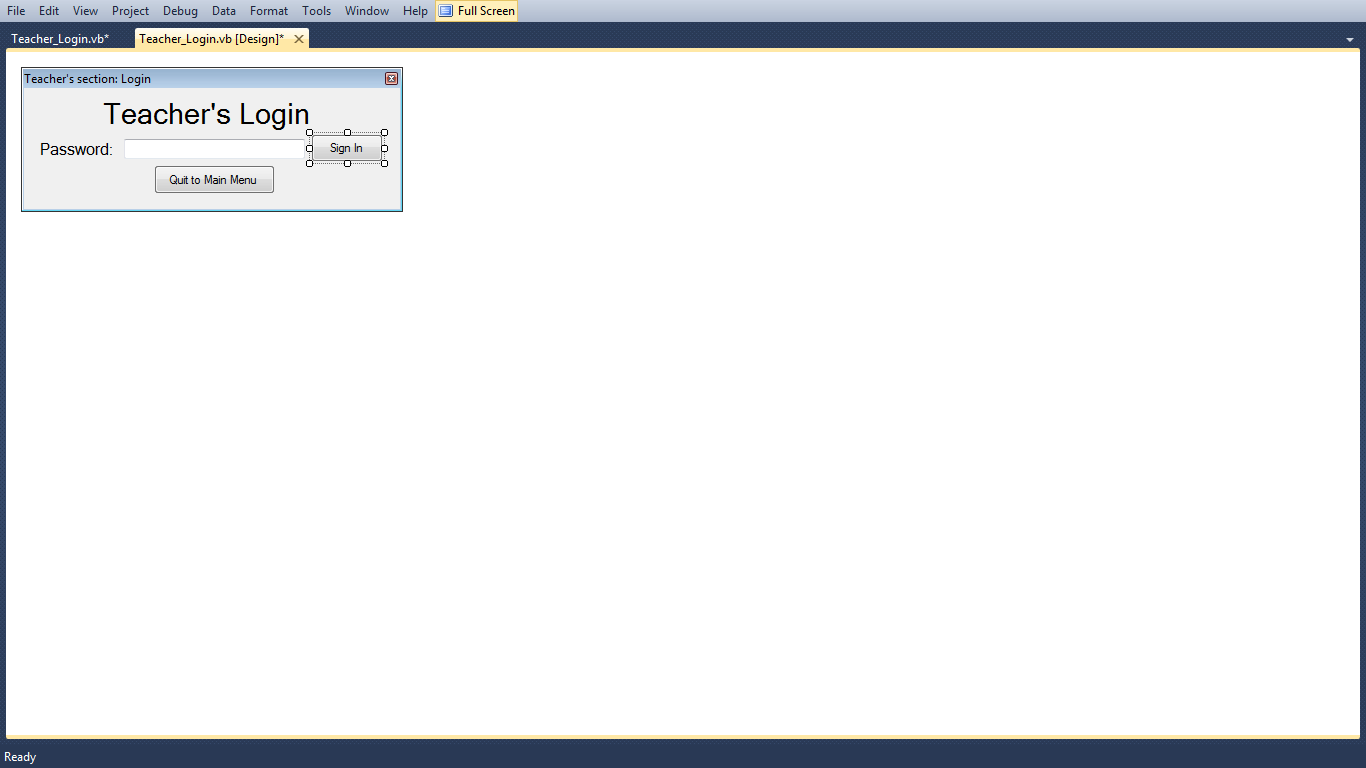
When this button is pressed this form is closed and the Main\_Menu form is displayed.

Teacher\_Login

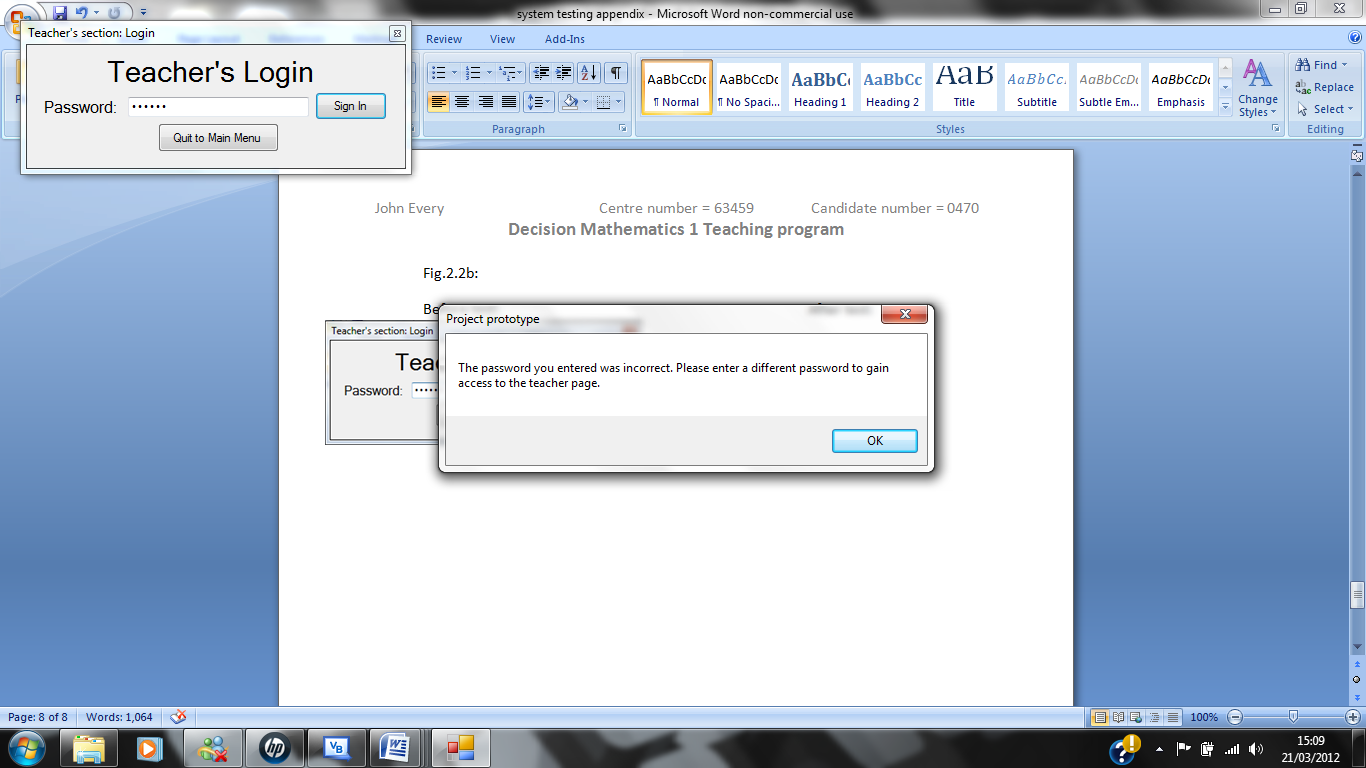
Description: This form enables a teacher to log in to the teacher’s section so that they can delete records saved by users. The password that needs to be entered into the textbox to gain access to the teacher’s section is “Euler”; any other passwords entered will be rejected when the “Sign in” button is pressed.

This is the text box that the password must be entered in to gain access to the teacher’s section; once the user has been allowed access in to the teacher’s section the Teacher\_Menu form is displayed.

Screenshot:



This button when pressed closes this form and displays the Main\_Menu form.

Message boxes/messages

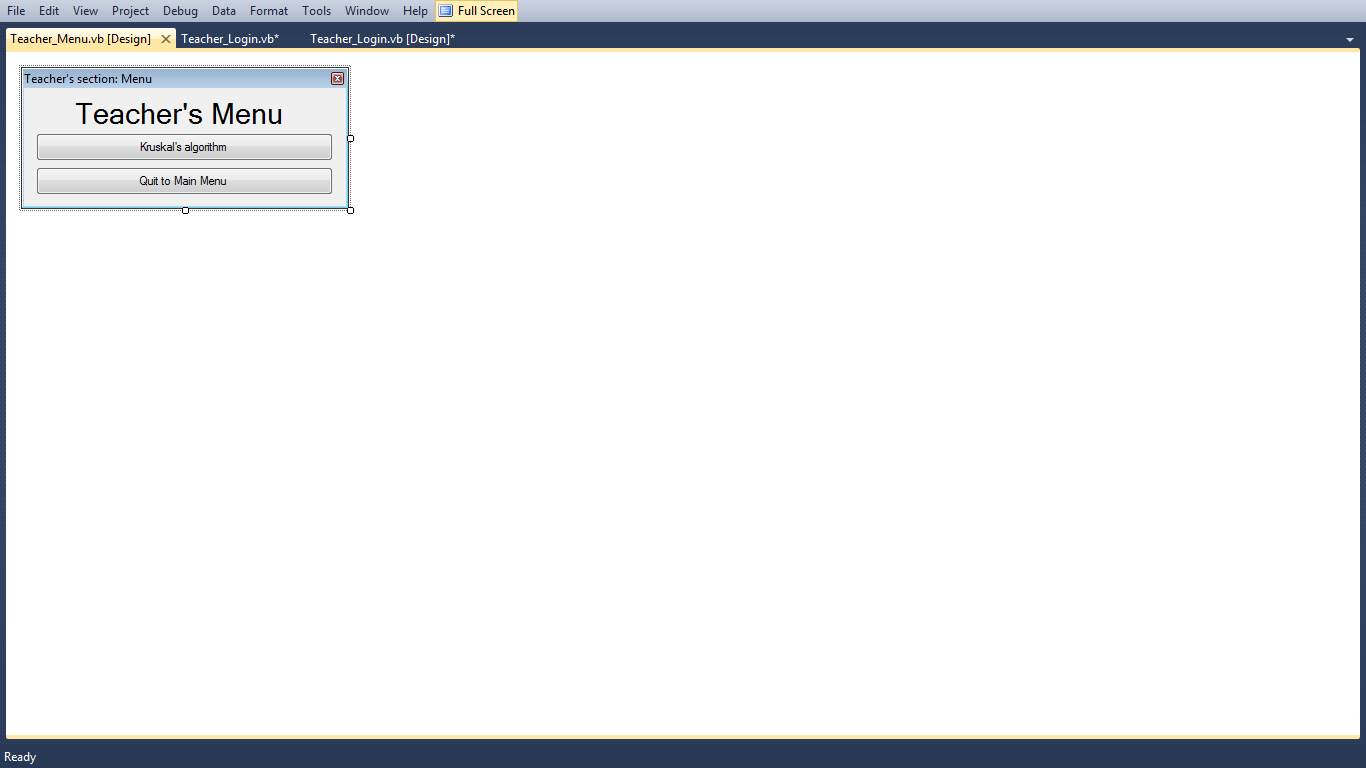
This message box is displayed when the user has entered an incorrect password and pressed the “Sign In” button.

Teacher\_Menu

Description:

This form is used to navigate through the teacher’s section so that the teacher can find what they want quickly and easily. The reason for having Kruskal’s algorithm as one of the choices instead of just update times achieved is because I wanted my program to be easily updated to incorporate other aspects of the D1 syllabus. Therefore if I or someone else wanted to do a section on Prim’s algorithm network form then they could easily do so without changing any of the work that I have done.

Screenshot:

****

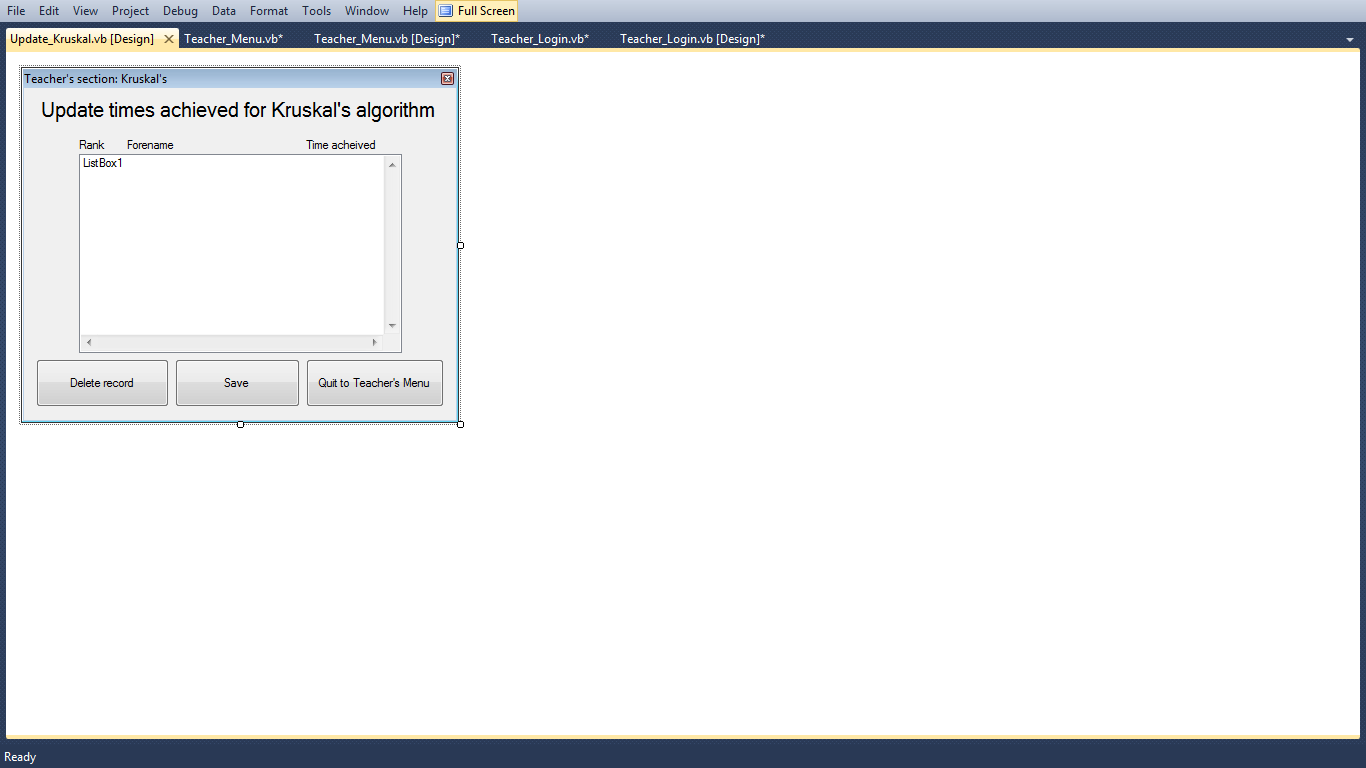
When this button is pressed the Update\_Kruskal form is displayed to the user.

This button closes this form and displays the Main\_Menu form.

Update\_Kruskal

Description: This is used by teachers to change the leader board of times achieved by deleting records; this is done by selecting individual records and then pressing the “Delete” button. To save these changes they must press the “Save” button; they can do this after any number of records have been deleted. The form itself displays the leader board in a list box; the same as on the Times\_Achieved form. This is done in the same way on this form as the Times\_Achieved form because the times achieved are sorted into order each time the form is loaded using the bubble sort; which produces a rank for each user.

Screenshot:

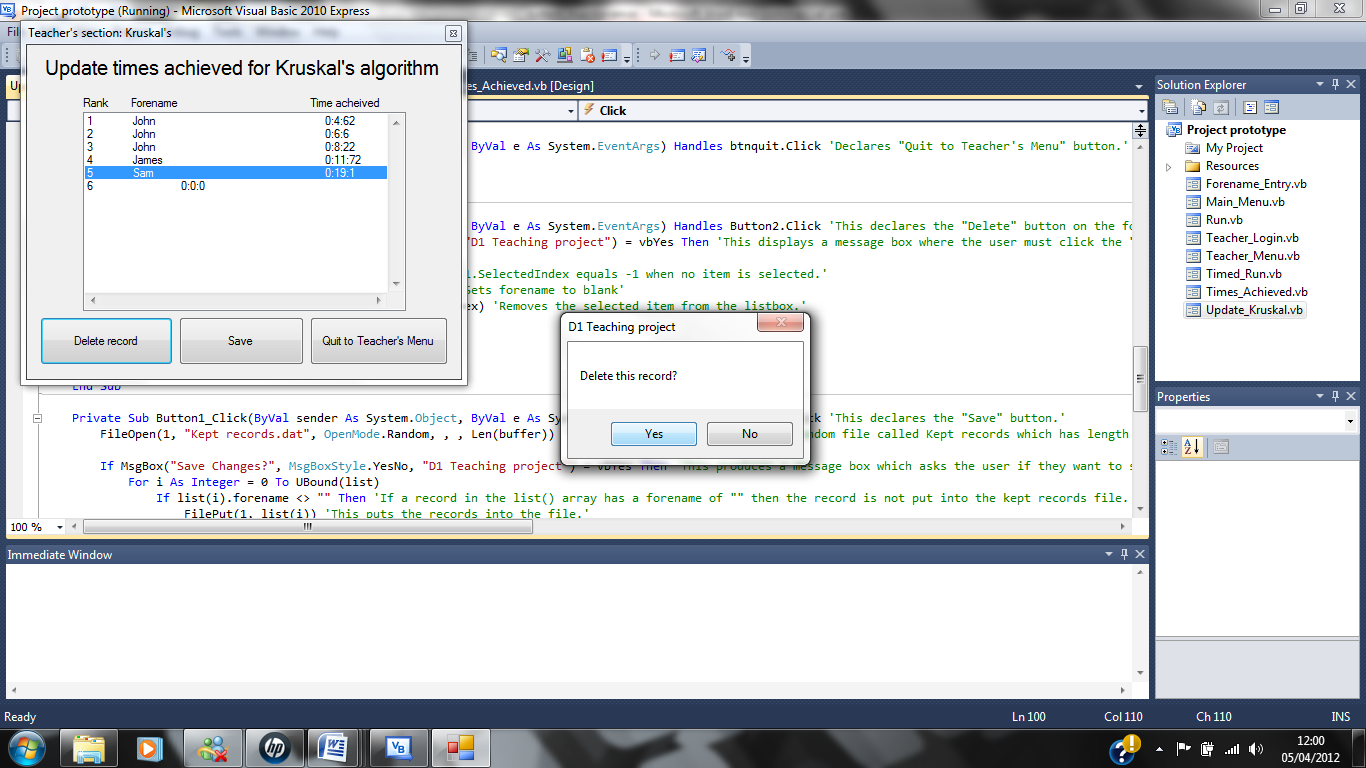


The leader board of times achieved by users is displayed in this list box when the form is loaded.

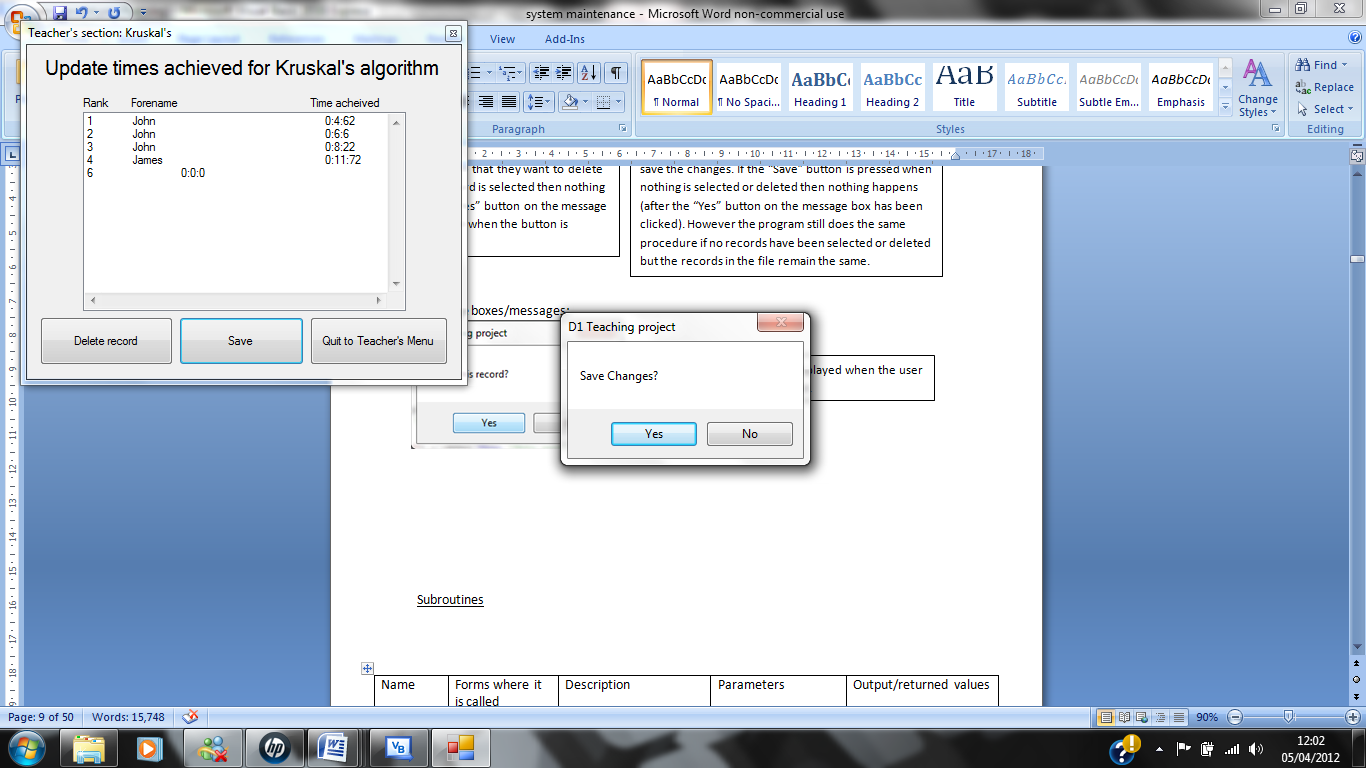
This button closes this form and displays the Teacher\_Menu form to the user.

When this button is pressed the deleted records from the list box are made permanent on the file. After the user has clicked the “Yes” button on the message box, which asks the user if they are sure that they want to save the changes. If the “Save” button is pressed when nothing is selected or deleted then nothing happens (after the “Yes” button on the message box has been clicked). However the program still does the same procedure if no records have been selected or deleted but the records in the file remain the same.

When this button is pressed and a record has been selected then the selected record is deleted from the list box after the user has clicked the “Yes” button on the message box; which asks the user whether they are sure that they want to delete the record. If no record is selected then nothing happens (after the “Yes” button on the message box has been clicked) when the button is pressed.

Message boxes/messages:

This message box is displayed when the user clicks the delete button.



This is displayed when the user presses the “Save” button on the form.

Subroutines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Forms where it is called | Description | Parameters | Output/returned values |
| Getweightings() | Timed\_Run | This subroutine works out the arc weightings for each arc using a random number generator. This then fills the weightings() array, | None. | The weightings() array filled with 12 numbers. |
| Stopwatch() | Timed\_Run | This subroutine uses the value of the global variable millisec to display a working conventional stopwatch on screen. It also stops the user from completing the timed run in over 10 minutes. | None. | Outputs a new value of the variables millisec, sec and min. If millisec is equal to 100 when the function is called it outputs a new value of the variables min, sec and millisec. The new value of sec when millisec equals 100 equals sec+1 and the new value of millisec = 0. When sec = 60 and millisec = 100 then the new value of min = min+1, sec = 0 and millisec = 0. Also when min = 9, sec = 59 and millisec = 99 a message box is displayed to the user. |
| Colouring() | Run, Timed\_Run | This subroutine colours the arcs red that are in the minimum connector and the vertices connected to these arcs are coloured blue. | Which arcs are coloured depend on the value of the variable position which was worked out in a different section of code. Arcs are coloured if the variable arcscoloured < 6 in the Timed\_Run form only. | In the Run form the variable arcschosen equals a new value. In the Timed\_Run form arcscoloured equals a new value. In both forms the colour of two vertices are changed to blue and the colour of one arc is changed to red each time the subroutine is called. If arcscoloured equals 6 in the Timed\_Run form then the Forename\_Entry form is displayed instead of the Timed\_Run form. |
| bubblesort() | Times\_Achieved and Update\_Kruskal | This subroutine sorts the list of records in the leaderboard.dat file into order from shortest to longest time. This is so that they can be output to the screen and displayed as a leader board. | times(), which is an array that is structured using the structure Forename\_Entry.timerecords. This array contains the records that are in the list(); which is used in another section of code. | It outputs an ordered list to screen in a list box. |
| swap() | Times\_Achieved and Update\_Kruskal | This subroutine swaps two numbers whose positions need to be swapped in the list()/times() array when it is called in the bubblesort subroutine. | i = the first of the two records and j = the second of the two records; both I and j are structured using the structure Forename\_Entry.timerecords and f = whether a swap has taken place. f corresponds to the flag variable in the bubblesort() subroutine. | Outputs a new i and j value and outputs f = true. These output values are then used in the bubblesort() subroutine to help sort the records into order using the bubble sort algorithm. |

**A sample of detailed algorithm design**

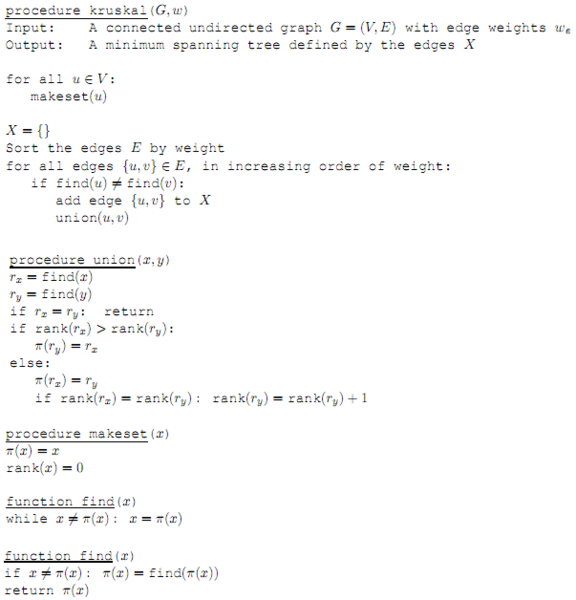
Kruskal’s algorithm

Description: This algorithm is used to find the minimum connector of a weighted network; the minimum connector produced by this algorithm contains a number of arcs equal to the number of vertices in the network minus 1. This algorithm is used a lot in industry and engineering, for example it is needed to find the minimum amount of phone line cable to connect a number of cities. Kruskal’s algorithm is known as a greedy algorithm, meaning that it makes the best choice at each stage with the hope of finding an optimal solution. Kruskal’s finds the minimum connector of the network by selecting the shortest unselected edge provided a cycle is not formed. This is done at each stage until all the vertices are connected where a minimum spanning tree/minimum connector is formed. Another algorithm used for the same purposes as Kruskal’s algorithm is Prim’s algorithm, however Kruskal’s algorithm is found to be much easier to perform than Prim’s by a number of students.

The step by step method of Kruskal’s algorithm is shown below:

1. Choose the shortest edge/arc (if there is more than one, choose any of the shortest).
2. Choose the next shortest edge/arc in the network (it doesn’t have to be joined to the edges already connected) and add it to the minimum connector.
3. Choose the next shortest edge/arc which doesn’t create a cycle and add it to the minimum connector.
4. Repeat step 3 until all of the vertices are connected then stop.

Kruskal’s algorithm has cubic complexity (O (n3)) which means that it does take quite a while for a computer to produce the solution to a minimum connector problem.

[](http://mathforum.org/mathimages/imgUpload/Kruskal-pseuocode.PNG)Pseudo code:

Code used in program in both the Run and Timed\_Run forms (The code below is taken from the Run form and therefore will be slightly different in the Timed\_Run form. Also the code for the cycles would be different for a different network; however the basis of the code would be the same.):

Do Until arcschosen = 6

high = highest(weightings)

lowest = high 'This is so that the lowest value in the array is set to the highest value in the array of weightings so that the lowest value in the array can be found each time the algorithm is run.'

For i = LBound(weightings) To UBound(weightings) 'The program checks the code for the lowest value in the array by checking all of the values from the begining of the array to the end.'

If lowest >= weightings(i) Then

If done(i) = False Then 'This is so that lowest = weightings(i) only when the value in the weightings array hasn't already been chosen as being in the minimum connector or doesn't create a cycle by picking it.'

lowest = weightings(i)

position = i + 1 'The position in the array is equal to i + 1 because i starts at 0 and position starts at 1.'

End If

End If

Next

If TrueorFalse() Then 'This is so that if a cycle has been formed by this arc being in the minimum connector then the corressponding arc won't be coloured'

Colouring() 'Colours arcs and vertices'

Me.Show() 'The code below is used so that the user must press the "OK" button on the message box for the algorithm to move onto the next stage.'

MsgBox("Go to next stage", MsgBoxStyle.OkOnly, "D1 Teaching project")

End If

Loop

Label21.Text = ""

Label22.Text = ""

Label23.Text = "You have completed a run of Kruskal's algorithm and you have found the minimum connector of the network." 'Tells the user that they are done'

Label24.Text = "However for the exam you will have to draw the minimum connector separately from the network given." 'Help for user'

End Sub

Function highest(ByVal arr() As Integer) As Integer 'This function is used to find the highest number in the weightings array'

Dim high As Integer = arr(0) 'This sets the variable high to the value in the weightings array at i = 0, so that if a value in the array is larger than this then the value of high will be the highest value in the array.'

For i = 1 To UBound(arr)

If arr(i) > high Then high = arr(i)

Next

Return high 'This returns high to the place where highest was called in the code above so that the program can find the lowest value in the array during each iteration of the algorithm.

End Function

Dim cycles(30) As Boolean 'This is to stop the algorithm from recognising the same cycle twice so that when the cycle has been recognised and the appropriate action has been taken by the algorithm the cycle at a certain point in the array will be set to true so that the program will not recognise it again.'

Function TrueorFalse() As Boolean 'This function is used to change the arc chosen value to True in the done() array so that the program knows not to use the same arc again in the rest of the iterations.'

Dim i As Integer = 0

done(position - 1) = True 'Sets the value in the done() array to True; where i=position-1 . This causes the program to miss out the same arc weight during the next iteration so that this function is not called to again with the same arc.'

If arcschosen >= 3 Then 'This is so that the algorithm only looks for cycles when the number of arcs in the minimum connector is more than or equal to 3.'

If position = 1 Or position = 5 Or position = 4 Then 'From here down are the possible cycles that could be formed in the network.'

If cycles(i) = False Then 'This is so that if the program sets cycle(i) to true meaning the cycle has been prevented before then it won't try to be prevented again.'

If done(0) = True And done(4) = True And done(3) = True Then 'If all these three values in the done array are true then this means that a cycle will occur and therefore needs to be prevented.'

cycles(i) = True

Return False 'This prevents a cycle from forming, because this stops the arc from being coloured.'

Else : i = i + 1 'This causes the program to read the next block of code for another cycle because more than one cycle may be formed by one arc being chosen and therefore need to be prevented and stopped from happening again.'

End If

Else : i = i + 1 'This causes the program to read the next block of code for another cycle because more than one cycle may be formed by one arc being chosen and therefore need to be prevented and stopped from happening again.'

End If

Else : i = i + 1 'This causes the program to read the next block of code for another cycle because more than one cycle may be formed by one arc being chosen and therefore need to be prevented and stopped from happening again.'

End If

If position = 4 Or position = 10 Or position = 9 Then

If cycles(i) = False Then

If done(3) = True And done(9) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 6 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(5) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 10 Or position = 7 Or position = 6 Then

If cycles(i) = False Then

If done(9) = True And done(6) = True And done(5) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 7 Or position = 8 Or position = 11 Then

If cycles(i) = False Then

If done(6) = True And done(7) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 10 Or position = 11 Or position = 8 Or position = 6 Then

If cycles(i) = False Then

If done(9) = True And done(10) = True And done(7) = True And done(5) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 8 Or position = 11 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(7) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 5 Or position = 10 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(4) = True And done(9) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 11 Or position = 8 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(10) = True And done(7) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 7 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(6) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 8 Or position = 12 Or position = 3 Then

If cycles(i) = False Then

If done(7) = True And done(11) = True And done(2) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 11 Or position = 12 Or position = 3 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(10) = True And done(11) = True And done(2) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 3 Or position = 12 Or position = 11 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(2) = True And done(11) = True And done(10) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 7 Or position = 3 Or position = 12 Or position = 11 Then

If cycles(i) = False Then

If done(6) = True And done(2) = True And done(11) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 7 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(6) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 6 Or position = 4 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(5) = True And done(3) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 4 Or position = 6 Or position = 7 Or position = 9 Then

If cycles(i) = False Then

If done(3) = True And done(5) = True And done(6) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

End If

Return True

End Function

Bubble sort

Description: The bubble sort sorts an array of numbers of any length n, into order from the smallest number to the largest number. It does this by comparing the first two numbers in the array if the second is less than the first then they swap, then the second and third numbers are compared and if the third is less than the second then they swap. This continues until you reach the end of the array where you perform another iteration of the sort from the start of the array. After the first iteration the last number in the array will have been sorted and after the second iteration the last two numbers will have been sorted, the method continues in the same way until no swaps take place. However when n is large the performance of the bubble sort in sorting the array is reduced. This is due to it having quadratic complexity (O (n2)), however it can be made to be more efficient but the bubble sort I have used is the conventional bubble sort. I used this sort because it is being used to sort a maximum of 100 records; therefore it will be a good enough sort for what I want it to do.

The step by step method of the bubble sort is shown below:

Step 1: Compare the first two numbers in the array.

Step 2: If the second number is smaller than the first then swap the two numbers.

Step 3: Compare the second number with the next number.

Step 4: Repeat steps 2 and 3 for all pairs of numbers until you reach the end of the list.

Step 5: Repeat steps 1 to 4 until no more swaps are made.

Pseudo code:

procedure bubbleSort( A : list of sort able items ) repeat swapped = false for i = 1 to length(A) - 1 inclusive do: if A[i-1] > A[i] then swap( A[i-1], A[i] ) swapped = true end if end for until not swapped end procedure

Code used in program in the Times\_Achieved and Update\_Kruskal forms:

Some of the basis of the following code was taken from some work that I did in class on the bubble sort.

Sub bubblesort(ByRef times() As Forename\_Entry.timerecords) 'Below is the same as in the Times\_Achieved form.'

Dim flag As Boolean = True

Dim swapforbidden As Boolean

Dim count As Integer

Do Until flag = False

flag = False

For count = 0 To UBound(times) - 1

swapforbidden = False

If times(count + 1).time1min > 0 Or times(count + 1).time1sec > 0 Or times(count + 1).time1millisec > 0 Then

If times(count).time1min > times(count + 1).time1min Then

swap(times(count), times(count + 1), flag)

swapforbidden = True

ElseIf times(count).time1min < times(count + 1).time1min Then

swapforbidden = True

End If

If swapforbidden = False Then

If times(count).time1sec > times(count + 1).time1sec And times(count).time1min <= times(count + 1).time1min Then

swap(times(count), times(count + 1), flag)

swapforbidden = True

ElseIf times(count).time1sec < times(count + 1).time1sec Then

swapforbidden = True

End If

End If

If swapforbidden = False Then

If times(count).time1millisec > times(count + 1).time1millisec Then

swap(times(count), times(count + 1), flag)

swapforbidden = True

End If

End If

End If

Next

Loop

End Sub

Sub swap(ByRef i As Forename\_Entry.timerecords, ByRef j As Forename\_Entry.timerecords, ByRef f As Boolean) 'Below is the same as in the Times\_Achieved form.'

Dim temp As Forename\_Entry.timerecords

temp = i

i = j

j = temp

f = True

End Sub

**Annotated listings of program code/macro code and tailoring**

Form = Main\_Menu

Public Class Main\_Menu

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'Declares quit button.'

End 'This is used so that when the user presses the "Quit" button the debugging of the solution ends and exits the program.'

End Sub

Private Sub ComboBox1\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged

If ComboBox1.SelectedItem = "Run" Then 'If the user presses the drop down button on the combobox and selects the "Run" option then the program will take the user to the Run form and hide this form.'

Run.Show()

Me.Hide()

End If

If ComboBox1.SelectedItem = "Timed Run" Then 'If the user presses the drop down button on the combobox and selects the "Timed Run" option then the program will take the user to the Timed\_Run form and hide this form.'

Timed\_Run.Show()

Me.Hide()

End If

If ComboBox1.SelectedItem = "Times Achieved" Then 'If the user presses the drop down button on the combobox and selects the "Times Achieved" option then the program will take the user to the Times\_Achieved form and hide this form'

Times\_achieved.Show()

Me.Hide()

End If

End Sub

Private Sub btnt\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnt.Click 'Declares "Teacher's Section" button'

Teacher\_Login.Show() 'When the user clicks the "Teacher's section" button the program will take the user to the Teacher\_Login form and hide this form'

Me.Hide()

End Sub

End Class

Form = Run

Public Class Run

Dim weightings() As Integer = {5, 9, 6, 3, 4, 8, 5, 10, 6, 2, 9, 3} 'This is the 1-dimensional array of arc weightings'

Dim done() As Boolean = {False, False, False, False, False, False, False, False, False, False, False, False} 'This is the 1-dimensional array that tells the program whether an arc can be selected to be in the minimum connector or is in the minimum connector. Initially they are all set to false which indicates that none of them are initially in the minimum connector and all of them can be selected to be in the minimum connector.'

Dim lowest, high, position, arcschosen As Integer 'lowest = the lowest number in the array that the program has come across during an iteration. high = the highest number in the array. Position = the position of the arc weight in the array. arcschosen = The number of arcs in the minimum connector which have been chosen by the program.

Dim listboxdisplayed As Boolean 'This variable indicates whether the listbox is being displayed at the present time.'

Private Sub Run\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Main\_Menu.Hide() 'This hides the Main\_Menu form'

Label2.Text = "A" 'The label initialising below is so that the vertices are given their letters and the arcs are given their arc weights.'

Label3.Text = "B"

Label4.Text = "C"

Label5.Text = "D"

Label6.Text = "E"

Label7.Text = "F"

Label8.Text = "G"

Label9.Text = weightings(0)

Label10.Text = weightings(1)

Label11.Text = weightings(2)

Label12.Text = weightings(3)

Label13.Text = weightings(4)

Label14.Text = weightings(5)

Label15.Text = weightings(6)

Label16.Text = weightings(7)

Label17.Text = weightings(8)

Label18.Text = weightings(9)

Label19.Text = weightings(10)

Label20.Text = weightings(11)

Label21.Text = "Find the minimum connector for this network using Kruskal's algorithm. Draw the minimum connector."

Label22.Text = "If you have the D1 revision guide then turn to page 23 where the example allows you to follow the use of the algorithm in this run."

Label23.Text = ""

Label24.Text = ""

listboxdisplayed = False 'This variable has been set to false because the listbox is not displayed when the form is loaded.'

ListBox1.Hide() 'This hides the listbox'

arcschosen = 0 'These two variables have both been set to 0 so that they don't affect the running of the algorithm until the variable values have been changed by the algorithm.'

position = 0

End Sub

Private Sub Label25\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label25.Click

'This code below shows listbox 1 when the listbox is not currently displayed and hides it when the listbox 1 is currently displayed. This code is only read when the user clicks on label 25.

If listboxdisplayed = False Then

ListBox1.Show()

listboxdisplayed = True

ElseIf listboxdisplayed = True Then

ListBox1.Hide()

listboxdisplayed = False

End If

End Sub

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'This declares the quit button'

Main\_Menu.ComboBox1.Text = "Kruskal's algorithm" 'This piece of code is used so that when the user presses the "Quit to Main Menu" button combo box 1 on the Main\_Menu form will have "Kruskal's algorithm" written in it every time.'

Main\_Menu.Show() 'This shows the Main\_Menu form'

Me.Close() 'This closes this form'

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Button1.Hide()

Do Until arcschosen = 6

high = highest(weightings)

lowest = high 'This is so that the lowest value in the array is set to the highest value in the array of weightings so that the lowest value in the array can be found each time the algorithm is run.'

For i = LBound(weightings) To UBound(weightings) 'The program checks the code for the lowest value in the array by checking all of the values from the begining of the array to the end.'

If lowest >= weightings(i) Then

If done(i) = False Then 'This is so that lowest = weightings(i) only when the value in the weightings array hasn't already been chosen as being in the minimum connector or doesn't create a cycle by picking it.'

lowest = weightings(i)

position = i + 1 'The position in the array is equal to i + 1 because i starts at 0 and position starts at 1.'

End If

End If

Next

If TrueorFalse() Then 'This is so that if a cycle has been formed by this arc being in the minimum connector then the corressponding arc won't be coloured'

Colouring() 'Colours arcs and vertices'

Me.Show() 'The code below is used so that the user must press the "OK" button on the message box for the algorithm to move onto the next stage.'

MsgBox("Go to next stage", MsgBoxStyle.OkOnly, "D1 Teaching project")

End If

Loop

Label21.Text = ""

Label22.Text = ""

Label23.Text = "You have completed a run of Kruskal's algorithm and you have found the minimum connector of the network." 'Tells the user that they are done'

Label24.Text = "However for the exam you will have to draw the minimum connector separately from the network given." 'Help for user'

End Sub

Function highest(ByVal arr() As Integer) As Integer 'This function is used to find the highest number in the weightings array'

Dim high As Integer = arr(0) 'This sets the variable high to the value in the weightings array at i = 0, so that if a value in the array is larger than this then the value of high will be the highest value in the array.'

For i = 1 To UBound(arr)

If arr(i) > high Then high = arr(i)

Next

Return high 'This returns high to the place where highest was called in the code above so that the program can find the lowest value in the array during each iteration of the algorithm.

End Function

Dim cycles(30) As Boolean 'This is to stop the algorithm from recognising the same cycle twice so that when the cycle has been recognised and the appropriate action has been taken by the algorithm the cycle at a certain point in the array will be set to true so that the program will not recognise it again.'

Function TrueorFalse() As Boolean 'This function is used to change the arc chosen value to True in the done() array so that the program knows not to use the same arc again in the rest of the iterations.'

Dim i As Integer = 0

done(position - 1) = True 'Sets the value in the done() array to True; where i=position-1 . This causes the program to miss out the same arc weight during the next iteration so that this function is not called to again with the same arc.'

If arcschosen >= 3 Then 'This is so that the algorithm only looks for cycles when the number of arcs in the minimum connector is more than or equal to 3.'

If position = 1 Or position = 5 Or position = 4 Then 'From here down are the possible cycles that could be formed in the network.'

If cycles(i) = False Then 'This is so that if the program sets cycle(i) to true meaning the cycle has been prevented before then it won't try to be prevented again.'

If done(0) = True And done(4) = True And done(3) = True Then 'If all these three values in the done array are true then this means that a cycle will occur and therefore needs to be prevented.'

cycles(i) = True

Return False 'This prevents a cycle from forming, because this stops the arc from being coloured.'

Else : i = i + 1 'This causes the program to read the next block of code for another cycle because more than one cycle may be formed by one arc being chosen and therefore need to be prevented and stopped from happening again.'

End If

Else : i = i + 1 'This causes the program to read the next block of code for another cycle because more than one cycle may be formed by one arc being chosen and therefore need to be prevented and stopped from happening again.'

End If

Else : i = i + 1 'This causes the program to read the next block of code for another cycle because more than one cycle may be formed by one arc being chosen and therefore need to be prevented and stopped from happening again.'

End If

If position = 4 Or position = 10 Or position = 9 Then

If cycles(i) = False Then

If done(3) = True And done(9) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 6 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(5) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 10 Or position = 7 Or position = 6 Then

If cycles(i) = False Then

If done(9) = True And done(6) = True And done(5) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 7 Or position = 8 Or position = 11 Then

If cycles(i) = False Then

If done(6) = True And done(7) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 10 Or position = 11 Or position = 8 Or position = 6 Then

If cycles(i) = False Then

If done(9) = True And done(10) = True And done(7) = True And done(5) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 8 Or position = 11 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(7) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 5 Or position = 10 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(4) = True And done(9) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 11 Or position = 8 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(10) = True And done(7) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 7 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(6) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 8 Or position = 12 Or position = 3 Then

If cycles(i) = False Then

If done(7) = True And done(11) = True And done(2) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 11 Or position = 12 Or position = 3 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(10) = True And done(11) = True And done(2) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 3 Or position = 12 Or position = 11 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(2) = True And done(11) = True And done(10) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 7 Or position = 3 Or position = 12 Or position = 11 Then

If cycles(i) = False Then

If done(6) = True And done(2) = True And done(11) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 7 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(6) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 6 Or position = 4 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(5) = True And done(3) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 4 Or position = 6 Or position = 7 Or position = 9 Then

If cycles(i) = False Then

If done(3) = True And done(5) = True And done(6) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

End If

Return True

End Function

Sub Colouring()

If position = 1 Then

LineShape9.BorderColor = Color.Red 'Lineshape9 is the arc that corresponds to the arc weight that is in position 1 of the weightings() array. If position = 1 then the lineshape is coloured red.'

OvalShape2.BackColor = Color.Blue 'The vertices that are joined to the arc corresponding to the arc weight at position = 1 in the weightings() array are changed colour to blue so that the user knows that these vertices have been connected in the minimum connector.'

OvalShape3.BackColor = Color.Blue

arcschosen = arcschosen + 1 'This increases the arcschosen value by 1 because one more arc has just been chosen to go into the minimum connector.

End If

If position = 2 Then 'Same as above but it is used when position = 2, therefore lineshape1 is used instead.'

LineShape1.BorderColor = Color.Red

OvalShape1.BackColor = Color.Blue

OvalShape3.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 3 Then 'Same as above but it is used when position = 3, therefore lineshape7 is used instead.'

LineShape7.BorderColor = Color.Red

OvalShape1.BackColor = Color.Blue

OvalShape10.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 4 Then 'Same as above but it is used when position = 4, therefore lineshape3 is used instead.'

LineShape3.BorderColor = Color.Red

OvalShape2.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 5 Then 'Same as above but it is used when position = 5, therefore lineshape6 is used instead.'

LineShape6.BorderColor = Color.Red

OvalShape3.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 6 Then 'Same as above but it is used when position = 6, therefore lineshape2 is used instead.'

LineShape2.BorderColor = Color.Red

OvalShape1.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 7 Then 'Same as above but it is used when position = 7, therefore lineshape12 is used instead.'

LineShape12.BorderColor = Color.Red

OvalShape1.BackColor = Color.Blue

OvalShape8.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 8 Then 'Same as above but it is used when position = 8, therefore lineshape8 is used instead.'

LineShape8.BorderColor = Color.Red

OvalShape1.BackColor = Color.Blue

OvalShape9.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 9 Then 'Same as above but it is used when position = 9, therefore lineshape10 is used instead.'

LineShape10.BorderColor = Color.Red

OvalShape2.BackColor = Color.Blue

OvalShape8.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 10 Then 'Same as above but it is used when position = 10, therefore lineshape13 is used instead.'

LineShape13.BorderColor = Color.Red

OvalShape8.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 11 Then 'Same as above but it is used when position = 11, therefore lineshape4 is used instead.'

LineShape4.BorderColor = Color.Red

OvalShape8.BackColor = Color.Blue

OvalShape9.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

If position = 12 Then 'Same as above but it is used when position = 12, therefore lineshape5 is used instead.'

LineShape5.BorderColor = Color.Red

OvalShape9.BackColor = Color.Blue

OvalShape10.BackColor = Color.Blue

arcschosen = arcschosen + 1

End If

End Sub

End Class

Form = Timed\_Run

Public Class Timed\_Run

Dim weightings(11) As Integer 'Same as in Run.'

Dim done() As Boolean = {False, False, False, False, False, False, False, False, False, False, False, False}

Dim lowest, high, position, position1, position2, position3, position4, position5, position6, arcschosen, arcscoloured, userchoice As Integer 'arcscoloured = the number of arcs coloured red, userchoice = the users choice of arc which is equal to the position of the corressponding arc weight in the weightings() array.

Dim listboxdisplayed, started As Boolean 'started indicates whether the user has pressed the "Start" button.'

Public min, sec, millisec As Integer 'min = number of minutes elapsed, sec = number of seconds elapsed and millisec = number of milliseconds elapsed.'

Private Sub Timed\_Run\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Randomize() 'This is the Randomize function which is used to "seed" the Rnd() function with a starting point based upon the system time when the application gets loaded; this means that the sequence of generated numbers each time the form is loaded will be different each time.'

Getweightings() 'This is the call to the subroutine Getweightings()'

Label1.Text = "A" 'These label statements are used to set the text displayed in the label to what is desired by me.

Label3.Text = "B"

Label4.Text = "C"

Label5.Text = "D"

Label6.Text = "E"

Label7.Text = "F"

Label8.Text = "G"

Label9.Text = ""

Label10.Text = ""

Label11.Text = ""

Label12.Text = ""

Label13.Text = ""

Label14.Text = ""

Label15.Text = ""

Label16.Text = ""

Label17.Text = ""

Label18.Text = ""

Label19.Text = ""

Label20.Text = ""

Label23.Text = "0:0:0"

Label24.Text = ""

Label25.Text = ""

ListBox1.Hide() 'Hides listbox 1'

listboxdisplayed = False

userchoice = 0 'These variables have all been set to zero before the algorithm is run, so that the running of it is not affected by these variables before their values are changed by the algorithm.

arcschosen = 0

position = 0

position1 = 0 'position1 = position of arc weight in weightings() array corresponding to the first arc in the minimum connector'

position2 = 0 'position2 = position of arc weight in weightings() array corresponding to the second arc in the minimum connector'

position3 = 0 'position3 = position of arc weight in weightings() array corresponding to the third arc in the minimum connector'

position4 = 0 'position4 = position of arc weight in weightings() array corresponding to the fourth arc in the minimum connector'

position5 = 0 'position5 = position of arc weight in weightings() array corresponding to the fifth arc in the minimum connector'

position6 = 0 'position6 = position of arc weight in weightings() array corresponding to the sixth arc in the minimum connector'

End Sub

Private Sub Getweightings() 'This subroutine is used to find the weightings for the 12 arcs by filling the array weightings().'

For i = 0 To 11

weightings(i) = Math.Ceiling(Rnd() \* 20) 'This stops any arc weight being more than 20 and the "Math.Ceiling" part rounds the random number(produced by the random number generator) up to the nearest higher integer value; so that 0 can't be an arc weight either.'

Next

End Sub

Private Sub Label29\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label29.Click

'Same as in Run'

If listboxdisplayed = False Then

ListBox1.Show()

listboxdisplayed = True

ElseIf listboxdisplayed = True Then

ListBox1.Hide()

listboxdisplayed = False

End If

End Sub

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'Same as in Run'

Main\_Menu.ComboBox1.Text = "Kruskal's algorithm"

Main\_Menu.Show()

Me.Close()

End Sub

Private Sub btnstart\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnstart.Click

started = True 'This variable is set to true to indicate that the user has started the timed run. This is to make sure that nothing happens when the user clicks on a lineshape before pressing the "Start" button.'

btnstart.Hide()

'The following lines are used to set the text of the labels to an arc weight on the network so that a user can find the minimum connector of the network. Also these are declared when the user presses the "Start" button so that the user is prevented from cheating.'

Label9.Text = weightings(0)

Label10.Text = weightings(1)

Label11.Text = weightings(2)

Label12.Text = weightings(3)

Label13.Text = weightings(4)

Label14.Text = weightings(5)

Label15.Text = weightings(6)

Label16.Text = weightings(7)

Label17.Text = weightings(8)

Label18.Text = weightings(9)

Label19.Text = weightings(10)

Label20.Text = weightings(11)

Do Until arcschosen = 6 'The number of arcs in the minimum connector is equal to the number of vertices in the network minus one. Therefore because there are 7 vertices in this network then there are always going to be 6 arcs in the minimum connector.'

'Same as in Run; below:'

high = highest(weightings)

lowest = high

For i = LBound(weightings) To UBound(weightings)

If lowest >= weightings(i) Then

If done(i) = False Then

lowest = weightings(i)

position = i + 1

End If

End If

Next

If TrueorFalse() Then

arcschosen = arcschosen + 1 'This is put here because the colouring of the arcs is done once the user clicks on a lineshape and the algorithm is performed on the weightings() array before this can take place.'

If arcschosen = 1 Then

position1 = position 'This is where the position value of the first arc in the minimum connector is set.'

End If

If arcschosen = 2 Then

position2 = position 'This is where the position value of the second arc in the minimum connector is set.'

End If

If arcschosen = 3 Then

position3 = position 'This is where the position value of the third arc in the minimum connector is set.'

End If

If arcschosen = 4 Then

position4 = position 'This is where the position value of the fourth arc in the minimum connector is set.'

End If

If arcschosen = 5 Then

position5 = position 'This is where the position value of the fifth arc in the minimum connector is set.'

End If

If arcschosen = 6 Then

position6 = position 'This is where the position value of the sixth arc in the minimum connector is set.'

End If

End If

Loop

Timer1.Enabled = True 'This enables the timer so that the stopwatch will start to count up.'

End Sub

Function highest(ByVal arr() As Integer) As Integer 'Below is the same as in Run'

Dim high As Integer = arr(0)

For i = 1 To UBound(arr)

If arr(i) > high Then high = arr(i)

Next

Return high

End Function

Dim cycles(30) As Boolean

Function TrueorFalse() As Boolean 'Below is the same as in Run'

Dim i As Integer = 0

done(position - 1) = True

If arcschosen >= 3 Then

If position = 1 Or position = 5 Or position = 4 Then

If cycles(i) = False Then

If done(0) = True And done(4) = True And done(3) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 4 Or position = 10 Or position = 9 Then

If cycles(i) = False Then

If done(3) = True And done(9) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 6 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(5) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 10 Or position = 7 Or position = 6 Then

If cycles(i) = False Then

If done(9) = True And done(6) = True And done(5) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 7 Or position = 8 Or position = 11 Then

If cycles(i) = False Then

If done(6) = True And done(7) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 10 Or position = 11 Or position = 8 Or position = 6 Then

If cycles(i) = False Then

If done(9) = True And done(10) = True And done(7) = True And done(5) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 8 Or position = 11 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(7) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 5 Or position = 10 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(4) = True And done(9) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 11 Or position = 8 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(10) = True And done(7) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 7 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(6) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 8 Or position = 12 Or position = 3 Then

If cycles(i) = False Then

If done(7) = True And done(11) = True And done(2) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 5 Or position = 10 Or position = 11 Or position = 12 Or position = 3 Or position = 2 Then

If cycles(i) = False Then

If done(4) = True And done(9) = True And done(10) = True And done(11) = True And done(2) = True And done(1) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 3 Or position = 12 Or position = 11 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(2) = True And done(11) = True And done(10) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 7 Or position = 3 Or position = 12 Or position = 11 Then

If cycles(i) = False Then

If done(6) = True And done(2) = True And done(11) = True And done(10) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 7 Or position = 9 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(6) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 1 Or position = 2 Or position = 6 Or position = 4 Then

If cycles(i) = False Then

If done(0) = True And done(1) = True And done(5) = True And done(3) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

If position = 4 Or position = 6 Or position = 7 Or position = 9 Then

If cycles(i) = False Then

If done(3) = True And done(5) = True And done(6) = True And done(8) = True Then

cycles(i) = True

Return False

Else : i = i + 1

End If

Else : i = i + 1

End If

Else : i = i + 1

End If

End If

Return True

End Function

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick 'This is where Timer1 is declared and each time the timer ticks 1 millisecond this code will be read line by line.'

millisec = millisec + 1 'This is so that each time the timer ticks the milisecond value will increase by one.'

stopwatch() 'This calls the subroutine stopwatch().'

Label23.Text = min & ":" & sec & ":" & millisec 'This displays the min, sec and millisec values (that were worked out in stopwatch()) in label 23.'

End Sub

Sub stopwatch() 'This subroutine is used to make sure that the stopwatch displayed on screen behaves like a conventional stopwatch.'

If millisec = 100 Then 'This is so that the stopwatch on the form won't display 100 milliseconds and instead display 1 second and 0 milliseconds.'

sec = sec + 1

millisec = 0

End If

If sec = 60 Then 'This is so that the stopwatch on the form won't display 60 seconds and instead display 1 minute and 0 seconds.'

min = min + 1

sec = 0

End If

If min = 10 And sec = 0 And millisec = 0 Then 'The code below is used to stop the user from taking longer than 10 minutes therefore if the min value gets to 10 then the timer will stop and the message box will be shown but 10:0:0 won't be shown.'

Timer1.Enabled = False

MsgBox("You have failed to complete a timed run of Kruskal's algorithm in the time limit. You will now be taken back to the main menu", MsgBoxStyle.OkOnly, "D1 Teaching project")

Main\_Menu.ComboBox1.Text = "Kruskal's algorithm" 'This has been put in so that when the user goes back to the main menu the combobox will always display what I want it to display.'

Main\_Menu.Show() 'This shows the Main\_Menu form.'

Me.Close() 'This closes this form.'

End If

End Sub

Private Sub LineShape7\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape7.Click

If started = True Then 'This is to stop the code below from being read if the user hasn't pressed the start button.'

userchoice = 3 'position of the arc weight corresponding to this arc chosen (by the user) in the weightings() array.'

If arcscoloured = 0 Then

position = position1 'This sets position to the correct answer at each stage of the algorithm.'

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then 'This prevents the arc selected by the user from being coloured if the arc chosen is the wrong one.'

Label24.Text = "" 'The user is prevented from being confused so that only the incorrect message is shown and not both the correct and incorrect messages.'

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position 'If userchoice equals the position of the arc weight in the weightings() array corresponding to the arc chosen by the algorithm then the user has chosen the correct arc.'

Colouring() 'This is where the subroutine Colouring() is called.'

End If

End If

End Sub

Private Sub LineShape5\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape5.Click 'Same as previous.'

If started = True Then

userchoice = 12

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape8\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape8.Click 'Same as previous.'

If started = True Then

userchoice = 8

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape12\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape12.Click 'Same as previous.'

If started = True Then

userchoice = 7

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape2.Click 'Same as previous.'

If started = True Then

userchoice = 6

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape1.Click 'Same as previous.'

If started = True Then

userchoice = 2

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape4.Click 'Same as previous.'

If started = True Then

userchoice = 11

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape6\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape6.Click 'Same as previous.'

If started = True Then

userchoice = 5

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape13\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape13.Click 'Same as previous.'

If started = True Then

userchoice = 10

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape3.Click 'Same as previous.'

If started = True Then

userchoice = 4

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape9\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape9.Click 'Same as previous.'

If started = True Then

userchoice = 1

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Private Sub LineShape10\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles LineShape10.Click 'Same as previous.'

If started = True Then

userchoice = 9

If arcscoloured = 0 Then

position = position1

End If

If arcscoloured = 1 Then

position = position2

End If

If arcscoloured = 2 Then

position = position3

End If

If arcscoloured = 3 Then

position = position4

End If

If arcscoloured = 4 Then

position = position5

End If

If arcscoloured = 5 Then

position = position6

End If

If userchoice <> position Then

Label24.Text = ""

Label25.Text = "The arc you chose was incorrect; please choose another."

Else : userchoice = position

Colouring()

End If

End If

End Sub

Sub Colouring() 'This subroutine is used to colour the arcs chosen by the user that are correct at each stage of the algorithm and colour the vertices of the network.'

If arcscoloured < 6 Then 'The code below should only be read if the number of arcs coloured by the program is less than 6.'

If position = 1 Then

LineShape9.BorderColor = Color.Red 'Lineshape9 is the arc that corresponds to the arc weight that is in position 1 of the weightings() array. If position = 1 then the lineshape is coloured red.'

arcscoloured = arcscoloured + 1 'After an arc is coloured red the variable arcscoloured = arcscoloured + 1.'

OvalShape2.BackColor = Color.Blue 'Same as in Run'

OvalShape3.BackColor = Color.Blue

Label25.Text = "" 'Stops both the correct and incorrect message from being displayed at the same time.'

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 2 Then 'Same as above but it is used when position = 2, therefore lineshape1 is used instead.'

LineShape1.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape1.BackColor = Color.Blue

OvalShape3.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 3 Then 'Same as above but it is used when position = 3, therefore lineshape7 is used instead.'

LineShape7.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape1.BackColor = Color.Blue

OvalShape10.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 4 Then 'Same as above but it is used when position = 4, therefore lineshape3 is used instead.'

LineShape3.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape2.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 5 Then 'Same as above but it is used when position = 5, therefore lineshape6 is used instead.'

LineShape6.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape3.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 6 Then 'Same as above but it is used when position = 6, therefore lineshape2 is used instead.'

LineShape2.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape1.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 7 Then 'Same as above but it is used when position = 7, therefore lineshape12 is used instead.'

LineShape12.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape1.BackColor = Color.Blue

OvalShape8.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 8 Then 'Same as above but it is used when position = 8, therefore lineshape8 is used instead.'

LineShape8.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape1.BackColor = Color.Blue

OvalShape9.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 9 Then 'Same as above but it is used when position = 9, therefore lineshape10 is used instead.'

LineShape10.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape2.BackColor = Color.Blue

OvalShape8.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 10 Then 'Same as above but it is used when position = 10, therefore lineshape13 is used instead.'

LineShape13.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape8.BackColor = Color.Blue

OvalShape11.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 11 Then 'Same as above but it is used when position = 11, therefore lineshape4 is used instead.'

LineShape4.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape8.BackColor = Color.Blue

OvalShape9.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

If position = 12 Then 'Same as above but it is used when position = 12, therefore lineshape5 is used instead.'

LineShape5.BorderColor = Color.Red

arcscoloured = arcscoloured + 1

OvalShape9.BackColor = Color.Blue

OvalShape10.BackColor = Color.Blue

Label25.Text = ""

Label24.Text = "The arc you clicked was correct. Click on the next arc that you think is in the minimum connector."

End If

End If

If arcscoloured = 6 Then 'If arcscoloured = 6 then the minimum connector has been produced on screen.'

Label24.Text = ""

Timer1.Enabled = False 'If the minimum connector has been displayed then the user has completed the timed run meaning that the stopwatch is stopped. This piece of code stops the timer whcih in turn stops the stopwatch on screen.'

MsgBox("You have completed a timed run of Kruskal's algorithm and you have found the minimum connector of the network. Press the Enter key to go to the forename entry page", MsgBoxStyle.OkOnly, "D1 Teaching project")

Forename\_Entry.Show() 'If they press the "OK" button on the message box then the forename entry page is displayed to the user so that they can enter their forename.'

Me.Hide()

End If

End Sub

End Class

Form = Forename\_Entry

Public Class Forename\_Entry

Structure timerecords 'This is the structure timerecords which is used to structure how the records are saved to the file in the Times\_Achieved and the Update\_Kruskal forms.'

<VBFixedString(50)> Dim forename As String 'The variable forename has a fixed string of 50 so that a person's forename can only contain a max of 50 characters.'

Dim time1min As Integer 'Minute value of time achieved'

Dim time1sec As Integer 'Second value of time achieved'

Dim time1millisec As Integer 'Millisec value of time achieved'

End Structure

Dim namelength As Integer 'This variable is the length of the forename input by the user.'

Dim buffer As timerecords 'This means that buffer is structured like time records meaning that each record has a forename, a time1min, a time1sec and a time1millisec.'

Dim recno As Integer

Dim maxexceeded As Boolean 'If it is set to true then the max number of records saved in the file has been reached.'

Private Sub Forename\_Entry\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

buffer.time1millisec = Timed\_Run.millisec 'This must be done because the user's timed run results need to be saved into the file.'

buffer.time1sec = Timed\_Run.sec

buffer.time1min = Timed\_Run.min

Label5.Text = buffer.time1min & ":" & buffer.time1sec & ":" & buffer.time1millisec 'This displays the stopwatch on screen so that the user can see the time they achieved.'

recno = 0

FileOpen(1, "Leaderboard.dat", OpenMode.Random, , , Len(buffer)) 'Leaderboard file opened.'

While Not EOF(1) And recno <= 99 'buffer is only filled when this is true'

FileGet(1, buffer) 'This fills the buffer'

recno = recno + 1 'This adds 1 to recno so that the program adds each record into buffer.'

End While

If recno > 99 Then 'If recno is larger than 99 then the number of records in the buffer has been exceeded by the new record.'

maxexceeded = True

End If

FileClose(1)

End Sub

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

namelength = TextBox1.TextLength 'This makes the variable namelength equal to the length of the text entered by the user in the textbox.'

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

If namelength > 50 Or TextBox1.Text = "" Then 'This makes it so that the user must enter a forename that is 50 characters or less but contains more than 0 characters.'

MsgBox("This is an invalid forename because it either contains no characters or it contains over 50 characters. Can you please enter your forename again?", MsgBoxStyle.OkOnly, "D1 Teaching project")

Else : buffer.forename = TextBox1.Text

buffer.time1millisec = Timed\_Run.millisec 'This has been entered again so that the time achieved is saved with the forename.'

buffer.time1sec = Timed\_Run.sec

buffer.time1min = Timed\_Run.min

Label5.Text = buffer.time1min & ":" & buffer.time1sec & ":" & buffer.time1millisec

FileOpen(1, "Leaderboard.dat", OpenMode.Random, , , Len(buffer))

If maxexceeded = True Then 'If the max number of records in the file has been exceeded then the user is not allowed to save his record.'

MsgBox("Data not saved because the maximum of 100 records has been reached; contact Andy Bradley", MsgBoxStyle.OkOnly, "D1 Teaching project")

FileClose(1) 'This message box is displayed and the file is closed.'

Else : FilePut(1, buffer, recno + 1) 'This inputs the record just entered by the user into the file as the last record.'

FileClose(1)

recno = recno + 1 'This sets recno so that the new record has been put in this position in the buffer.'

MsgBox("Your forename and the time you acheived have been saved to the leader board.", MsgBoxStyle.OkOnly, "D1 Teaching project")

End If

Timed\_Run.Close() 'This closes the Timed\_Run form because it is not needed any more.'

Main\_Menu.ComboBox1.Text = "Kruskal's algorithm" ' Below is the same as in the Run, Timed\_Run and Times\_Achieved forms.'

Main\_Menu.Show()

Me.Close()

End If

End Sub

End Class

Form = Times\_Achieved

Public Class Times\_achieved

Const Max = 99 'Constant of 99 so that list array size is 100 records.'

Public list(Max) As Forename\_Entry.timerecords

Dim buffer As Forename\_Entry.timerecords 'This is the variable buffer which was defined in the Forename\_Entry form.'

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'Same as in Run and Timed\_Run.'

Main\_Menu.ComboBox1.Text = "Kruskal's algorithm"

Main\_Menu.Show()

Me.Close()

End Sub

Private Sub Times\_Achieved\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Dim i As Integer = 0

FileOpen(1, "Leaderboard.dat", OpenMode.Random, , , Len(buffer)) 'Opens binary random file with name "Leaderboard.dat" which has a length equal to buffer.'

While Not EOF(1) 'Keeps reading the code in the while loop until the end of the file has been reached.'

FileGet(1, buffer) 'This gets the contents of buffer in the file.'

list(i) = buffer 'Here list(i) = contents of buffer at first record, second record etc.'

i = i + 1 'This adds 1 to i so that when the loop comes around again the list array will be filled with the second, third and fourth record etc.'

End While

FileClose(1) 'Closes file.'

bubblesort(list) 'Performs the bubble sort on the list() array.'

i = 0

If list(i).forename > "" Then 'This is to stop the program from inputting blank records into the textbox.'

ListBox1.Items.Add(i + 1 & vbTab & list(i).forename & vbTab & list(i).time1min & ":" & list(i).time1sec & ":" & list(i).time1millisec) 'This inputs the records into the listbox with the persons rank, forename and time achieved. The VB tabs are used to keep everything in line.'

End If

'Both pieces of code above and below are also used to make the program save all of the 100 records into the file but still be able to use both the Times\_Achieved and Update\_Kruskal's forms'.

While list(i).forename > "" And i < Max 'This stops an error message being displayed by vb of it reading outside the bounds of the array.'

i = i + 1 'This makes it so that the next record is put into the listbox during each iteration.'

ListBox1.Items.Add(i + 1 & vbTab & list(i).forename & vbTab & list(i).time1min & ":" & list(i).time1sec & ":" & list(i).time1millisec)

End While

End Sub

Sub bubblesort(ByRef times() As Forename\_Entry.timerecords) 'This is the subroutine for the bubble sort which is performed on the list array which is equal to the times array.'

Dim flag As Boolean = True 'Indicates that a swap has taken place and the bubble sort must keep on being applied.'

Dim swapforbidden As Boolean 'Indicates whether a swap is allowed to take place.'

Dim count As Integer 'This is the number that denotes the position of the record in the times/list arrays.'

Do Until flag = False 'Therefore the loop is performed until no swaps are performed.'

flag = False 'No swaps taken place at start.'

For count = 0 To UBound(times) - 1

swapforbidden = False 'Swaps are allowed once the iterations have started.'

If times(count + 1).time1min > 0 Or times(count + 1).time1sec > 0 Or times(count + 1).time1millisec > 0 Then 'A time that is 0:0:0 is not possible and therefore can't be sorted.Therefore where there are no records in positions of the array they are not sorted and therefore don't affect what is shown in the listbox.'

If times(count).time1min > times(count + 1).time1min Then 'Swap only takes place if the current record's min value is more than the next record's min value.'

swap(times(count), times(count + 1), flag) 'Swaps the current record with the next record.'

swapforbidden = True 'If this happens then another swap between the two records can't take place during this iteration.'

ElseIf times(count).time1min < times(count + 1).time1min Then 'If this is true then another swap during this iteration is also not possible.'

swapforbidden = True

End If

If swapforbidden = False Then 'If a swap is allowed.'

If times(count).time1sec > times(count + 1).time1sec Then 'If both the current and next record's min values are equal then the sec value of the current record must be more than the next record for a swap to take place.'

swap(times(count), times(count + 1), flag)

swapforbidden = True

ElseIf times(count).time1sec < times(count + 1).time1sec Then

swapforbidden = True

End If

End If

If swapforbidden = False Then

If times(count).time1millisec > times(count + 1).time1millisec Then 'If the both the current and next record's min and sec values are equal then the millisec value of the current record must be more than the next record for a swap to take place.'

swap(times(count), times(count + 1), flag)

swapforbidden = True

End If

End If

End If

Next

Loop

End Sub

Sub swap(ByRef i As Forename\_Entry.timerecords, ByRef j As Forename\_Entry.timerecords, ByRef f As Boolean) 'This subroutine swaps the two records that need to be swapped from the bubblesort subroutine.'

Dim temp As Forename\_Entry.timerecords

temp = i 'temp = current record'

i = j 'i = next record'

j = temp 'j = current record'

f = True 'Therefore current record now equals next record and next record now equals current record, therefore swap has taken place.'

End Sub

End Class

Form = Teacher\_Login

Public Class Teacher\_Login

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'Declares the "Quit to Main Menu button.'

Main\_Menu.Show() 'Displays the Main Menu.'

Me.Close() 'Closes this form.'

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click 'Declares the "Sign in" button'

If TextBox1.Text = "Euler" Then 'If the Text written into the text box by the user equals "Euler" then they have entered the correct password; the teacher's menu is shown and this form is closed.'

Teacher\_Menu.Show()

Me.Close()

Else

MsgBox("The password you entered was incorrect. Please enter a different password to gain access to the teacher page.", MsgBoxStyle.OkOnly, "D1 Teaching project") 'If the user enters the incorrect password then this displays a message box that tells them this and they then must enter a new forename.'

TextBox1.Text = "" 'So that they don't have to delete what is currently in the text box to enter a new name.'

End If

End Sub

End Class

Form = Teacher\_Menu

Public Class Teacher\_Menu

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'Declares the "Quit to Main Menu" button.'

Main\_Menu.Show() 'The Main Menu is then shown and this form is closed.'

Me.Close()

End Sub

Private Sub btnk\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnk.Click 'Declares the "Kruskal's algorithm" button.'

Update\_Kruskal.Show() 'If this button is pressed then the Update\_Kruskal form is displayed to the user.'

Me.Hide() 'This form is then hidden.'

End Sub

End Class

Form = Update\_Kruskal

Public Class Update\_Kruskal

Const Max = 99 'These three lines of code are the same as in the Times\_Achieved form.'

Public list(Max) As Forename\_Entry.timerecords

Dim buffer As Forename\_Entry.timerecords

Private Sub Update\_Kruskal\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load 'Below is the same as in the Times\_Achieved form.'

Dim i As Integer = 0

FileOpen(1, "Leaderboard.dat", OpenMode.Random, , , Len(buffer))

While Not EOF(1)

FileGet(1, buffer)

list(i) = buffer

i = i + 1

End While

FileClose(1)

bubblesort(list)

i = 0

If list(i).forename > "" Then

ListBox1.Items.Add(i + 1 & vbTab & list(i).forename & vbTab & list(i).time1min & ":" & list(i).time1sec & ":" & list(i).time1millisec)

End If

While list(i).forename > "" And i < Max

i = i + 1

ListBox1.Items.Add(i + 1 & vbTab & list(i).forename & vbTab & list(i).time1min & ":" & list(i).time1sec & ":" & list(i).time1millisec)

End While

End Sub

Sub bubblesort(ByRef times() As Forename\_Entry.timerecords) 'Below is the same as in the Times\_Achieved form.'

Dim flag As Boolean = True

Dim swapforbidden As Boolean

Dim count As Integer

Do Until flag = False

flag = False

For count = 0 To UBound(times) - 1

swapforbidden = False

If times(count + 1).time1min > 0 Or times(count + 1).time1sec > 0 Or times(count + 1).time1millisec > 0 Then

If times(count).time1min > times(count + 1).time1min Then

swap(times(count), times(count + 1), flag)

swapforbidden = True

ElseIf times(count).time1min < times(count + 1).time1min Then

swapforbidden = True

End If

If swapforbidden = False Then

If times(count).time1sec > times(count + 1).time1sec And times(count).time1min <= times(count + 1).time1min Then

swap(times(count), times(count + 1), flag)

swapforbidden = True

ElseIf times(count).time1sec < times(count + 1).time1sec Then

swapforbidden = True

End If

End If

If swapforbidden = False Then

If times(count).time1millisec > times(count + 1).time1millisec Then

swap(times(count), times(count + 1), flag)

swapforbidden = True

End If

End If

End If

Next

Loop

End Sub

Sub swap(ByRef i As Forename\_Entry.timerecords, ByRef j As Forename\_Entry.timerecords, ByRef f As Boolean) 'Below is the same as in the Times\_Achieved form.'

Dim temp As Forename\_Entry.timerecords

temp = i

i = j

j = temp

f = True

End Sub

Private Sub btnquit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnquit.Click 'Declares "Quit to Teacher's Menu" button.'

Teacher\_Menu.Show() 'Shows the Teacher\_ Menu form'

Me.Close() 'Closes this form.'

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click 'This declares the "Delete" button on the form.'

If MsgBox("Delete this record?", MsgBoxStyle.YesNo, "D1 Teaching project") = vbYes Then 'This displays a message box where the user must click the "Yes" button to delete the selected record.'

If ListBox1.SelectedIndex > -1 Then 'The listbox1.SelectedIndex equals -1 when no item is selected.'

list(ListBox1.SelectedIndex).forename = "" 'Sets forename to blank'

ListBox1.Items.RemoveAt(ListBox1.SelectedIndex) 'Removes the selected item from the listbox.'

End If

End If

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click 'This declares the "Save" button.'

FileOpen(1, "Kept records.dat", OpenMode.Random, , , Len(buffer)) 'This creates and opens a binary random file called Kept records which has length equal to buffer.'

If MsgBox("Save Changes?", MsgBoxStyle.YesNo, "D1 Teaching project") = vbYes Then 'This produces a message box which asks the user if they want to save the changes to the listbox to the file. If they press the "Yes" button then the program saves the changes to the file.'

For i As Integer = 0 To UBound(list)

If list(i).forename <> "" Then 'If a record in the list() array has a forename of "" then the record is not put into the kept records file. The records that have been given a blank forname were given these during the delete button subroutine; these were the records that were selected and deleted by the user.'

FilePut(1, list(i)) 'This puts the records into the file.'

End If

Next

FileClose(1)

My.Computer.FileSystem.DeleteFile("Leaderboard.dat") 'This deletes the current leaderboard file.'

FileSystem.Rename("Kept records.dat", "Leaderboard.dat") 'This changes the name of the "Kept records.dat" file to "Leaderboard.dat", therefore it now contains the new leaderboard of times achieved.'

End If

End Sub

End Class

**Procedure and variable lists with descriptions for program code**

Procedures/functions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Forms where it is called | Description | Parameters | Output/returned values |
| highest(), Type = Integer | Run and Timed\_Run | This function is used to find the largest number in the weightings() array. This is then used to find the lowest number in the array at every stage in another section of code. | arr(), which contains the arc weightings that are contained in the weightings() array. | The value of the variable high as integer which is equal to the largest number in the weightings() array. |
| TrueorFalse(), Type = Boolean | Run and Timed\_Run | This function outputs whether the arc chosen by the program can go into the minimum connector as Boolean. It does this by setting the value of the arc in the done() array (which corresponds to the same position number as its arc weight in the weightings() array) to true, so that in another section of code the same arc is not chosen again. This function then finds out whether a cycle will be formed with this arc being chosen, if it will then the function returns false and the code in the If statement won’t be read. If it doesn’t then it will return true and the code in the If statement will be read; where the function is called. However most of the code in this function won’t be read unless the variable arcschosen is greater than or equal to 3. | None. | Returns true or false depending on whether the arc chosen is in the minimum connector or not. |

Variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Forms where it is declared | Type | Description | Reference in design dictionary |
| weightings() (Global) | Run and Timed\_Run | Integer | This is the array of arc weightings used to find the minimum connector of the network. | This is short term data that is deleted after the form is closed in the Timed\_Run form but set and is therefore long term data in the Run form. However this is a one-dimensional array rather than a two-dimensional array. The reason for this is explained in my appraisal in the changes section. The arc weights that are worked out in the Timed\_Run form are worked out correctly and all are in the boundaries set in my design data dictionary in the design section; which was proved in the testing section (test 3.1). |
| done() (Global) | Run and Timed\_Run | Boolean | This is the array that contains true or false statements at each position. These indicate whether an arc which is at the same position in the done() array as its corresponding arc weight in the weightings() array cannot be chosen to be in the minimum connector. | None, this is because I decided to use this during construction. |
| cycles() (Global) | Run and Timed\_Run | Boolean | This is the array of cycles which can contain 31 values, meaning there are 31 possible cycles that can occur. However in the coding for the TrueorFalse() function there is only 17 cycles. This is so that if someone else who is looking at my program spots a cycle that I have missed then they can add the code for it more easily. | None, this is because I decided on how I was going to code Kruskal’s algorithm after the design stage. However it is long term data because the cycles have been coded for in advance therefore all the possible cycles are being stored in the code all the time, as is the size of the array. |
| lowest (Global) | Run and Timed\_Run | Integer | This contains the lowest number in the weightings() array. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because after the first stage the lowest number in the weightings() array is kept, then during the second stage the value of lowest changes. However the lowest values at each stage are the same in the Run form but different in the Timed\_Run form. |
| high (Global) | Run and Timed\_Run | Integer | This contains the highest number in the weightings() array. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because after the application of Kruskal’s algorithm on either form has been completed the value of high is not known by the program. However the value of high is constant during the application of the algorithm; on both forms. |
| position (Global) | Run and Timed\_Run | Integer | This variable is equal to the position of an arc weight in the weightings() array + 1. This is because the position in the array starts at 0 but the position variable’s lowest number equals 1 and its highest is 12. This is easier to understand so that instead of saying an arc weight is at position = 0 in the array (how VB has it) it is actually at position = 1 in the array. This makes more sense and allows me to use this value separately instead of using i which is used in other places. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because during the application of the algorithm the position value changes during every stage, however it is used more in the Timed\_Run form than the Run form. |
| arcschosen (Global) | Run and Timed\_Run | Integer | This holds the number of arcs in the minimum connector chosen by the program. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because during the application of the algorithm the value of arcschosen increases by 1 when an arc is chosen by the program to go into the minimum connector. When the forms are loaded again this value is set back to 0 meaning its past value has been deleted. |
| listboxdisplayed (Global) | Run and Timed\_Run | Boolean | This variable indicates whether listbox1 on both forms is currently being displayed. It is called in label 25’s event handler in the Run form but it is called in label 29’s event handler in the Timed\_Run form. | None, this is because I decided how I was going to code the help button after the design stage. However this is short term data because the variable will only have a value during debugging but after debugging the value will be forgotten by the program. |
| i (Local) | Run and Timed\_Run | Integer | This variable holds the position number of a value in the weightings() array, the done() array and the cycles() array. This variable is also used in the Times\_Achieved and Update\_Kruskal forms to indicate the position number of a record in the list() array which is then used to output the records to screen. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because the values of i are changed during the application of the algorithm, however the values in the three arrays will still have the same i values. For both the weightings() and done() arrays the values of i are between 0 and 11 but in the cycles() array i can be between 0 and 30. In the list() array i can take a value from 0 to 99. |
| postion1 (Global) | Timed\_Run | Integer | This variable holds the value of the variable position of the first arc/arc weight in the minimum connector. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because this variable is given a value once the “Start” button has been pressed on the form; which it keeps until debugging has stopped. However this value will more likely change the next time a timed run is started. |
| position2 (Global) | Timed\_Run | Integer | This variable holds the value of the variable position of the second arc/arc weight in the minimum connector. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because this variable is given a value once the “Start” button has been pressed on the form; which it keeps until debugging has stopped. However this value will more likely change the next time a timed run is started. |
| position3 (Global) | Timed\_Run | Integer | This variable holds the value of the variable position of the third arc/arc weight in the minimum connector. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because this variable is given a value once the “Start” button has been pressed on the form; which it keeps until debugging has stopped. However this value will more likely change the next time a timed run is started. |
| position4 (Global) | Timed\_Run | Integer | This variable holds the value of the variable position of the fourth arc/arc weight in the minimum connector. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because this variable is given a value once the “Start” button has been pressed on the form; which it keeps until debugging has stopped. However this value will more likely change the next time a timed run is started. |
| position5 (Global) | Timed\_Run | Integer | This variable holds the value of the variable position of the fifth arc/arc weight in the minimum connector. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because this variable is given a value once the “Start” button has been pressed on the form; which it keeps until debugging has stopped. However this value will more likely change the next time a timed run is started. |
| position6 (Global) | Timed\_Run | Integer | This variable holds the value of the variable position of the sixth arc/arc weight in the minimum connector. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because this variable is given a value once the “Start” button has been pressed on the form; which it keeps until debugging has stopped. However this value will more likely change the next time a timed run is started. |
| arcscoloured (Global) | Timed\_Run | Integer | Holds the number of arcs coloured by the program; this also means the number of arcs chosen by the user that they have chosen correctly. | None, this is because I knew about how I was going to construct Kruskal’s algorithm after the design section. However this is short term data because during the application of the algorithm the value of arcscoloured increases by 1 when an arc is coloured by the program. When the form is loaded again this value is set back to 0 meaning its past value has been deleted. |
| userchoice (Global) | Timed\_Run | Integer | This variable holds the users choice of arc which is equal to the position of the corresponding arc weight in the weightings() array. | None, however it does store the user’s choices every time (short term data) they are made which could be useful for another purpose. |
| started (Global) | Timed\_Run | Boolean | Indicates whether the user has pressed the “Start” button on the form. | None, however it is used to make sure that nothing happens when the user clicks on a lineshape before the “Start” button is pressed. |
| min (Global) | Timed\_Run, however it is a public variable therefore it is used in other forms. | Integer | This variable holds the number of minutes elapsed since the “Start” button was pressed. | This is used to get the number of minutes in the time achieved by the user at the timed run; which is mentioned in the design data dictionary as being long term data. However the max value that this variable can have is 9 instead of 59 in my design data dictionary. |
| sec (Global) | Timed\_Run, however it is a public variable therefore it is used in other forms. | Integer | This variable holds the number of minutes elapsed since the “Start” button was pressed. | This is used to get the number of seconds in the time achieved by the user at the timed run; which is mentioned in the design data dictionary as being long term data. |
| millisec (Global) | Timed\_Run, however it is a public variable therefore it is used in other forms. | Integer | This variable holds the number of minutes elapsed since the “Start” button was pressed. | This is used to get the number of milliseconds in the time achieved by the user at the timed run; which is mentioned in the design data dictionary as being long term data. |
| forename (Global) | Forename\_Entry, however this is used in other forms as it is declared in the structure timerecords. | String | This variable holds the forename that the user entered. The forename held can have a maximum of 50 characters and it must contain more than 0 characters. | This is long term data because it is stored along with the time achieved into the “Leaderboard.dat” file. This is explained in my design data dictionary. |
| time1min (Global) | Forename\_Entry, however it is used in other forms as it is declared in the structure timerecords. | Integer | This variable holds the user’s time achieved in minutes. | This is long term data and the value of this variable for each user who enters a forename is saved into the “Leaderboard.dat” file along with the user’s time achieved in seconds, milliseconds and their forename. However their rank is not saved into the file because this aspect was changed to this after I did my design data dictionary. |
| time1sec (Global) | Forename\_Entry, however it is used in other forms as it is declared in the structure timerecords. | Integer | This variable holds the user’s time achieved in seconds. | This is long term data and the value of this variable for each user who enters a forename is saved into the “Leaderboard.dat” file along with the user’s time achieved in minutes, milliseconds and their forename. |
| time1millisec (Global) | Forename\_Entry, however it is used in other forms as it is declared in the structure timerecords. | Integer | This variable holds the user’s time achieved in milliseconds. | This is long term data and the value of this variable for each user who enters a forename is saved into the “Leaderboard.dat” file along with the user’s time achieved in minutes, seconds and their forename. |
| namelength (Global) | Forename\_Entry | Integer | This variable holds the number of characters in the name entered by the user. This can have a maximum value of 50 and a minimum value of 1. | This is short term data because the value of this variable is changed each time the “OK” button on the form is pressed. This variable is used to stop users entering a forename that contains over 50 characters. This is explained in my design data dictionary in the validation required section. |
| buffer (Global) | Forename\_Entry, Update\_Kruskal and Times\_Achieved | timerecords, meaning that the value of buffer contains a forename (String), time1min (Integer), time1sec (Integer), time1millisec (Integer). | This variable holds the record of a user’s timed run, meaning that it holds the user’s forename and time achieved at the timed run. | Each record is saved into the “Leaderboard.dat” file, therefore the record that buffer holds changes when the records are being saved into the file this being the Times\_Achieved form and the Update\_Kruskal form. In the Forename\_Entry form the user’s record equals buffer. This is then saved at the end of the file. The values and past values of buffer are saved into the file making it long term data. This is how the user’s records are saved into the file as explained in my design data dictionary. |
| recno (Global) | Forename\_Entry | Integer | This indicates the record in the file the algorithm has read to. It also indicates how many records are in the file once it is finished loading them into the file. | This variable is used to help fill the buffer with records so that the correct record number in buffer is saved into the file. This was needed in the file organisation and processing section of my design data dictionary, so that all of the records will be saved into the file. |
| maxexceeeded (Global) | Forename\_Entry | Boolean | This indicates whether the max number of records in the file has been reached. | This prevents more than 100 records being saved into to the file because I decided to have a limit on the number of records that can be saved into the file; this is different to what I said in the dictionary. However this variable is an example of short term data. |
| Max (Global) | Times\_Achieved and Update\_Kruskal’s | Integer | This variable is a constant value this being 99. It is used to make sure that the number of records that can be saved in the file remains at 100. It also defines the size of the list() array. | This is long term data because it has a fixed value which doesn’t change during or after debugging. This ensures that the max amount of files that can be saved in the file remains at 100. Some of this is explained in the dictionary. |
| list() (Global) | Times\_Achieved and Update\_Kruskal’s | timerecords, this means that all of the records in the list() array are structured using the structure that is coded for on the Forename\_Entry form. | This is an array which contains all of the records in the file and therefore contains all of the past values of buffer (apart from the records that were deleted on the Update\_Kruskal form). This then makes it easier to display the records in the list boxes on these forms. | This is long term data and is explained in the file organisation and processing section of the design data dictionary. |
| flag (Local) | Times\_Achieved and Update\_Kruskal’s | Boolean | This variable indicates whether a swap has taken place during an iteration of the bubble sort. This allows the bubble sort to be performed correctly. | This is short term data; however the bubble sort results themselves are long term data because they are saved into the file but without the ranks which are generated each time the Times\_Achieved and Update\_Kruskal’s forms are loaded. |
| swapforbidden (Local) | Times\_Achieved and Update\_Kruskal’s | Boolean | This variable indicates whether a swap is allowed during the rest of an iteration of the bubble sort. | This allows the bubble sort to function properly for its purpose this being sorting all of the records into order from shortest to longest time achieved and a rank generated; this is explained in the long term data sections of my design data dictionary. |
| count (Local) | Times\_Achieved and Update\_Kruskal’s | Integer | This variable is the value of the position number in the times() array used in the bubble sort so that each record in the file is sorted. | See long term data sections of my design data dictionary. |
| temp (Local) | Times\_Achieved and Update\_Kruskal’s | Integer | Temp = the first record out of the two chosen to swap positions. Its data structure is the structure timerecords declared in the Forename\_Entry form. | This enables the bubble sort to perform properly, therefore see long term data sections in dictionary. |