COOLCULATOR

# Lewis Reed - SMW

# Centre Number: 55235

# Candidate Number: 3488

[Lewis Reed - SMW 1](#_Toc129591955)

[Centre Number: 55235 1](#_Toc129591956)

[Candidate Number: 3488 1](#_Toc129591957)

[ANALYSIS 4](#_Toc129591958)

[BACKGROUND 4](#_Toc129591959)

[USER INTERVIEW 4](#_Toc129591960)

[INITIAL REQUIREMENTS 5](#_Toc129591961)

[RESEARCH 7](#_Toc129591962)

[PROTOTYPE 11](#_Toc129591963)

[DATA FLOW DIAGRAM FOR FTP SERVER SHARE SYSTEM 17](#_Toc129591964)

[ADDITIONAL REQUIREMENTS AFTER PROTOTYPE 19](#_Toc129591965)

[DESIGN 21](#_Toc129591966)

[MODULAR SYSTEM AND CHART 21](#_Toc129591967)

[MORE IN-DEPTH STRUCTURE CHARTS 22](#_Toc129591968)

[MAIN METHODS EXPLAINED 28](#_Toc129591969)

[DATA DICTIONARY 29](#_Toc129591970)

[STRUCTURES (RECORDS) USED 38](#_Toc129591971)

[VALIDATION 39](#_Toc129591972)

[FILE HANDLING 40](#_Toc129591973)

[PSEUDOCODE 40](#_Toc129591974)

[USER INTERFACE DESIGN 47](#_Toc129591975)

[IMPLEMENTATION 52](#_Toc129591976)

[COMPLEXITY INDEX 52](#_Toc129591977)

[PublicInfo MODULE 53](#_Toc129591978)

[SIMPLEfrm CLASS 65](#_Toc129591979)

[COOLCULATORfrm CLASS 69](#_Toc129591980)

[CLASSfrm CLASS 73](#_Toc129591981)

[SHAREfrm CLASS 74](#_Toc129591982)

[RESULTSfrm CLASS 75](#_Toc129591983)

[GRAPHINGfrm CLASS 77](#_Toc129591984)

[SESfrm CLASS 80](#_Toc129591985)

[EQUATIONMAKERfrm CLASS 81](#_Toc129591986)

[TABLEfrm CLASS 85](#_Toc129591987)

[TESTING 88](#_Toc129591988)

[SIMPLE-STYLE CALCULATOR MODE 88](#_Toc129591989)

[SCIENTIFIC CALCULATOR MODE 92](#_Toc129591990)

[CLASS MODE 97](#_Toc129591991)

[SHARING AN ANSWER MODE 99](#_Toc129591992)

[VIEW CLASS’S RESULTS MODE 101](#_Toc129591993)

[EQUATION BUILDER MODE 103](#_Toc129591994)

[START/END/STEP ENTERING MODE 109](#_Toc129591995)

[GRAPHING MODE 111](#_Toc129591996)

[TABLE BUILDING MODE 116](#_Toc129591997)

[EVALUATION 120](#_Toc129591998)

[COMPARISON OF PROJECT AGAINTS REQUIREMENTS 120](#_Toc129591999)

[USER FEEDBACK 123](#_Toc129592000)

[FINAL ADDITIONAL REQUIREMENT FOR POSSIBLE IMPROVEMENT 123](#_Toc129592001)

# ANALYSIS

## BACKGROUND

Lawrie is an A-level Core Mathematics student. He does not consider Core Mathematics his most important subject, nor the area he would like to pursue a career in. Yet, this is the subject where he is forced to spend £34.99 on a Casio FX-991EX calculator[[1]](#footnote-2). Should this calculator get lost, damaged, or stolen, Lawrie must spend another £34.99.

What Lawrie needs is an affordable calculator suitable for A-level Core Mathematics, along with his other A-level needs like Biology and Chemistry equations. There are a lot of parts to a standard Casio FX-991EX calculator that are required. This includes basic addition and multiplication operations, mathematical constants, complex functions like factorial and trigonometry, table building and graphing.

However, the standout feature for Lawrie is the ability to save his calculations in a neater, online format. It gets saved into a QR code on the calculator, then, once scanned, the Casio website is brought up, displaying the whole calculation and the answer in a user-friendly format.

## USER INTERVIEW

User – Lawrence Procter

18 y/o

A-level Core Mathematics Student

**Q - What is the current procedure to getting a calculator for A-level maths?**

A – Firstly, my school will recommend me the model needed for the course. After that, it’s as simple as finding an easy place to find it, like Amazon, then ordering the calculator. I believe the calculator can also be found in retail shops like WHSmith.

**Q – That all sounds easy, but what problems did you encounter with this procedure?**

A – Well, it is inconvenient. There is a waiting time for delivery, or there is time taken out of my day to go to the shop to find it. Also, sometimes I barely know what I’m looking for. I can never remember FXJ-981w…, or whatever the calculator is called. If I get the wrong one, I could lose a lot of money. The school cannot pay for the calculator for me, so I have to dip fairly deep into my bank account to buy the calculator, which is something I’d rather not do.

**Q – There must be a way to make this less difficult. What solutions would ease this procedure for you?**

A – I would prefer if the process was instant. Like, if I got given the calculator in the first maths lesson by a teacher. It would be cool as well if the school could buy them at a reduced price in bulk, so the students can get them for free. That would also help because I wouldn’t need to worry about getting the wrong one.

**Q – That would be a big help. What are the problems with implementing these solutions, however?**

A – If the school bought everyone calculators it would use its very limited funding, even if found at lower prices which would be a difficult task in itself.

**Q – Yep, and then you’d still need resources for other subjects. What would be the new COOLCULATOR system achieve?**

A – COOLCULATOR will be free and available at an instantly on my screen. If it has familiar functions and a friendly user-interface, then it will be easy to switch to and easy to use. I do hope it doesn’t take too much room on my screen though.

**Q – What would be your preferred type of input to the virtual calculator?**

A – I think it depends on what I’m trying to input. I’d prefer on-screen buttons for things like sin, cos and tan, but being able to type in numbers would be very useful.

**Q – How would you like your calculations outputted?**

A – There should be a screen with the answers, where the colours are contrasted and clear, and the writing is in an easy-to-read font. I love the feature found on the more expensive Casio calculators, where they create a QR code storing my answer. When I scan it with my phone, it brings up a very nicely formatted version of my calculation and answer. However, I do wish it was actually stored online, so I could share with classmates.

**Q – What calculations do you use your scientific calculator for?**

A – My calculator actually turns out to be very useful in biology and chemistry as there is a lot of different types of long equations which need to be solved. In core maths, there is lots of use of tables, solving equations like quadratics, and some graphing. I need a range of features and functions to cover every maths topic.

**Q – How would you improve your current maths lessons?**

A – I wish the lessons were more interactive and less pure maths as I am a social person. Also, I am usually unsure where I am compared to the average in the class or year. I would love if there was some way to see everyone else’s answers, then we could work together as a class to make sure everyone gets the right answer.

**Q – And, finally, what do you wish your current calculator could do?**

A – I’d love it to be able to display my graphs and tables as that would be useful for core maths. I wish it had a quicker, more readily available mode which could solve equations. Also, it would be cool if the buttons were easier to read and understand to make it more enjoyable to use.

## INITIAL REQUIREMENTS

1. Once the COOLCULATOR loads up, it must go straight to the COOLCULATOR screen where calculations can be made, rather than a menu or something similar, as to save the user hassle and time in a lesson. They will be presented with:

* Numerical buttons showing simply with the number in a white, calculator-style font and on a consistent COOLCULATOR-blue background. When they are clicked, the value of the button is involved in the calculation.
* Operator buttons with formatting consistent with numerical buttons. However, when any of these buttons are clicked, their corresponding operator will be called for the current calculation.
* A CLASS button will show once a calculation has been completed. This will be clicked when the user wants to upload or view other’s answers.
* More buttons for scientific functions with consistent format. These buttons will include:
  + Indices – The user will click this button after typing in a value, then type a number, in superscript, to be the power of that value.
  + Trigonometric functions (sin, cos and tan) – The user will type a value after clicking this button. They will end the function by inserting a close bracket after their value.
  + Constants – These will be used in the same way as the numerical buttons, where the value will be of the constant clicked.
  + Logarithmic functions – This will work similarly to the trigonometric functions, however the user will write a value when they want to find the result of its log base-10 or natural log.

1. Also, the COOLCULATOR will display an extra ‘equation’ button to access more complex modes. The next screen will consist of modes which will allow users to:

* Make tables – This involves the user giving an equation, start, end and step, then the COOLCULATOR displays a table showing the index and the corresponding value in an appropriate format.
* Graph sketcher – The same input to the table maker will be implemented here, but instead of displaying the data in the form of a table with rows and columns, the data will be shown across a graph.
* Solve algebraic equations – The user will be able to enter an expression involving an ‘x’ and equal it to 0, and then solve the equation, finding a value for x.

1. There should be a friendly user-interface. The font needs to be ascetically pleasing while being easy to read. The calculator should also have multiple input methods, making it as efficient as possible for the user to enter their maths problems:

* Using on-screen buttons – All functions will be on-screen, meaning the user can call any function they need by using touchscreen and/or their cursor to click the appropriate buttons.
* Keyboard input – Some functions will be able to be typed in using certain keys on a keyboard, if an appropriate key can be allocated to that function, such as:
* Number row – All numbers and decimal points can be inputted from the keyboard by pressing the corresponding number key and the ‘full stop’ key respectively.
* ‘A’ or ‘a’ input – Either of these keys will call the answer function which will make the next numerical value equal to the last answer.
* ‘C’ or ‘c’ input – Either of these keys will call the clear function, which will clear the whole calculation entered.
* ‘+’, ‘-‘, ‘\*’ or ‘x’ or ‘X’, ‘/’, ‘=’ or Enter key, input – Any of these keys will call their corresponding operator for the current calculation.
* ‘^’ – This key can allow the user to enter the superscript mode, allowing them to type the power to their numerical values.

1. The user must be able to upload their answers to an online format with consistent fonts and colours, and share their answers with the class. This will be achieved once the CLASS button is pressed. They should have three options:

* Upload – Be able to enter their names, along with their inputted calculation and outputted answer.
* Download – Retrieve the class’s results straight from online and display the answers with their corresponding student’s names in an easy-to-read table or list format.
* Create a new class – The user should be able to clear the results and create a new blank class where students can input their results.

1. For any extra modes or tasks which take more room on the screen, the program should open new forms over the COOLCULATOR to keep the formatting neat, avoiding overlapping text or taking too much room on the screen. This will apply to any of the extra, complex modes in addition to the CLASS screen used to upload or view other’s answers.

## RESEARCH

The most affordable option for Lawrie would be to go for a more basic calculator, such as the Casio SL-460L:

The output is very simple and user-friendly. It shows only the numbers in the current calculation, and whether there is a value stored in the memory.

The buttons are generally easy to understand with big and clear lettering. However, the exception to this are the memory buttons as the labelling is not clear (MRC, M-, M+) and may confuse the user.

Although, this calculator does not have a key feature for Lawrie’s use, where he would like to see and save his calculations in a readable, user-friendly format. There is no QR code or any other connection to an external, online device.

But, a quick Google search will reveal similar calculators already online. For example, the one found at <https://www.online-calculator.com>:



The output is even more user-friendly than a physical version. The font is more recognisable, and numbers can be displayed clearer and sharper due to the calculator using the more advanced screen of a smartphone/computer. This would allow Lawrie to screenshot, save his work and calculations as he goes along, in a neat format.

The virtual buttons are just as, if not more clear on the online option, with the more ascetically-pleasing colours and designs.

This option is completely free. All that is needed is a device with an internet connection. This is the approach that will be taken by the COOLCULATOR.

These first two calculators fill their purposes very well. They are simple, easy to use and have a very user-friendly layout. The minimalist design will be used in the COOLCULATOR. However, the obvious issue with Lawrie using these calculators is the limited functionality. He would only be able to do basic addition, subtraction, multiplication and division. As soon as he reaches something like trigonometry, he would be stuck. Lawrie needs a scientific calculator.

The only other affordable option Lawrie has is to use the scientific calculator he already owns from his use at GCSE level Mathematics. This is the Casio FX-83GT Plus:



The output is identical to the output on the FX-991EX. The calculation is shown separate from the answer, which is seen in the bottom right corner.

Some of the buttons are understandable and easy to use, like the numbers, operators and functions like sin, cos and tan. However there are more complex functions used for A-level that are hard to read and determine the use for.

Yet, along with the first physical calculator option, this calculator does not have any QR code creating capability, nor any other alternative to connecting to an external device for neater format saving.

Another Google search later, similar online scientific calculators are found, including Google’s themselves:

Shape

Description automatically generated with medium confidence

The output is very similar to the first virtual calculator option. The numbers are displayed with contrasting colours in a recognisable font.

When the answer history button is pressed, previous calculations along with their answers are displayed, allowing for easy screenshotting and saving of work.

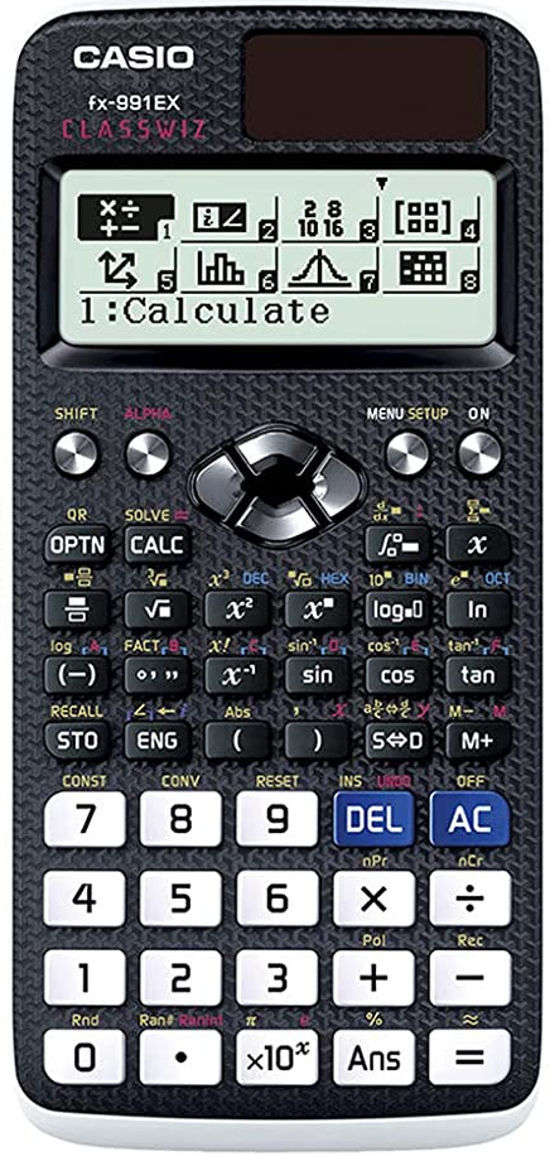
A screenshot of a computer

Description automatically generated with medium confidence

The buttons, including the ones for complex A-level functions, are clearer and easier to understand than the physical calculator.

The issue with these solutions is the lack of extra modes. Unless Lawrie uses multiple websites for each different mode, he will not have access to important features found in the FX-991EX calculator. However, this would take too much time and be too much hassle for him in a lesson with work to complete. Also, the GCSE level calculator has significantly less modes than the A-level equivalent, as seen:

FX-83GT Plus

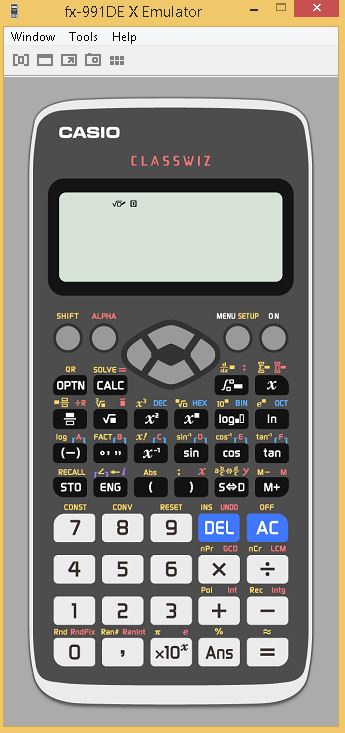


FX-991EX

A picture containing text

Description automatically generated &

After searching long for a solution, Lawrie thought he hit the jackpot. He discovered that Casio themselves provide a virtual emulator for their FX-991EX calculator, as seen here[[2]](#footnote-3):



The emulator was perfect as it was local to his devices and had all the features he needed.

A person holding a calculator

Description automatically generated

The design was clearly an exact replica of the real calculator.

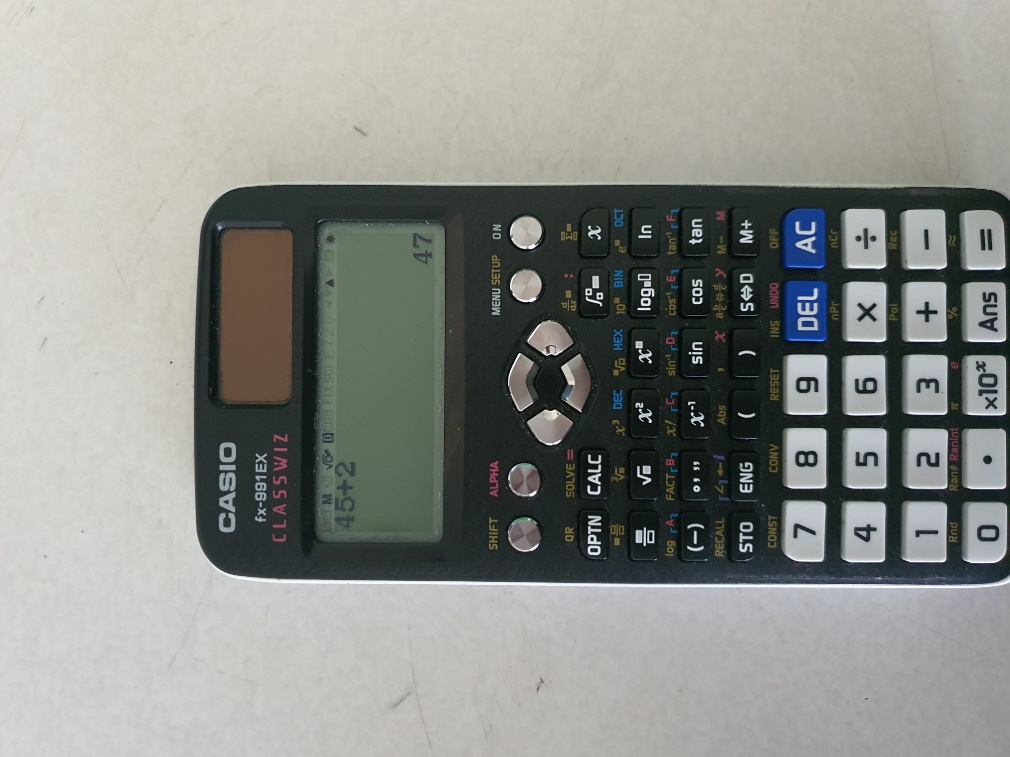
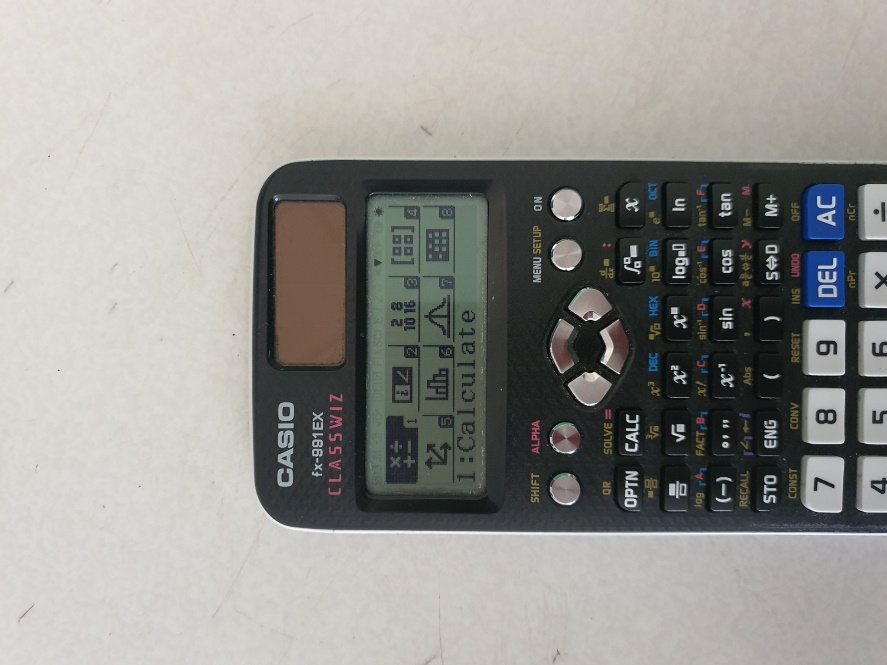
The buttons and controls were familiar, and the output was identical to the one on the physical display.

The layout was perfect for Lawrie as he had seen and used it before on older Casio calculators. There was nothing new to confuse him and all use cases of the physical machine were possible on the emulator.

He was very much satisfied, until, three months later, where he was hit with a message saying his “free trial” was over and he now needed to pay an extortionate subscription fee to keep the emulator.

Lawrie is one example of many students who need a permanent solution which will not damage their bank accounts. The solution will need to be easily accessible and have all the functions he needs for his A-level course.

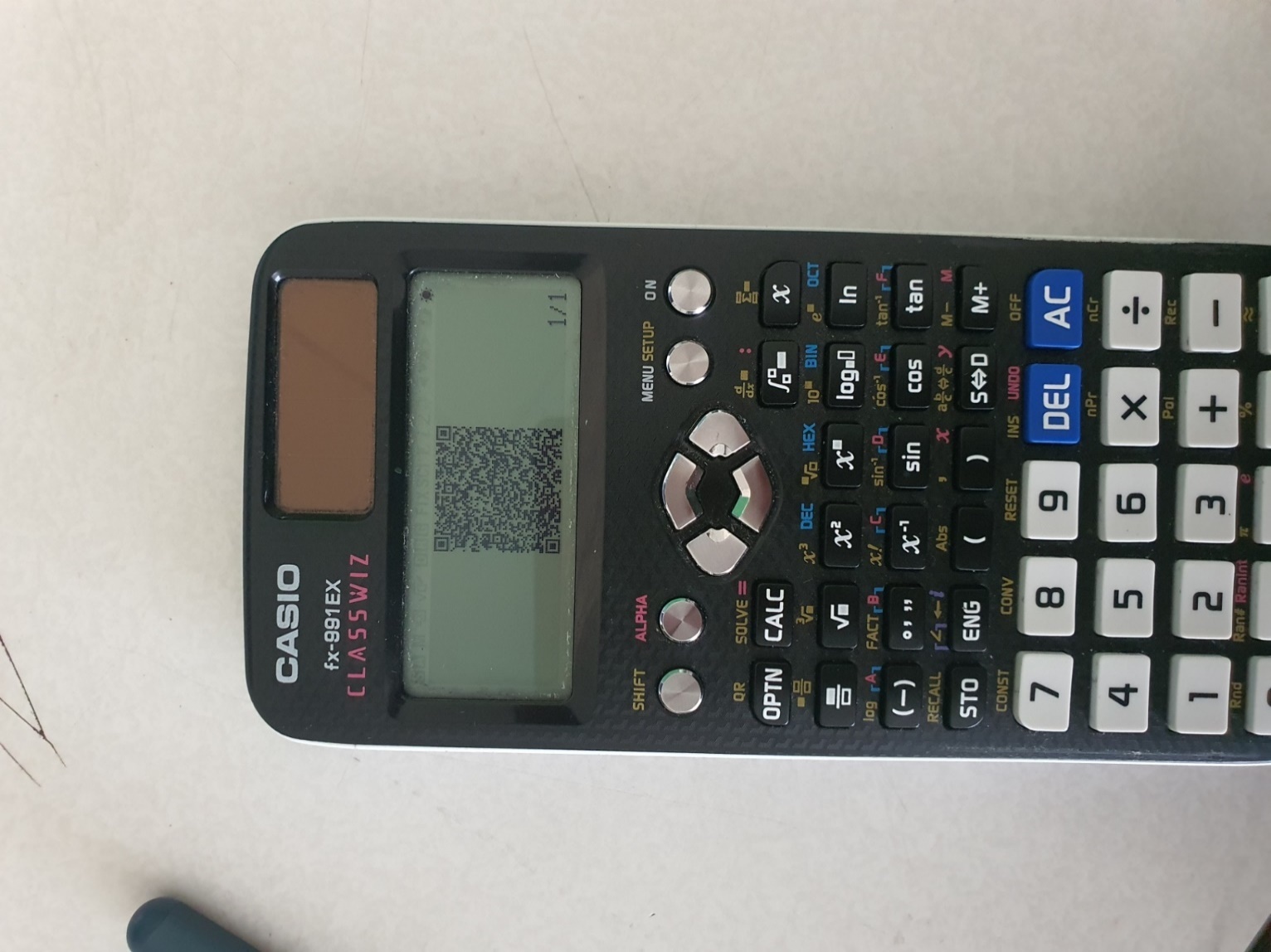
The aim for the COOLCULATOR is very much to be the solution, where it can be a viable and affordable alternative to the usually required Casio FX-991EX.

The FX-991EX calculator uses a small Liquid-crystal display (LCD). This limits the resolution and amount of information they can display while still being legible. However, COOLCULATOR will have a portion of a high-resolution computer screen to use. This will allow for a colour scheme, creative fonts, more forms, and an overall more friendly user-interface.

The menus are very cramped and titles can only be seen when hovering over the option.

The user-entered calculation is shown in top left of the screen, while the answer is displayed at the bottom right corner. However, for longer calculations and visually larger answers, this is inconvenient and they overlap. Making it more difficult to read and understand the information being displayed on the screen.

Answers are displayed dull and colourless. The selected options on the top row are hard to read and understand what each option means.



Casio’s approach to displaying a calculation and answer in a neater and more readable format is to show the maths online, using a QR code storing the URL and calculation data. When scanned with an external device, the Casio website is displayed with the calculation and answer previously displayed on the calculator’s screen.

The data in the QR code is very cramped, meaning the identifying squares for the scanner are very small. This makes it difficult to use the code.

This technique requires another device from Lawrie as he has to scan the code. This creates more hassle and could make him more disorganised as he will have to keep his workings out in a different place. This also wastes valuable time for Lawrie in his maths lessons as he finds his resources.

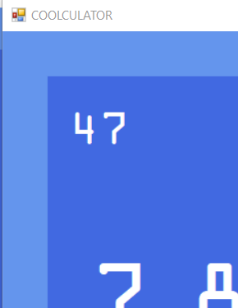
Graphical user interface

Description automatically generated with low confidenceHowever, the data is never actually stored online. The COOLCULATOR will have internet access, meaning the data can be sent to a server, then shared with other people also.

The information is displayed very plain and with little detail. The functions of the page are very limited. It is to be assumed that this feature is simply to save working out for later, or to view calculations in a neater format.

## A picture containing icon Description automatically generatedPROTOTYPE

This is the front end of a simple-style COOLCULATOR. This uses contrasting colours and a consistent, friendly-looking font. It is capable of interpreting individual keypresses made by the user and using this to create a calculation. The answer is then displayed on the top screen.



Minimal formatting is used, where the calculator only displays the buttons and controls with no borders. As well as this, the output only contains one number at a time, as well as the operation clicked. This keeps the display clean and all the on-screen information easy to understand and read. The information uses a friendly font, with a calculator style while still staying legible.

More complex functions will use buttons with the same formatting and style, just sitting horizontally next to other buttons on the calculator.

Icon

Description automatically generatedIcon

Description automatically generated with low confidenceFor example, trigonometry and square root functions will look like this:

Icon

Description automatically generatedThese functions will allow the user to type the numerical value after the open bracket, then select a close bracket when they would like the function to end using a button like this:

* Text

  Description automatically generatedThis calculator looks like the simple version but has more complex buttons using the format above.
* The simple calculator mode can be accessed using the ‘SIMPLE’ button.
* The ‘EQUATION’ button can be used to go to the extra, more complex modes.
* Due to the larger window and ability to use indices, this calculator has a cursor, which points to where the user is typing, and whether or not it is a power or not.

For example, when the user is typing 42, the cursor will look like this:

A picture containing icon

Description automatically generatedIcon

Description automatically generated

Graphical user interface, text

Description automatically generatedHere is the front-end of the online class section:

The fonts and design theme have been kept consistent in the additional windows. The controls are simple and easy to use:

* The CLICK FOR NEW CLASS button resets the results data and uploads it to the FTP server to create a new class.
* Graphical user interface, application

  Description automatically generatedThe SHARE YOUR ANSWER button opens another form where the user can enter a username, then share their working to the class, as seen here.
* It then downloads the class’s data file from the FTP server, adds the student’s calculation and answer to the file, then reuploads it to the server.
* The CLASS RESULTS button downloads the class’s data file from the FTP server and then displays the data in a user-friendly format.
* Text

  Description automatically generatedThe list of names on the left is shown using a ListBox in Visual Studio.
* The calculation(s) is/are displayed on the right of the ListBox using a Label.
* Formatting and font have stayed consistent.
* If there are too many calculations to display, where the label goes off the screen, a vertical scroll bar appears.

Text

Description automatically generated with medium confidence

Text

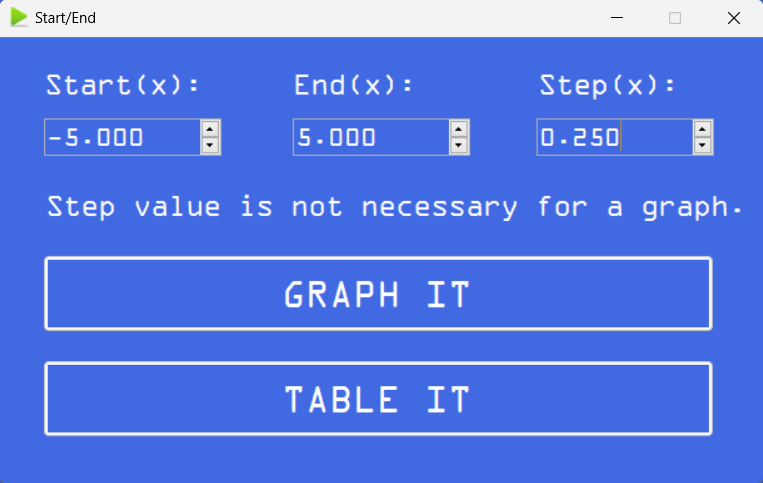
Description automatically generated

In addition, is the extra modes. The screen to build an equation to create a table or graph can be seen here:

* Graphical user interface

  Description automatically generatedThe screen has an area to type in the equation, with available trigonometric functions as clear buttons on the screen.
* A cursor is shown where the user is typing to make it easier to enter the equation.
* A large button to enter the next stage of graphing or table making is clearly marked at the bottom of the screen.
* A button to switch to the solver mode is shown clearly in the top right corner.

Before the user creates their graph or table they must say where they want it to start and end, as well as the step of the table. To do this intuitively, they are presented with a dedicated window:

* The user can enter the start and end values to 3 decimal places, and the step value to 2 decimal places.
* The step value is ignored if entered and the graph mode is chosen.

For the graphing mode, the Visual Studio ‘chart’ tool will be used. Here is an example of using this to graph y=sin(x):

* Chart, line chart

  Description automatically generatedThe graph is clear and optimised for what an A-level mathematics student is used to.
* Both axis cross at zero and background grid lines removed.
* One custom axis label added on the axis, saying “pi”. Custom label adding would be most appropriate when the line crosses the axis.
* Formatting has stayed consistent, with the colour of the line the same as the background.

To draw a tangent, the user will have to give an x value of where they would like the tangent drawn. For the input, they could get a screen which looks like this:

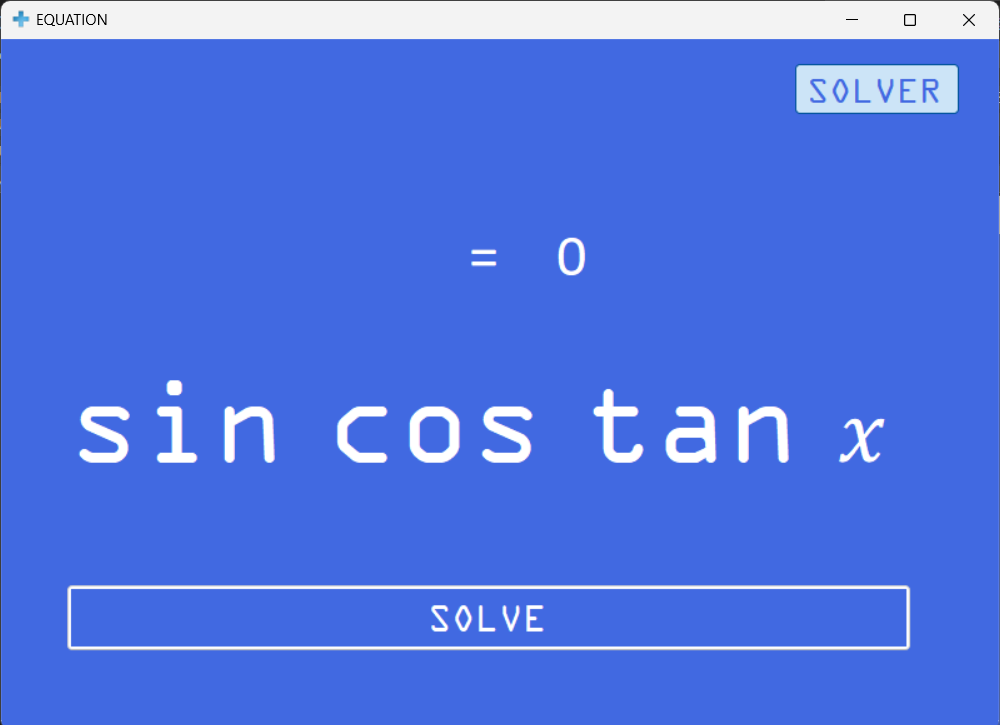
* Icon

  Description automatically generated with medium confidenceThe x value is also to the third decimal place for consistent accuracy.

Graphical user interface

Description automatically generatedWhen the user choses to build a table, they are presented with a ListBox in the ‘x’ column, and plain text in the f(x) column:

* When an x value is selected, the value of the calculation using the selected x is shown in the f(x) column.
* The user also has sorting options. They can click the down arrow to sort the f(x) values in ascending order, or the up arrow for descending order. If they want the original table they created, they can click the ‘ORIGINAL’ button.
* In addition to sorting, they can search in the f(x) values. The user does this by typing a value in the search box and then clicking the search button. The corresponding x value is then selected in the ListBox, revealing their desired f(x).

If the user does not want to create a graph or table, they can choose to solve an equation instead by pressing the solve button. Once the button is pressed, their screen will look like this:

* The user will type to enter their equation on the left side of the equation.
* Once they’ve completed the equation, they can press the ‘SOLVE’ button and a message box will appear stating a value for x.
* If the equation is unsolvable, the message box will appear stating the appropriate error.

## DATA FLOW DIAGRAM FOR FTP SERVER SHARE SYSTEM

Diagram

Description automatically generatedLevel 1:

Level 2:

Diagram

Description automatically generated

## ADDITIONAL REQUIREMENTS AFTER PROTOTYPE

After creating the prototype, I showed it to my user, Lawrie, to get some further feedback and criticism.

**Q – What do you think of the look and feel of the prototype?**

A – I really like the colours and font. It is refreshing to see something other than the usual grey and black screens I am used to. However, I would prefer the COOLCULATOR to fill up its entire window, rather than having a thick border in a different shade of blue. I think this would help maximise the use of screen space.

**Q – What would you improve about the so-far COOLCULATOR?**

A – I am used to scientific functions having more simple buttons. For example, I would rather just have a ‘sin’ button rather than a ‘sin(‘ button, even if it does output an open bracket to the screen.

**Q – That makes sense. Anything else you would add?**

A – My calculator gives an error message depending on what caused the error, and I hope the COOLCULATOR can implement this as I would hate for it to crash at an instance where I make an error. For example, it will show a “Math Error” if I have tried to square root a negative number.

**Final Requirements:**

1. Once the COOLCULATOR loads up, it must go straight to the COOLCULATOR screen where calculations can be made, rather than a menu or something similar, as to save the user hassle and time in a lesson. They will be presented with:

* Numerical buttons showing simply with the number in a white, calculator-style font and on a consistent COOLCULATOR-blue background. When they are clicked, the value of the button is involved in the calculation.
* Operator buttons with formatting consistent with numerical buttons. However, when any of these buttons are clicked, their corresponding operator will be called for the current calculation.
* A CLASS button will show once a calculation has been completed. This will be clicked when the user wants to upload or view other’s answers.
* More buttons for scientific functions with consistent format. These buttons will include:
  + Indices – The user will click this button after typing in a value, then type a number, in superscript, to be the power of that value.
  + Trigonometric functions (sin, cos and tan) – The user will type a value after clicking this button. They will end the function by inserting a close bracket after their value.
  + Constants – These will be used in the same way as the numerical buttons, where the value will be of the constant clicked.
  + Logarithmic functions – This will work similarly to the trigonometric functions, however the user will write a value the want to find the log base-10 of.

1. Also, the COOLCULATOR will display an extra ‘equation’ button to access more complex modes. The next screen will consist of modes which will allow users to:

* Make tables – This involves the user giving an equation, start, end and step, then the COOLCULATOR displays a table showing the index and the corresponding value in an appropriate format.
* Graph sketcher – The same input to the table maker will be implemented here, but instead of displaying the data in the form of a table with rows and columns, the data will be shown across a graph.
* Solve algebraic equations – The user will be able to enter an expression involving an ‘x’ and equal it to 0, and then solve the equation, finding a value for x.

1. There should be a friendly user-interface. The font needs to be ascetically pleasing while being easy to read. The calculator should also have multiple input methods, making it as efficient as possible for the user to enter their maths problems:

* Using on-screen buttons – All functions will be on-screen, meaning the user can call any function they need by using touchscreen and/or their cursor to click the appropriate buttons.
* Keyboard input – Some functions will be able to be typed in using certain keys on a keyboard, if an appropriate key can be allocated to that function, such as:
* Number row – All numbers and decimal points can be inputted from the keyboard by pressing the corresponding number key and the ‘full stop’ key respectively.
* ‘A’ or ‘a’ input – Either of these keys will call the answer function which will make the next numerical value equal to the last answer.
* ‘C’ or ‘c’ input – Either of these keys will call the clear function, which will clear the whole calculation entered.
* ‘+’, ‘-‘, ‘\*’ or ‘x’ or ‘X’, ‘/’, ‘=’ or Enter key, ‘^’ input – Any of these keys will call their corresponding operator for the current calculation.
* ‘^’ – This key can allow the user to enter the superscript mode, allowing them to type the power to their numerical values.

1. The user must be able to upload their answers to an online format with consistent fonts and colours, and share their answers with the class. This will be achieved once the CLASS button is pressed. They should have three options:

* Upload – Be able to enter their names, along with their inputted calculation and outputted answer.
* Download – Retrieve the class’s results straight from online and display the answers with their corresponding student’s names in an easy-to-read table or list format.
* Create a new class – The user should be able to clear the results and create a new blank class where students can input their results.

1. For any extra modes or tasks which take more room on the screen, the program should open new forms over the COOLCULATOR to keep the formatting neat, avoiding overlapping text or taking too much room on the screen. This will apply to any of the extra, complex modes in addition to the CLASS screen used to upload or view other’s answers.
2. Error messages must be showed, describing the error at hand. This includes “Math Error” as seen on the original Casio calculator, which occurs when the calculation is not mathematically possible (e.g. √(-2) = Math Error).
3. Buttons and user interface must be kept as simple and legible as possible. This includes only indicating ‘sin’ rather than ‘sin(‘ or ‘√’ rather than ‘√(‘.
4. As the COOLCULATOR will be used on another device’s screen, the utility of the limited space should be maximised. This means the COOLCULATOR and all its external modes will have as much room of the window taken up as possible while still being user-friendly. This also means formatting needs to be extra neat and minimalistic, like not having a ‘box’ for the COOLCULATOR and stretching it out to the edges of the window itself.

# DESIGN

## MODULAR SYSTEM AND CHART

The system opens with a familiar looking, basic-style calculator. This is where the user can enter numbers and standard operators to calculate an expression. Once a calculation is complete, the user can open the class form. From here, they can bring up a form to share their answers, to view others’ or to reset the class’s.

Instead, the user can switch to the scientific calculator mode, where they can enter complex expressions involving brackets and mathematical functions like sin, cos, tan, log etc. From this calculator they can also access the class form if they’d like to share their scientific calculations.

Or, the user can enter the equation mode from the scientific calculator. The mode will allow the user to give the COOLCULATOR an equation and either graph, table or solve it. For example, they could type a polynomial equation. Here, they could choose to graph the equation in a new form, where a tangent can also be added.

A picture containing timeline

Description automatically generated

## MORE IN-DEPTH STRUCTURE CHARTS

Here is a more detailed version of what happens when the Scientific Calculator is open:Diagram

Description automatically generated

From the scientific calculator screen, the user can either add a character or calculate their inputted equation

This continues to the class form, which a structure chart is shown here for:

If the user clicks the class button, they can either choose to share their results, view class’s results or clear the class’s results.

Diagram

Description automatically generated

Otherwise, there is the graphing mode:

Equation, start/end values

Diagram

Description automatically generated

Each coordinate is calculated to build a smooth line graph

This is where they can draw a tangent:

Diagram

Description automatically generated

Drawing the tangent undergoes the same process as the original graph, where each coordinate is calculated and plotted individually

In addition, there is the table mode route:

Once a table is created, the user can quick sort the table, or search for a term which will involve a quick sort then a binary search.

The method of creating a table is similar to the original Casio calculator. It will add a point to the table which has the output of f(x) in one column and its corresponding x in another column.

Diagram

Description automatically generated

Diagram

Description automatically generatedFinally, the user could choose the solve-equation route:

The looping method described here is known as the Newton-Raphson method. It is used in mathematics as a way to find a solution to an equation equaling 0 using iterations.

## MAIN METHODS EXPLAINED

The simple-style COOLCULATOR:

The first, more simple version of the calculator uses the efficient method of converting to reverse Polish notation first. This is done with the shunting yard algorithm, where the infix input from the user is converted to postfix. To execute this algorithm, each character in the user’s input is looked at individually.

Numbers are added to the end of a temporary string, until an operator is encountered. Then, the temporary string is added on to the end of the final RPN string and emptied. The operator is pushed onto an operator stack. If the priority of the top operator in the stack is less than that of the current operator, then the stack is emptied and added to the final RPN string before the current operator is pushed.

Once the RPN expression has been built, the program can evaluate it using a stack data structure. Here's how it works:

1. Initialize an empty stack.
2. Read each token of the RPN expression from left to right.
3. If the token is an operand, push it onto the stack.
4. If the token is an operator, pop the top two operands from the stack, apply the operator to them, and push the result back onto the stack.
5. Repeat steps 3 and 4 until all tokens have been processed.
6. The result of the RPN expression is the only item remaining on the stack.

Calculating Scientific Expressions:

The scientific calculator with more complex functions uses an arbitrary method with a number queue and an operator queue. Each character in the equation is looked at individually. If the character is a number, it is added to the current number temporary string. If the character is a plus or minus sign, the number in the temporary string is added onto the number queue and the character is added to the operator queue.

If the character is higher priority, like multiplication/division, brackets or indices, the calculation is evaluated with the number in the temporary string. This way, the calculations are always completed in order according to BIDMAS.

When the end of the equation has been reached, the first number is dequeued and set to the answer variable. The operators are dequeued one at a time and depending on what the operator is, the answer will plus/minus the next dequeued number. When the operator queue has been emptied, all calculations involving the numbers in the queue will have been completed and it also would be empty, while the final answer is stored in the answer variable.

Differentiation:

For differentiation, the COOLCULATOR utilises a piece of external software called Maxima. The software can be used in many ways, but the most appropriate is its console form. When a derivative is required, a process is started where the form loads up the Maxima software in a writeable format.

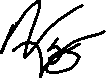
The expression is converted to a form which Maxima can understand, then the differentiation function is written to the console window. The whole window is then read into a string, and it is parsed to find the output given.

Maxima’s differentiation function has limitations, meaning not every expression can be differentiated using it. This could create an error when trying to parse the output, meaning the option to differentiate is not given in these cases.

## DATA DICTIONARY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FORM | SUBROUTINE | DATA TYPE | IDENTIFIER | WHAT IT STORES | EXAMPLE |
| PublicInfo  Module | [Global] | String | calculation | The answer output and user’s inputs are added to this variable to be stored for when the user wants to share their answer with the class. | 2+4=6 |
| String | userEquation | The user’s equation, which they inputted to create either a graph or a table. | 2x+4x2+2 |
| Double | startValue  endValue | The starting and finishing x value of the user’s table or graph. | -5  7 |
| Double | stepValue | The step value between each x value in the user’s table. | 0.5 |
| Constant Double | eValue  piValue | The mathematical constants e and pi. | 2.718  3.142 |
|  | Constant String | SuperScriptCharacters  NormalCharacters | All superscript and non-superscript characters used, in the same order in each string. | ⁰¹²³⁴⁵⁶⁷⁸⁹⁺⁻˟ᐟ⁽⁾ᐢᵉˣˢⁱⁿᵃᵒᵗᶜˡᵍ·  0123456789+-×/()πexsinaotclg. |
| Calculate | String  Integer | expression  token | The equation given by the user to calculate and the current index in the string. | 2+(3\*2.2)  2 |
| Queue of String | operandQueue  operatorQ | All numerical values and plus/minus operators from an expression in order. | 2.2, -5, 7  +, - |
| String | currentNumber | Temporary number being used before reaching an operator and being enqueued. | 10.5 |
| Boolean | isDivide | Whether the current number should be divided or multiplied by the next number. | True |
| String | answer | The output of popping and then calculating the results from the queues until they are emptied, where it will store the final answer. | -9.8 |
| String  Integer | temporaryQueue  bracketCount | When an open bracket is encountered, a temporary queue string is built using a bracket counter. | 3\*2.2  2 |
| String | superscriptQueue | When an power to a function is encountered the superscript characters are converted to their corresponding normal characters and put in a superscript queue. | 2+2 |
| Integer | temporaryToken | An extra index pointer for building queues without changing the original x position. | 5 |
| String | trigFunction | Indicator to which trigonometry function is being used. | s |
| Boolean | isLn | Indicator to the base of the log function. | True |
| Integer | closeBracketCount | The number of close brackets when counting the brackets to determine the end of the queue. | 2 |
| IsNumber  Superscript  ReverseSuperscript  IsSuperscript | String | digit | Parameter for subroutines to check a certain property of a character. | 5 |
| RemoveE | String | number | The numerical value parameter for which an ‘E’ needs to be removed. | 4E+05 |
| String | finalNumber | The number which stores the final value without the ‘E’ and in normal standard form. | 4x105 |
| ReplaceX | String | value | The numerical value which replaces the ‘x’ in the expression. | 2.5 |
| RemoveSpaces  ReplaceX  ConvertToMaxima-  Form  ParseIndices | String | finalExpression | The final equation which stores the modified version of the original input. | 4.2+5 |
| Differentiate  GetOutputArray  ParseIndices | Array of Strings | outputArray | A line in each index of the array from the result of the output from the Maxima software. | 2 5,  x 3x |
| GetOutputArray  ParseIndices | Process | maxima | The process to load the Maxima software, read to it and write to it. | Maxima.bat |
| String | output | The whole output from the process. | [All content in console window] |
| Integer | token | The value which indicates the position of the current character. | 244 |
| ParseIndices | Queue of Powers | powerQueue | Each power from the output array. | 2, 5 |
| Power | temporaryPower | The current power which the value and position is being built before being enqueued. | 5 |
| String | powerValue | The value of the power being dequeued. | 5 |
| Integer | lineNumber | The line currently being read from in the process. | 1 |
| Upload  Download | FtpWebRequest | ftpRequest | The web request to the FTP server. | DirectCast(  URL) |
| Array of Bytes | ftpFile | All the bytes read from the class text file. | [Bytes from content of file] |
| Stream | ftpStream | The stream from the web request to the FTP server. | [Stream to the URL] |
| Stream | ftpOutput | The class file on the FTP server. | Class.txt |
| Integer | bytesIn | The number of bytes read into the buffer. | 1024 |
| Array of Bytes | buffer(1023) | The bytes read from the stream. | [Bytes from content of class file on server] |
| SIMPLEfrm  COOLCULATOR  frm  EQUATION  MAKER  frm | [Global] | String | previousAnswer | The answer from the user’s previous calculation. | 10 |
| Boolean | toSuperscript | Determines whether the next character is displayed in superscript. | False |
| Label | Screenlbl  SimpleScreenlbl  Cursorlbl  Classlbl | The properties of the labels on the screen. | [User-inputted properties] |
| Char | e.KeyChar | The character of the key inputted from the user by keyboard. | 8 |
| Integer | lengthToRemove | The length the text on the screen should be reduced to after a backspace press. | 5 |
| Nlbl\_Click | Label | labelClicked | The label clicked by the user, retrieved by the ‘sender’ parameter. | N8lbl |
| AddCharacter | String | character | The character to be added to the calculator screen. | 8 |
| AddCharacter  CreateRPN  CalculateRPN | Integer | expressionLength | The length of the expression in the current label. | 25 |
| String | expression | The user-inputted expression to be converted to RPN form and calculated. | 4+5 |
| CreateRPN  CalculateRPN  CreateRPN  CalculateRPN | String | RPNExpression | The converted version of the input equation. | 4 5 + |
| String | currentNumber | The temporary numerical value being built before being added to the RPNEq. | 4 |
| Integer | token | The value indicating the index of the current character of the input equation. | 1 |
| Operator | currentOperator | The temporary current operator being built before it is pushed onto the stack. | [+ with its priority] |
| Stack of Operators | operatorStack | The operators pushed from the input equation. | [\*,+,- with assigned priorities] |
| CreateRPN  CalculateRPN  CalculateRPN  GetAnswer | Stack of Strings | numStack | The numbers pushed onto the stack after being built in the curNum string. | 4, 5 |
| String | number1  number2 | The two numbers popped from the top of the stack for a calculation. | 4  5 |
| CalculateRPN  GetAnswer  DisplayAns | Char | operator | The operator used in the current calculation. | + |
| String | result | The output from the calculate subroutine with the user’s expression as input. | 9 |
| [Global] | Label | Sharelbl  NewClasslbl  Resultslbl | The properties of the labels on the screen. | [User-inputted properties] |
| CLASSfrm | Resultslbl\_Click  Sharebtn\_Click  RESULTSfrm\_Load  NamesListbox\_  SelectedIndex  Changed | Array of Strings | fileText | Each index of the array is a line from the class text file. | 16LaadAdi,  4+5=9 |
| CLASSfrm  SHAREfrm  RESULTSfrm | [Global] | ListBox | NamesListbox | Each item in the list box is a student’s username. | 16Laadadi, 16PeatyAle |
| RESULTSfrm | [Global]  NamesListbox\_  SelectedIndex  Changed | Label | Calculationslbl | The calculation answer corresponding to the selected student. | 4+5=9 |
| Integer | token | Pointer to the current character in the index of the fileText array. | 4 |
| NamesListbox\_  SelectedIndex  Changed  [Global] | String | temporary | The temporary number which indicates when a new line should be added to the Calculationslbl. | 22 |
| Chart | Graph | This is where the points are plotted to build the graph and tangent. | [VB chart tool] |
| GRAPHINGfrm | GRAPHING  frm\_Load  Tangentbtn  \_Click | Double | gradient | The gradient at one point in the graph or of the tangent. | 10.2 |
| GRAPHING  frm\_Load  Tangentbtn  \_Click  Tangentbtn  \_Click | Double | previousX  previousY | The previous x and y values plotted on the tangent or graph. | 102  10 |
| String  Double | y  x | The next x and y values calculated to be plotted on the graph or tangent. | 10  2 |
| Integer | currentSeriesIndex | The pointer in the name of a series on the chart to which series is being used. | 1 |
| Double | tangentX  tangentY | The x and y values of the graph where the tangent will meet. | 15  3 |
| [Global] | NumericUpDown | Startupdown  Endupdown  Stepupdown | The numeric values chosen by the user for the start, end and step. | -5  5  0.5 |
| SESfrm | [Global] | Label | currentLabel | The current label which the user is typing into. | LHSlbl |
| EQUATION  MAKER\_  frm | [Global]  SolveEquation | Label | GraphTableExpressionlbl | The user-inputted expression for the graph or table. | 4x+2 |
| Label | RHSlbl  LHSlbl | The right hand sides and left hand sides of the user-inputted equation to be solved. | 4x+2  10 |
| String | RHS  LHS | The user-inputted RHS and LHS of the equation. | 4x+2  10 |
| SolveEquation  [Global] | String | fx | The equation to be solved which equals to 0. | (4x+2)-(10) |
| String | dfx | The derivative of the equation to be solved. | 4 |
| Double | tolerance | The tolerance level to determine how accurate the answer is. | 0.0001 |
| Integer | maxIterations | The maximum number of iterations the subroutine will execute of the Newton-Raphson method. | 1000 |
| Double | x  x1 | The current and next values for x. | 1.54  1.56 |
| Integer | i | The number of iterations executed. | 100 |
| Double | fxVal  dfxVal | The value calculated when substituting the current x value into the fx and dfx expressions. | 10  4 |
| Array of Rows | RowArray(endValue  -startvalue/stepVal  ue) | Each index is a row in the table. | 5, 12 |
| GraphTablebtn\_Click | String | calculationTest | The test value of the result from attempting to calculate the user’s expression | 45 |
| TABLEfrm | [Global]  Sort\_Click | Boolean | desc | Determines whether the table’s fx values are in descending order or not. | True |
| Label | Contentslbl | The fx value corresponding to the selected x value. | 12 |
| ListBox | xlistbox | Each item is the x value in a row of the table. | 5 |
| TextBox | Searchtxt | The user-inputted search term in the table. | 7 |
| Label | lbl | Which arrow label was clicked, retrieved from the ‘sender’ parameter. | ↑ |
| QuickSort | Integer | newBound | The return from the partition function, to be used as the next left or right bound in the next quick sort. | 7 |
| QuickSort  Partition | Integer | left  right | The values indicating the start and end of the list to be quick sorted. | 0  20 |
| Integer | i | The value being counted up to the pivot in the quick sort. | 0 |
| Partition  CreateRowArray | Integer | j | The value being counted down to the pivot in the quick sort. | 20 |
| Double | pivot | The pivot in the list for the quick sort. | 5.5 |
| Row | temporary | The temporary row used for swapping positions in the list. | 5, 12 |
| Integer | rowIndex | The pointer to the current index of the RowArray. | 3 |
| BinarySearch | Double | target | The search term to be found. | 7 |
| BinarySearch | Integer | left  right  middle | The values indicating the start, end and middle values in the list where the item is being searched for. | 0  12  6 |
| Array of Rows | sortedRowArray | The new, sorted version of the original row array. | [Same as RowArray, with different order] |
|  |  |  |  |

## STRUCTURES (RECORDS) USED



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FORM | IDENTIFIER | PROPERTIES | DESCRIPTION | EXAMPLE |
| PublicInfo Module | Power | value As String  position As Integer | This structure is used when parsing the output of the Maxima software. Each power will have a value and a position which lines up with the line below. | value = 5  position = 250 |
| SIMPLEfrm | Operator | character As Char  priority As Integer | Operators in converting to RPN have different priorities. This structure stores an operator’s function and its priority. | character = +  priority = 2 |
| TABLEfrm | Row | xValue As Double  fxValue As String  originalIndex As Integer | Each row on the table has an x value - given by the user, an f(x) value – calculated with the x value and the equation given by the user, and an original index in the table where every row was before a sort. | xValue = 5.5  fxValue = 97.86  origIndex = 4 |

## VALIDATION

To eliminate most validation of user input to the calculator’s screen altogether, the system works that only valid character key presses are registered, like numbers, brackets and operators.

However, even though only valid keypresses are accepted, the user can still enter a mathematically or syntactically invalid expression. This could occur when there are more open brackets than close brackets, when one number has more than one decimal point or the user is attempting to square root a negative number, for example.

Therefore, more validation checks will have to be implemented which identifies errors in the input before attempting to calculate.

An example of this would be using a Try Catch block:

TRY

userInput <- USERINPUT

answer <- CALCULATE(userInput)

CATCH ex AS MathException

answer <- “MATH ERROR.”

CATCH ex AS SyntaxException

Answer <- “SYNTAX ERROR.”

END TRY

In the above pseudocode, the TRY block attempts to read user input, evaluate the expression using the custom CALCULATE function, and print the result to the console. If an error occurs during the evaluation of the expression, the code will CATCH the exception, identify whether it is a MathException or a SyntaxException, and print an appropriate error message to the console. If an unexpected error occurs, the code will catch it in the final CATCH block and print an error message to the console.

In addition to this, validation is needed to make sure the user inputs a valid school username when they share their answer. This will be in the form: 16ReedLew

To accomplish this, the program will use regex, with the pattern: ^\d\d[A-Z][a-z]+[A-Z][a-z][a-z]$

This will involve some basic code, such as:

pattern 🡨 “^\d\d[A-Z][a-z]+[A-Z][a-z][a-z]$”

username 🡨 USERINPUT

valid 🡨 FALSE

IF REGEXMATCH(pattern, username) = TRUE THEN

valid 🡨 TRUE

END IF

## FILE HANDLING

There is one file in use in the COOLCULATOR program which is the Class.txt file. This file stores the name of the students on one line, and the calculations they uploaded on the next. It is stored mainly on the FTP server but is downloaded and saved locally whenever it is in use, to always keep and updated version of the file. When a user shares an answer, it is downloaded, edited locally, then reuploaded.

Once the file is read into a byte array, it is uploaded by first creating an FtpWebRequest which is linked to the FTP server and has the correct credentials. Then, a stream is created. The program uses the stream to overwrite the Class.txt file stored on the server with the byte array from the one stored locally. The stream is then closed and disposed.

When downloading, the process is similar. An empty Class.txt file overrides the current local file. Then, an FtpWebRequest and a stream are created, where the stream reads the bytes from the file in the server into a buffer. After, the buffer is written into the local empty file and the next bytes are read into a new buffer. The process is repeated until all the bytes in the file are read and written into the local file.

An example of what the contents of the text file will look like can be seen here:

A picture containing graphical user interface

Description automatically generated

In the case of multiple calculations, each calculation is separated by a space character.

The text is formatted where the student’s name is written, then their corresponding answer is on the next line.

## PSEUDOCODE

A very interesting subroutine used in my project is the quick-sort procedure. This system uses a partition function and is recursive. How it works can be seen with this level 1 pseudocode:

SUBROUTINE QuickSort(Input the list, the leftmost index and the rightmost index)

Set index to the output of the Partition function

If the left is less than one before the index

Call the QuickSort subroutine with the list, left and one less than index

If the right is bigger than the index

Call the QuickSort subroutine with the list, one above the index and right

END SUBROUTINE

FUNCTION Partition(Input the list, the leftmost index and the rightmost index) As Integer

Set i to the leftmost index

Set j to the rightmost index



Set the pivot to the index in the middle of i and j of the list

While i is less than j:



While the i index of the list is less than pivot

Add 1 to i

While the j index of the list is more than pivot

Minus 1 from j

If I is less than or equal to j then

Swap the values at indexes i and j of the lost

Add 1 to i

Minus 1 from j

Return the value of i

END FUNCTION

Level 2:

SUBROUTINE QuickSort(list, left, right)

index 🡨 Partition(list, left, right)

IF left < index – 1 THEN

QuickSort(list, left, index – 1)

ELSEIF right > index THEN

QuickSort(list, index + 1, right)

END IF

END SUBROUTINE

FUNCTION Partition(list, left, right) AS INTEGER

i 🡨 left

j 🡨 right

pivot 🡨 list((left+right) / 2)

WHILE i <= j

WHILE list(i) < pivot

i = i + 1

END WHILE

WHILE list(j) > pivot

j = j – 1

END WHILE

IF i <= j THEN

temp 🡨 list(i)

list(i) = list(j)

list(j) = temp

i += 1

j -= 1

END IF

END WHILE

RETURN i

END FUNCTION

Another recursive section of the program is used when an open bracket is encountered during the scientific calculation process. It is designed to read everything inside the brackets into another variable, then call itself to calculate that variable, as seen with this level 1 pseudocode:

If the next character in the equation is an open bracket

Empty the queue

Set y to the character index plus 1

Set the bracketCount to 1

Loop until y equals the length of the equation or value at index y of equation equals a close bracket

If value at index y of equation is an open bracket, then

Increase bracketCount by 1

Add 1 to y

Set y to the character index plus 1

Repeat m equals 1 until m equals bracketCount with step 1

While the value at index y of the equation is not a close bracket

Add the value to the queue

Increase y by 1

If m is less than bracketCount then add a close bracket to the queue

Increase y by 1

Set currentNumber to the output of calling the Calculate function with the queue as the parameter

Level 2:

IF equation(character) = “(” THEN

Queue 🡨 Empty

y 🡨 character + 1

bracketCount 🡨 1

REPEAT UNTIL y = LEN(eq) OR equation(y) = “)”

IF equation(y) = “(” THEN

bracketCount = bracketCount + 1

END If

y = y + 1

LOOP

y = character + 1

FOR m 🡨 1 TO bracketCount

REPEAT UNTIL equation(y) = “)”

queue = queue & equation(y)

y = y + 1

LOOP

IF m < bracketCount THEN

queue = queue & “)”

END IF

y = y + 1

NEXT m

currentNumber = CALCULATE(queue)

END IF

In addition to recursion, the program is capable of graph plotting and, even more complex, tangent drawing. For example, here is the level 1 pseudocode for adding a tangent at x-coordinate “x”:

Level 1:

Set userX to the x-coordinate of the point on the graph where the tangent line will be drawn

Set userY to the calculation of the expression after substituting userX

Set gradient to the calculation of the differentiated expression after substituting userX

REPEAT x equals start of graph to end of graph with a very low step value

Set y to gradient \* (x – userX) + userY

Plot the point (x, y) on the tangent series

Level 2:

userX 🡨 USERXINPUT

userY 🡨 CALCULATE(SUBSTITUTEX(expression, userX))

gradient 🡨 CALCULATE(DIFFERENTIATE(SUBSTITUTEX(expression, userX)))

FOR x 🡨 startValue TO endValue STEP stepValue

y 🡨 gradient \* (x – userX) + userY

TANGENTPLOT(x, y)

NEXT x

As well as a graph, the user can display an equation in the form of a table:

FUNCTION generate\_table(equation, start, end, step)

table 🡨 []

x 🡨 start

WHILE x <= end

y 🡨 equation(x)

table.append(x,y)

x <- x + step

END WHILE

RETURN table

END FUNCTION

Example of usage:

equation = f(x): x\*x + 2\*x + 1

start = 0

end = 10

step = 0.5

table = generate\_table(equation, start, end, step)

This will generate a table of (x,y) values, where x ranges from 0 to 10 with a step of 0.5, and y is calculated using the equation y = xx + 2x + 1.

Another example:

equation <- f(x): math.sin(x)

start <- 0

end <- math.pi

step <- 0.1

table <- generate\_table(equation, start, end, step)

This will generate a table of (x,y) values, where x ranges from 0 to pi with a step of 0.1, and y is calculated using the equation y = sin(x).

The COOLCULATOR will also involve some equation solving where an ‘x’ in an equation would be found. The best one-for-all solution for this is the Newton-Raphson method:

FUNCTION newton\_raphson(f, df, x0, tolerance, max\_iterations)

x <- x0

FOR i <- 1 TO max\_iterations

fx <- f(x)

dfx <- df(x)

x\_new <- x - fx/dfx

IF abs(x\_new - x) < tolerance

RETURN x\_new

END IF

x <- x\_new

END FOR

RETURN "Error: Maximum iterations reached."

END FUNCTION

The above pseudocode defines a function named "newton\_raphson" which takes in five parameters:

* f: a function representing the equation that we want to find the root of.
* df: a function representing the derivative of the equation.
* x0: an initial guess for the root of the equation.
* tolerance: the desired level of accuracy for the root.
* max\_iterations: the maximum number of iterations allowed for the algorithm to converge.

The function begins by initializing the variable x with the provided initial guess x0. It then enters into a for loop that runs for the maximum number of iterations, max\_iterations.

Within the loop, the function calculates the value of fx by passing the current value of x to the provided function f and it calculates the value of dfx by passing the current value of x to the provided function df.

The new approximation of the root is then calculated as x\_new = x - fx/dfx using the Newton-Raphson formula.

Then, the function checks if the difference between the new approximation and the previous one is smaller than the desired tolerance level. If the difference is smaller than the tolerance, the function returns the new approximation as the root of the equation. If not, the new approximation becomes the new x value and the loop continues.

If the maximum number of iterations is reached without convergence, the function returns an error message "Error: Maximum iterations reached."

The simple COOLCULATOR mode can use reverse Polish notation and the shunting yard algorithm to evaluate the user’s inputs. Firstly, the user’s input will need to be converted to RPN form, which can be done using code like the shunting yard algorithm pseudocode below:

FOR EACH character IN expression

IF token IS A NUMBER

outputQueue.PUSH(token)

ELSE IF token IS AN OPERATOR

WHILE operatorStack.IS\_NOT\_EMPTY() AND operatorStack.PEEK() HAS HIGHER PRECEDENCE THAN character

outputQueue.PUSH(operatorStack.POP())

operatorStack.PUSH(token)

END\_IF

END\_FOR

WHILE operatorStack.IS\_NOT\_EMPTY()

outputQueue.PUSH(operatorStack.POP())

END WHILE

result = CONVERTSTACKTOSTRING(outputQueue)

The above pseudocode defines a function that takes an input expression as a string and returns the expression in RPN as a string.

The function starts by initializing two stacks: outputQueue and operatorStack. The outputQueue is used to store the final RPN expression, and the operatorStack is used to store operators temporarily.

The input expression is parsed character by character, and each character is processed in the for loop. If the character is a number, it is added to the output queue. If it is an operator, it is compared to the top of the operatorStack, and any operators with higher precedence are popped off the operatorStack and added to the outputQueue. The current operator is then pushed onto the operatorStack.

Once all tokens have been processed, any remaining operators on the operatorStack are popped off and added to the outputQueue.

Finally, the outputQueue is converted to a string using a custom CONVERTSTACKTOSTRING function and returned as the result of the function.

After the algorithm has been completed, the expression now needs to be evaluated. Evaluating an expression in RPN form can be done using pseudocode like below:

FOR EACH character IN RPNexpression

IF character IS A NUMBER

stack.PUSH(character)

ELSE IF character IS AN OPERATOR

num2 🡨 stack.POP()

num1 🡨 stack.POP()

IF character IS ADDITION

result 🡨 num1 + num2

ELSE IF character IS SUBTRACTION

result 🡨 num1 - num2

ELSE IF character IS MULTIPLICATION

result 🡨 num1 \* num2

ELSE IF character IS DIVISION

result 🡨 num1 / num2

stack.PUSH(result)

END\_IF

END\_FOR

answer 🡨 stack.POP()

The above pseudocode defines a function that takes an input expression in RPN as a string and returns the result of evaluating the expression as a number.

The function starts by initializing a stack called stack to store numbers and temporary results. The input expression is parsed character by character, and each character is processed in the FOR loop.

If the character is a number, it is pushed onto the stack. If the character is an operator, the top two numbers on the stack are popped off and the corresponding arithmetic operation is performed on them. The result is then pushed back onto the stack.

Once all characters have been processed, the final result is the only remaining number on the stack, so it is popped off and returned as the result of the function.

## USER INTERFACE DESIGN

The aim for the design of the user interface is to look as clean, friendly and easy-to-understand as possible. This is lacked hugely in alternative calculators, such as the Casio CLASSWIZ.

Graphical user interface, text, application

Description automatically generatedThe initial main scientific calculator screen design looks like this:

Screen continues clean theme and colours. In the actual COOLCULATOR there will not be a visible border.

Cursor designed to be pointing at where the user is currently typing.

CLASS menu option for user to share and view other’s class results.

Dark background colour contrasts well with white text colour.

Spaced out and clean design for on-screen buttons for the user to type in their equation.

Menu option for the COOLCULATOR’s modes fits in with the theme and lines up with the buttons to not look messy.

When using the CLASS feature, the user is met with three screens, as followed:

A picture containing text

Description automatically generated

The user will choose one of the three options by clicking on the text.

Fonts and colours match the original calculator screen.

Clean, simple menu design.

Once the user has completed a calculation and wants to share it, they a brought to a SHARE screen to enter their username and send it off to their classmates:

Graphical user interface

Description automatically generated with medium confidence

User types in their school username into the text box.

Theme is consistent and reversed for the text box to make it clear where the user must type.

SHARE button is made clear with white border around it. User presses this button and then this screen is closed.

Chart, treemap chart

Description automatically generatedThen, the user can also see their classmates’ answers in the RESULTS section, where a screen designed like this is presented to them:

A vertical scroll bar is shown when the contents of the label are overflown.

In the actual COOLCULATOR there will not be a visible border.

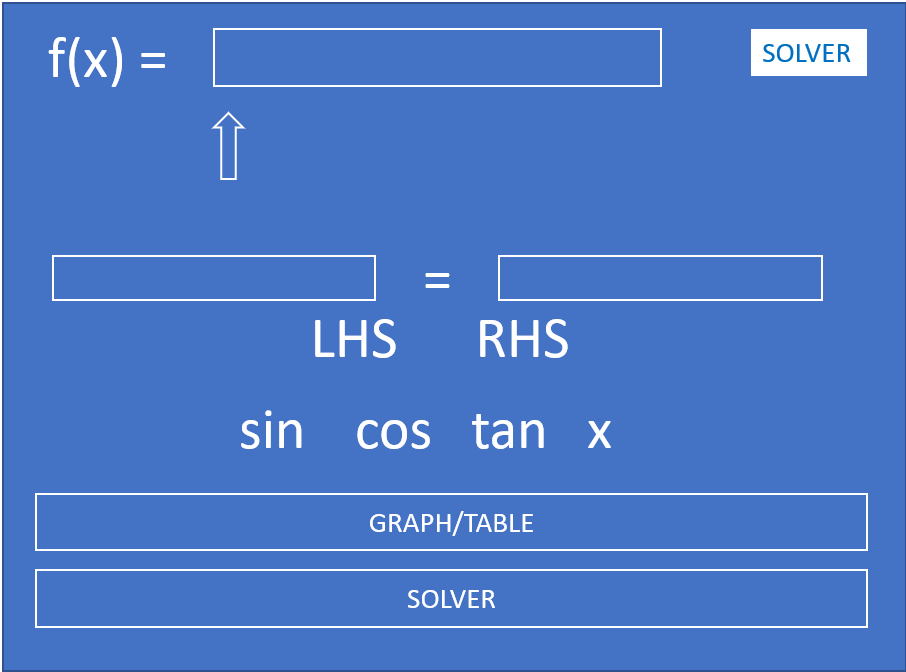
Theme is consistent and colours are reversed for the selected student.

Names are displayed in a ListBox. When a name is selected, their results appear in the adjacent label.

The user is also given a cursor here to assist them in typing a more complex equation.

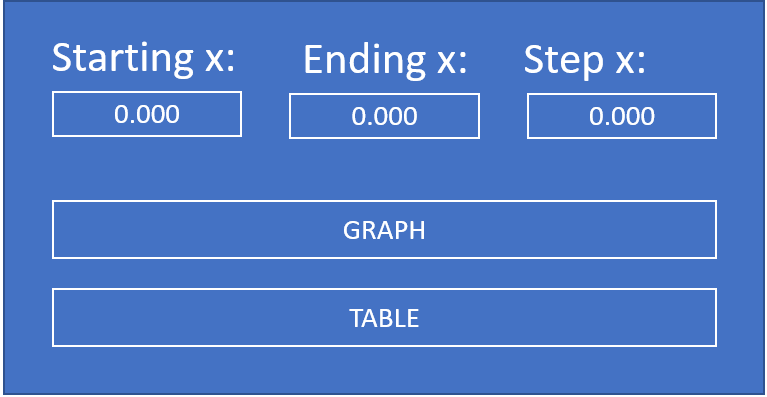
This is the text box where the user can enter their graph/table equation. It will not have a border in the final program.

The user is given a button to switch between the graph or table building mode and the equation solving mode.



Depending on the mode they are in, the user has two buttons to proceed to the next screen. One for their table/graph equation and one for solving.

Once switched to the solver mode, the user gets a LHS text box and a RHS text box to type their equation. They can use the LHS and RHS buttons to switch between the sides.



If the user selected the graph/table, a screen is brought up allowing them to type the starting, ending and step values to 3 decimal places for their build.

They then have the final choice of building a graph or table with their equation using one of the two buttons on screen.

Graphical user interface

Description automatically generated with medium confidence

The user’s graph is drawn and shown, taking up most of the window’s space, being as clear and readable for the user as possible.

If their equation can be differentiated, the user will be given the option to enter an x value for and then draw a tangent on their graph.

Chart, treemap chart

Description automatically generated

If they clicked the table option, the user has sorting abilities. They can sort the f(x) section in descending, ascending and its original order.

The x column will be displayed as a ListBox, where once an x value is selected, the corresponding f(x) value is shown in the f(x) column’s Label.

# IMPLEMENTATION

## COMPLEXITY INDEX

|  |  |  |
| --- | --- | --- |
| **Complexity** | **Description** | **Page** |
| Advanced search routines | Binary search for an f(x) value in the table. | [8](#_TABLEfrm_CLASS)6 |
| Advanced sort routines | Quick sort for the f(x) values in the table. | [8](#_TABLEfrm_CLASS)4 |
| Regular expressions (REGEX) | Requiring the student’s username in the form: 16ReedLew, 22WhiteSim. | [73](#_SHAREfrm_CLASS) |
| Recursion | Used with:  Temporary queues in the arbitrary calculation routine to keep the numerical BIDMAS order.  Recalling the converting an expression to Maxima form subroutine for any superscript powers in an expression.  The quick sort as each side of a pivot is arranged.  Replacing ‘x’ in an expression if the routine needs to be recalled as there is another x character. | [56](#_PublicInfo_MODULE)  [57](#_PublicInfo_MODULE)  [58](#_PublicInfo_MODULE)  [8](#_TABLEfrm_CLASS)4  [61](#_PublicInfo_MODULE) |
| Graph drawing/Dynamic creation of objects | The chart area and each series are dynamically created. Each point is plotted individually and manually until a smooth line is drawn on the graph. | [76](#_GRAPHINGfrm_CLASS) |
| File handling | The Class.txt file is locally stored and read into arrays, where each index of the array is a line in the text file. | [73](#_SHAREfrm_CLASS)  [74](#_RESULTSfrm_CLASS) |
| Use of a server (FTP) | The Class.txt file is uploaded and downloaded to the FTP server for use with other students. | [55](#_PublicInfo_MODULE) |
| Linking to external software | Maxima software is used for differentiation by loading it as a VB Process, reading and writing to it in a console window. Its standard input is redirected, and it is read in full before being parsed into the COOLCULATOR. | [63](#_PublicInfo_MODULE) |
| Complex data structures (stacks and queues) | The simple calculator mode uses reverse Polish notation. To convert the inputted expression into RPN, the shunting yard algorithm uses an operator stack.  Once in RPN, the program uses a number stack to calculate the RPN in the usual way.  The scientific calculator mode uses an arbitrary calculation method which involves building a numerical queue and a plus/minus operator queue.  Power, Operator and Row records are used to store multiple variables about one subject. | [66](#_SIMPLEfrm_CLASS)  [67](#_SIMPLEfrm_CLASS)  [55](#_PublicInfo_MODULE)  [56](#_PublicInfo_MODULE)  [54](#_PublicInfo_MODULE),[65](#_SIMPLEfrm_CLASS),[84](#_TABLEfrm_CLASS) |
| **Total number of lines of code** | **1272** | |

## PublicInfo MODULE

Option Strict On

Imports System.Net.WebRequestMethods.Ftp

Imports System.IO

Imports System.Net

Public Module PublicInfo

Public calculation As String = ""

Public userEquation As String

Public startValue, endValue, stepvalue As Double

Public Const eValue As Double = 2.7182818284590451

Public Const piValue As Double = 3.1415926535897931

Const SuperscriptCharacters As String = "⁰¹²³⁴⁵⁶⁷⁸⁹⁺⁻˟ᐟ⁽⁾ᐢᵉˣˢⁱⁿᵃᵒᵗᶜˡᵍ·"

Const NormalCharacters As String = "0123456789+-×/()πexsinaotclg."

Structure Power

Dim value As String

Dim position As Integer

End Structure

Public Sub Upload()

Dim ftpRequest As FtpWebRequest = DirectCast(WebRequest.Create("ftp://ftp.drivehq.com/Class.txt"), FtpWebRequest) 'Creating the web request

ftpRequest.Credentials = New NetworkCredential("16ReedLew", "COOLCULATORS") 'Setting the username and password for the request

ftpRequest.Method = UploadFile 'Setting the method of the request

Dim ftpFile() As Byte = File.ReadAllBytes("Class.txt") 'Putting contents of text file into an array of bytes

Dim ftpStream As Stream = ftpRequest.GetRequestStream() 'Creating the ftp stream

ftpStream.Write(ftpFile, 0, ftpFile.Length) 'Writing the array of bytes to the stream

ftpStream.Close() 'Closing the stream

ftpStream.Dispose() 'Disposing of the stream

End Sub

Public Sub Download()

Dim ftpOutput As Stream 'Creating the output stream

Dim bytesIn As Integer

Dim buffer(1023) As Byte

Dim ftpRequest As FtpWebRequest = DirectCast(WebRequest.Create("ftp://ftp.drivehq.com/Class.txt"), FtpWebRequest) 'Creating the web request

ftpRequest.Credentials = New NetworkCredential("16ReedLew", "COOLCULATORS") 'Setting the username and password for the request

ftpRequest.Method = DownloadFile 'Setting the method for the request

Dim ftpStream As Stream = ftpRequest.GetResponse.GetResponseStream() 'Creating the ftp stream

ftpOutput = File.Create("Class.txt") 'Setting the output stream to the text file

bytesIn = 1

Do Until bytesIn < 1

bytesIn = ftpStream.Read(buffer, 0, 1024) 'Reading and counting the bytes from the ftp stream into the buffer

If bytesIn > 0 Then ftpOutput.Write(buffer, 0, bytesIn) 'Writing the contents of the buffer to the text file

Loop

ftpOutput.Close() 'Closing the output stream

ftpStream.Close() 'Closing the ftp stream

End Sub

Public Function Calculate(ByVal expression As String) As String

Dim operandQueue, operatorQueue As New Queue(Of String)

Dim token As Integer = 0

Dim currentNumber As String = ""

Dim isDivide As Boolean

Do Until token >= expression.Length 'Parsing each character

If IsNumber(expression(token)) Then

NumberCharacter(expression, currentNumber, token)

ElseIf expression(token) = "×" Or expression(token) = "/" Then

MultiplicationCharacter(expression, token, currentNumber, isDivide)

ElseIf expression(token) = "+" Or expression(token) = "-" Then 'If + or -

AdditionCharacter(expression, token, currentNumber, operandQueue, operatorQueue)

ElseIf expression(token) = "(" Then 'If open bracket

OpenBracketCharacter(expression, token, currentNumber, isDivide)

ElseIf expression(token) = "s" Or expression(token) = "c" Or expression(token) = "t" Then 'if trig

TrigCharacter(expression, token, currentNumber, isDivide)

ElseIf expression(token) = "√" Then

SquareRootCharacter(expression, token, currentNumber, isDivide)

ElseIf expression(token) = "l" Then

LogCharacter(expression, token, currentNumber, isDivide)

ElseIf expression(token) = "π" Or expression(token) = "e" Then

ConstantCharacter(expression, token, currentNumber, isDivide)

ElseIf IsSuperscript(expression(token)) Then ' If indices

SuperscriptCharacter(expression, token, currentNumber)

Else

Return "s" 'Return an appropriate error if invalid character is found

End If

token += 1

Loop

If currentNumber <> "" Then operandQueue.Enqueue(currentNumber) 'Enqueueing the last number

Dim answer As String = operandQueue.Dequeue() 'Setting the answer to the first number

For x = 1 To operatorQueue.Count 'Dequeueing and evaluating the numbers with each operator

Select Case operatorQueue.Dequeue()

Case "+" : answer = CStr(CDbl(answer) + CDbl(operandQueue.Dequeue()))

Case "-" : answer = CStr(CDbl(answer) - CDbl(operandQueue.Dequeue()))

End Select

Next

If answer = "∞" Or answer = "NaN" Then answer = "i" 'Returning appropriate error if calculation is mathematically invalid

Return answer

End Function

Sub NumberCharacter(ByRef expression As String, ByRef currentNumber As String, ByRef token As Integer)

If token > 0 AndAlso expression(token - 1) = ")" Then

expression = expression.Insert(token, "×")

token -= 1

Else

currentNumber &= expression(token) 'Adding the numerical character to the current number

End If

End Sub

Sub MultiplicationCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef isDivide As Boolean)

Select Case expression(token) 'Setting the multiplication to the appropriate function

Case "×"c : isDivide = False

Case "/"c : isDivide = True

End Select

If IsNumeric(expression(token + 1)) Then 'Checking if the next character is a number

Dim temporaryQueue As String = ""

Dim temporaryToken As Integer = token + 1

Do Until temporaryToken = expression.Length OrElse Not IsNumber(expression(temporaryToken)) And Not IsSuperscript(expression(temporaryToken))

temporaryQueue &= expression(temporaryToken) 'Building the queue

temporaryToken += 1

Loop

Select Case expression(token) 'Evaluating the multiplication function

Case "×"c : currentNumber = CStr(CDbl(currentNumber) \* CDbl(Calculate(temporaryQueue)))

Case "/"c : currentNumber = CStr(CDbl(currentNumber) / CDbl(Calculate(temporaryQueue)))

End Select

token += temporaryQueue.Length

End If

End Sub

Sub AdditionCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef operandQueue As Queue(Of String), ByRef operatorQueue As Queue(Of String))

If IsMinus(expression, token) Then

currentNumber = "-"

Else

operandQueue.Enqueue(currentNumber) 'Enqueueing the current number to the number queue

currentNumber = ""

operatorQueue.Enqueue(expression(token)) 'Enqueueing the current operator to the operator queue

End If

End Sub

Sub OpenBracketCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef isDivide As Boolean)

Dim temporaryQueue As String = ""

Dim temporaryToken As Integer = token + 1

Dim bracketCount As Integer = 1

CountBrackets(expression, temporaryToken, bracketCount)

temporaryToken = token + 1

BuildQueue(expression, temporaryToken, bracketCount, temporaryQueue)

token += temporaryQueue.Length + 1

If temporaryToken < expression.Length AndAlso IsSuperscript(expression(temporaryToken)) Then 'Check if there is a power to the contents of the brackets

Dim superscriptQueue As String = ""

BuildSuperscriptQueue(expression, superscriptQueue, temporaryToken, token)

temporaryQueue = CStr(CDbl(Calculate(temporaryQueue)) ^ CDbl(Calculate(superscriptQueue))) 'Evaluating the brackets with its power

Else

temporaryQueue = Calculate(temporaryQueue)

End If

EvaluateCurrentNumber(currentNumber, temporaryQueue, isDivide)

End Sub

Sub TrigCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef isDivide As Boolean)

Dim trigFunction As String = expression(token)

Dim temporaryQueue As String = ""

token += 4

Dim temporaryToken As Integer = token

Dim bracketCount As Integer = 1

CountBrackets(expression, temporaryToken, bracketCount)

temporaryToken = token

BuildQueue(expression, temporaryToken, bracketCount, temporaryQueue)

token += temporaryQueue.Length

If temporaryToken < expression.Length AndAlso IsSuperscript(expression(temporaryToken)) Then 'Check if there is a power to the trig function

Dim superscriptQueue As String = ""

BuildSuperscriptQueue(expression, superscriptQueue, temporaryToken, token)

temporaryQueue = CStr(CDbl(CalculateTrig(temporaryQueue, trigFunction)) ^ CDbl(Calculate(superscriptQueue))) 'Evaluating the trig function with its power

Else

temporaryQueue = CalculateTrig(temporaryQueue, trigFunction)

End If

EvaluateCurrentNumber(currentNumber, temporaryQueue, isDivide)

End Sub

Sub SquareRootCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef isDivide As Boolean)

Dim temporaryQueue As String = ""

Dim temporaryToken As Integer = token + 2

Dim bracketCount As Integer = 1

CountBrackets(expression, temporaryToken, bracketCount)

temporaryToken = token + 2

BuildQueue(expression, temporaryToken, bracketCount, temporaryQueue)

token += temporaryQueue.Length + 3

If temporaryToken < expression.Length AndAlso IsSuperscript(expression(temporaryToken)) Then 'Check if there is a power to the squareroot function

Dim superscriptQueue As String = ""

BuildSuperscriptQueue(expression, superscriptQueue, temporaryToken, token)

temporaryQueue = CStr(Math.Sqrt(CDbl(Calculate(temporaryQueue))) ^ CDbl(superscriptQueue)) 'Evaluating the squareroot function with its power

Else

temporaryQueue = CStr(Math.Sqrt(CDbl(Calculate(temporaryQueue))))

End If

EvaluateCurrentNumber(currentNumber, temporaryQueue, isDivide)

End Sub

Sub LogCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef isDivide As Boolean)

Dim isLn As Boolean

Select Case expression(token + 1) 'Setting the appropriate base to the log function

Case "n"c : isLn = True

Case "o"c : isLn = False

End Select

Dim temporaryQueue As String = ""

Dim bracketCount As Integer = 1

Dim temporaryToken As Integer = token + 3

If isLn = False Then temporaryToken += 1

CountBrackets(expression, temporaryToken, bracketCount)

temporaryToken = token + 3

If isLn = False Then temporaryToken += 1

token = temporaryToken

BuildQueue(expression, temporaryToken, bracketCount, temporaryQueue)

token += temporaryQueue.Length

If temporaryToken < expression.Length AndAlso IsSuperscript(expression(temporaryToken)) Then 'Checking if there is a power to the log function

Dim superscriptQueue As String = ""

BuildSuperscriptQueue(expression, superscriptQueue, temporaryToken, token)

temporaryQueue = CStr(CDbl(CalculateLog(temporaryQueue, isLn)) ^ CDbl(Calculate(superscriptQueue))) 'Evaluating the log function with its power

Else

temporaryQueue = CalculateLog(temporaryQueue, isLn)

End If

EvaluateCurrentNumber(currentNumber, temporaryQueue, isDivide)

End Sub

Sub ConstantCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String, ByRef isDivide As Boolean)

Dim temporaryQueue As String = ""

Select Case expression(token) 'Setting the appropriate constant value

Case "π"c : temporaryQueue = CStr(piValue)

Case "e"c : temporaryQueue = CStr(eValue)

End Select

Dim temporaryToken As Integer = token + 1

If temporaryToken < expression.Length AndAlso IsSuperscript(expression(temporaryToken)) Then 'Checking if the constant has a power

Dim superscriptQueue As String = ""

BuildSuperscriptQueue(expression, superscriptQueue, temporaryToken, token)

temporaryQueue = CStr(CDbl(temporaryQueue) ^ CDbl(Calculate(superscriptQueue))) 'Evaluating the constant with its power

End If

EvaluateCurrentNumber(currentNumber, temporaryQueue, isDivide)

End Sub

Sub SuperscriptCharacter(ByRef expression As String, ByRef token As Integer, ByRef currentNumber As String)

Dim superscriptQueue As String = ""

Dim temporaryToken As Integer = token

Do Until temporaryToken = expression.Length OrElse Not IsSuperscript(expression(temporaryToken))

superscriptQueue &= ReverseSuperscript(expression(temporaryToken)) 'Building the queue for the power

temporaryToken += 1

Loop

currentNumber = CStr(CDbl(currentNumber) ^ CDbl(Calculate(superscriptQueue))) 'Evaluating the current number with its power

token += superscriptQueue.Length - 1

End Sub

Sub CountBrackets(ByRef expression As String, ByRef temporaryToken As Integer, ByRef bracketCount As Integer)

Dim closeBracketCount As Integer = 0

Do Until closeBracketCount = bracketCount 'Counting the brackets until they equal

If expression(temporaryToken) = "(" Then bracketCount += 1

If expression(temporaryToken) = ")" Then closeBracketCount += 1

temporaryToken += 1

Loop

End Sub

Sub BuildQueue(ByRef expression As String, ByRef temporaryToken As Integer, ByRef bracketCount As Integer, ByRef temporaryQueue As String)

For x = 1 To bracketCount 'Building the queue until the last bracket has reached

Do Until expression(temporaryToken) = ")"

temporaryQueue &= expression(temporaryToken)

temporaryToken += 1

Loop

If x < bracketCount Then temporaryQueue &= ")" 'Checking if the last bracket is reached

temporaryToken += 1

Next

End Sub

Sub BuildSuperscriptQueue(ByRef expression As String, ByRef superscriptQueue As String, ByRef temporaryToken As Integer, ByRef token As Integer)

Do Until temporaryToken = expression.Length OrElse Not IsSuperscript(expression(temporaryToken))

superscriptQueue &= ReverseSuperscript(expression(temporaryToken)) 'Adding the characters to the power queue

temporaryToken += 1

Loop

token += superscriptQueue.Length

End Sub

Sub EvaluateCurrentNumber(ByRef currentNumber As String, ByRef temporaryQueue As String, ByRef isDivide As Boolean)

If currentNumber <> "" Then 'Checking if there is a current number

Select Case currentNumber

Case "-" : currentNumber = CStr(-1 \* CDbl(temporaryQueue)) 'Checking if the current number is a minus sign

Case Else

Select Case isDivide 'Evaluating the queue with the current number with the appropriate multiplication function

Case False : currentNumber = CStr(CDbl(currentNumber) \* CDbl(temporaryQueue))

Case True : currentNumber = CStr(CDbl(currentNumber) / CDbl(temporaryQueue))

End Select

End Select

Else

currentNumber = temporaryQueue

End If

End Sub

Function CalculateLog(ByVal temporaryQueue As String, ByVal isLn As Boolean) As String

Select Case isLn 'Evaluating the log function with the appropriate base

Case True : Return CStr(Math.Log(CDbl(Calculate(temporaryQueue))))

Case Else : Return CStr(Math.Log10(CDbl(Calculate(temporaryQueue))))

End Select

End Function

Function CalculateTrig(ByVal temporaryQueue As String, ByVal trigFunction As String) As String

Select Case trigFunction 'Evaluating the appropriate trig function

Case "s" : Return CStr(Math.Sin(CDbl(Calculate(temporaryQueue))))

Case "c" : Return CStr(Math.Cos(CDbl(Calculate(temporaryQueue))))

Case "t" : Return CStr(Math.Tan(CDbl(Calculate(temporaryQueue))))

Case Else : Return "i"

End Select

End Function

Function IsNumber(ByVal digit As String) As Boolean

Select Case digit

Case "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "." : Return True

End Select

Return False

End Function

Function IsMinus(ByVal expression As String, ByVal token As Integer) As Boolean

If token = 0 And expression(token) = "-" Then 'Checking if the first character is a minus sign

Return True

ElseIf (Not IsNumber(expression(token - 1)) And Not IsNumber(ReverseSuperscript(expression(token - 1)))) And expression(token - 1) <> ")" And expression(token) = "-" Then 'Checking if the previous character is numerical

Return True

Else

Return False

End If

End Function

Function RemoveE(ByVal number As String) As String

If number.Contains("E") Then

Dim token As Integer = 0

Dim finalNumber As String = ""

Do Until number(token) = "E"

finalNumber &= number(token) 'Building the final number until an E is reached

token += 1

Loop

finalNumber &= "×10" 'Replacing the E with x10

token += 1

If number(token) = "+" Then 'Checking if the power is positive

token += 1

Else

finalNumber &= "⁻"

token += 1

End If

Do Until token = number.Length

finalNumber &= Superscript(number(token)) 'Adding the power in superscript characters

token += 1

Loop

Return finalNumber

Else

Return number

End If

End Function

Function ReplaceX(ByVal expression As String, ByVal value As String) As String

Dim token As Integer = 0

Dim finalExpression As String = ""

Do Until token = expression.Length OrElse (expression(token) = "x" Or expression(token) = "ˣ")

finalExpression &= expression(token) 'Build the result until an x is reached

token += 1

Loop

If finalExpression = expression Then Return expression 'Return the result if there was no x

If expression(token) = "ˣ" Then

finalExpression &= "⁽"

For temporaryToken = 0 To value.Length - 1

finalExpression &= Superscript(value(temporaryToken)) 'Add the value's characters in superscript to the result

Next

finalExpression &= "⁾"

Else

finalExpression &= "(" & value & ")" 'Add the value in brackets to the result

End If

token += 1

Do Until token = expression.Length

finalExpression &= expression(token) 'Add the rest of the expression to the result

token += 1

Loop

If finalExpression.Contains("x") Or finalExpression.Contains("ˣ") Then finalExpression = ReplaceX(finalExpression, value) 'If there is still an x then recall the function

Return finalExpression

End Function

Function Superscript(ByVal digit As String) As String

If digit = "-" Then Return "⁻"

If NormalCharacters.Contains(digit) Then

Return SuperscriptCharacters(NormalCharacters.IndexOf(digit)) 'Return the corresponding index of the superscript characters string

Else

Return digit

End If

End Function

Function ReverseSuperscript(ByVal digit As String) As String

If digit = "⁻" Then Return "-"

If SuperscriptCharacters.Contains(digit) Then

Return NormalCharacters(SuperscriptCharacters.IndexOf(digit)) 'Return the corresponding index of the normal characters string

Else

Return digit

End If

End Function

Function IsSuperscript(ByVal digit As String) As Boolean

If SuperscriptCharacters.Contains(digit) Then 'Check if the character is in the superscript characters string

Return True

Else

Return False

End If

End Function

Function Differentiate(ByVal expression As String) As String

expression = "diff(" & expression & ",x);"

Dim outputArray As String() = GetOutputArray(expression)

If outputArray(1)(2) = "i" Then

Return RemoveSpaces(outputArray(0))

Else

Return ParseIndices(outputArray)

End If

End Function

Function RemoveSpaces(ByVal expression As String) As String

Dim finalExpression As String = ""

Dim token As Integer = 5

Do Until token = expression.Length

If expression(token) <> " " Then finalExpression &= expression(token) 'Ignore space characters

token += 1

Loop

If finalExpression.Contains("log") Then finalExpression = finalExpression.Replace("log", "ln")

Return finalExpression

End Function

Function ConvertToMaximaForm(ByVal expression As String) As String

Dim token As Integer = 0

Dim finalExpression As String = ""

Dim temporaryQueue As String = ""

Do Until token >= expression.Length

If token > 0 Then

Select Case expression(token) 'Checks if a \* needs to be added

Case "x"c, "("c, "s"c, "c"c, "t"c : If IsNumber(expression(token - 1)) Or expression(token - 1) = ")" Or expression(token - 1) = "x" Then finalExpression &= "\*"

End Select

End If

If IsSuperscript(expression(token)) Then 'Checks if a ^ needs to be added

finalExpression &= "^("

Do Until token >= expression.Length OrElse (Not IsSuperscript(expression(token)))

temporaryQueue &= ReverseSuperscript(expression(token)) 'Build the queue of the power

token += 1

Loop

finalExpression &= ConvertToMaximaForm(temporaryQueue) & ")" 'Recall the function for the power queue

temporaryQueue = ""

End If

If token < expression.Length AndAlso expression(token) = "×" Then

finalExpression &= "\*" 'Replacing the times symbol with a \*

ElseIf token < expression.Length Then

finalExpression &= expression(token) 'Add the rest of the characters to the result

End If

token += 1

Loop

Return finalExpression

End Function

Function GetOutputArray(ByVal expression As String) As String()

Dim maxima As New Process()

Dim output As String

With maxima

.StartInfo.FileName = "maxima-5.46.0\bin\maxima.bat" 'Loading the maxima software

.StartInfo.UseShellExecute = False

.StartInfo.RedirectStandardInput = True

.StartInfo.RedirectStandardOutput = True

.Start()

.StandardInput.Write(expression) 'Write to the software

.StandardInput.Close()

output = .StandardOutput.ReadToEnd() 'Read from the software

.WaitForExit()

End With

Dim outputArray(2) As String

Dim token As Integer = 244 'Read from the first output line

Dim lineNumber As Integer = 0

Do Until token = output.Length OrElse (output(token) = vbLf And lineNumber = 1)

If Not output(token) = vbLf Then

outputArray(lineNumber) &= output(token) 'Build the output array with each line of the output

Else

lineNumber = 1 'Change to the second output line

End If

token += 1

Loop

Return outputArray

End Function

Function ParseIndices(ByVal outputArray As String()) As String

Dim powerQueue As New Queue(Of Power)

Dim temporaryPower As Power

temporaryPower.value = ""

Dim token As Integer = 0

Dim finalExpression As String = ""

Dim expression As String = outputArray(1)

Do Until token = outputArray(0).Length

Do Until outputArray(0)(token) <> " " 'Ignore initial spaces on the output line

token += 1

Loop

temporaryPower.position = token 'Set power position to the current index

Do Until token = outputArray(0).Length OrElse outputArray(0)(token) = " "

temporaryPower.value &= outputArray(0)(token) 'Build the value of the power

token += 1

Loop

powerQueue.Enqueue(temporaryPower) 'Enqueue the built power

temporaryPower.value = ""

Loop

token = 0

Dim powerValue As String

For x = 0 To powerQueue.Count

Do Until token = expression.Length

finalExpression &= expression(token) 'Build final expression

If expression(token) <> " " Then 'Check if there could be a power

If powerQueue.Count <> 0 AndAlso token = powerQueue.Peek.position - 1 Then 'Check if the position matches the position of the next power in the queue

powerValue = powerQueue.Dequeue.value

For y = 0 To powerValue.Length - 1

finalExpression &= Superscript(powerValue(y)) 'Add power to the expression in superscript characters

Next

End If

End If

token += 1

Loop

Next

Return RemoveSpaces(finalExpression)

End Function

End Module

## SIMPLEfrm CLASS

Option Strict On

Public Class SIMPLEfrm

Dim previousAnswer As String

Structure [Operator]

Dim character As Char

Dim priority As Integer

End Structure

Private Sub SIMPLEfrm\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles Me.KeyPress

Select Case e.KeyChar 'Comparing the inputting character with available valid characters

Case "0"c, "1"c, "2"c, "3"c, "4"c, "5"c, "6"c, "7"c, "8"c, "9"c, "."c, "+"c, "-"c : AddCharacter(e.KeyChar)

Case "X"c, "x"c, "\*"c : AddCharacter("×")

Case "/"c : AddCharacter("÷")

Case "C"c, "c"c : SimpleScreenlbl.Text = ""

Case "A"c, "a"c : AddCharacter(previousAnswer)

Case "="c, ChrW(Keys.Enter) : DisplayAns()

Case ChrW(Keys.Back)

If SimpleScreenlbl.Text.Length > 0 Then

SimpleScreenlbl.Text = SimpleScreenlbl.Text.Substring(0, SimpleScreenlbl.Text.Length - 1)

End If

End Select

End Sub

Private Sub Nlbl\_Click(sender As Object, e As EventArgs) Handles N0lbl.Click, N1lbl.Click, N2lbl.Click, N3lbl.Click, N4lbl.Click, N5lbl.Click, N6lbl.Click, N7lbl.Click, N8lbl.Click, N9lbl.Click, Decimallbl.Click, Dividelbl.Click, Addlbl.Click, Multiplylbl.Click, Minuslbl.Click

Dim labelClicked As Label = CType(sender, Label)

AddCharacter(labelClicked.Text)

End Sub

Private Sub ANSlbl\_Click(sender As Object, e As EventArgs) Handles ANSlbl.Click

AddCharacter(previousAnswer)

End Sub

Sub AddCharacter(ByVal character As String)

If SimpleScreenlbl.Text.Length <= 10 Then 'Checking if the character will fit on the screen

SimpleScreenlbl.Text &= character

End If

End Sub

Private Sub Clearlbl\_Click(sender As Object, e As EventArgs) Handles Clearlbl.Click

SimpleScreenlbl.Text = ""

End Sub

Private Sub Scientificlbl\_Click(sender As Object, e As EventArgs) Handles Scientificlbl.Click

COOLCULATORFfrm.Show()

Hide()

End Sub

Private Sub Equalslbl\_Click(sender As Object, e As EventArgs) Handles Equalslbl.Click

DisplayAns()

End Sub

Sub DisplayAns()

Dim result As String

Try

result = CalculateRPN(SimpleScreenlbl.Text)

If result = "∞" Then

MsgBox("MATH ERROR.",, "COOLCULATOR")

Else

SimpleScreenlbl.Text = result

previousAnswer = SimpleScreenlbl.Text

End If

Catch ex As Exception

MsgBox("SYTNAX ERROR.",, "COOLCULATOR") 'Display appropriate error if calculation of the user's expression fails

End Try

End Sub

Function CreateRPN(ByVal expression As String) As String

Dim operatorStack As New Stack(Of [Operator])

Dim RPNExpression As String = ""

Dim currentNumber As String = ""

Dim currentOperator As [Operator]

Dim token As Integer = 0

Do Until token >= expression.Length

If IsNumber(expression(token)) Then

currentNumber &= expression(token) 'Adding every number to the current number

ElseIf IsOperator(expression(token)) Then

RPNExpression &= currentNumber & " " 'Seperating each entry to the final expression with a space

currentNumber = ""

currentOperator = CreateOperator(expression(token))

While operatorStack.Count > 0 AndAlso (operatorStack.Peek.priority <= currentOperator.priority) 'Checking if operators in the stack need to be popped first

RPNExpression &= operatorStack.Pop.character 'Empty all operators with less or the same priority

End While

operatorStack.Push(currentOperator)

End If

token += 1

Loop

RPNExpression &= currentNumber & " " 'Add last number to final expression

For y = 1 To operatorStack.Count

RPNExpression &= operatorStack.Pop.character & " " 'Add any remaining operators

Next

Return RPNExpression.Substring(0, RPNExpression.Length - 1) 'Returning expression without final space character

End Function

Function CalculateRPN(ByVal expression As String) As String

Dim RPNExpression As String = CreateRPN(expression)

Dim token As Integer = 0

Dim currentNumber As String = ""

Dim numStack As New Stack(Of String)

Dim number1, number2 As String

Do Until token = RPNExpression.Length

If IsNumber(RPNExpression(token)) Then

currentNumber &= RPNExpression(token) 'Storing numbers in a temporary string

ElseIf RPNExpression(token) = " " And currentNumber <> "" Then

numStack.Push(currentNumber) 'Pushes the temporary number to the stack

currentNumber = ""

ElseIf IsOperator(RPNExpression(token)) Then

number2 = numStack.Pop() 'Pop the top two numbers on the stack

number1 = numStack.Pop()

numStack.Push(GetAnswer(number1, number2, RPNExpression(token))) 'Push the result of the calculation onto the number stack

End If

token += 1

Loop

Return numStack.Pop()

End Function

Function GetAnswer(ByVal number1 As String, ByVal number2 As String, ByVal [operator] As Char) As String

Select Case [operator] 'Returning the result of the calculation with the given operator and operands

Case "+"c : Return CStr(CDbl(number1) + CDbl(number2))

Case "-"c : Return CStr(CDbl(number1) - CDbl(number2))

Case "×"c : Return CStr(CDbl(number1) \* CDbl(number2))

Case "÷"c : Return CStr(CDbl(number1) / CDbl(number2))

Case Else : Return "s"

End Select

End Function

Function CreateOperator(ByVal token As Char) As [Operator]

Dim [operator] As [Operator]

[operator].character = token

Select Case token

Case "+"c, "-"c

[operator].priority = 2 'Assigns priority depending on the character

Case Else

[operator].priority = 1

End Select

Return [operator]

End Function

Function IsOperator(ByVal token As String) As Boolean

Select Case token

Case "+"c, "-"c, "÷"c, "×"c : Return True

End Select

Return False

End Function

End Class

## COOLCULATORfrm CLASS

Option Strict On

Public Class COOLCULATORFfrm

Dim previousAnswer As String

Dim toSuperscript As Boolean = False

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

Cursorlbl.Location = New Point(Screenlbl.Location.X + 26, Screenlbl.Location.Y + 2)

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size)

KeyPreview = True

Classlbl.Hide()

End Sub

Private Sub COOLCULATORfrm\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles Me.KeyPress

Select Case e.KeyChar 'Comparing the inputting character with available valid characters

Case "\*"c, "x"c, "X"c : AddCharacter("×")

Case "0"c, "1"c, "2"c, "3"c, "4"c, "5"c, "6"c, "7"c, "8"c, "9"c, "/"c, "+"c, "-"c, "("c, ")"c, "."c : AddCharacter(e.KeyChar)

Case "^"c : SuperscriptClick()

Case "="c, ChrW(Keys.Enter) : DisplayAns()

Case "C"c, "c"c : Clear()

Case "A"c, "a"c : AddCharacter("(" & previousAnswer & ")")

Case "e"c, "E"c : AddCharacter("e")

Case ChrW(Keys.Back)

Dim lengthToRemove As Integer = 1

If Screenlbl.Text.Length >= 3 Then 'Changing length of temporary string if there is a function

Select Case Screenlbl.Text(Screenlbl.Text.Length - 2)

Case "s"c, "ˢ"c, "ᵍ"c, "g"c : lengthToRemove = 4

Case "n"c, "ⁿ"c

If Screenlbl.Text(Screenlbl.Text.Length - 3) = "l" Then

lengthToRemove = 3

Else

lengthToRemove = 4

End If

End Select

End If

If Screenlbl.Text.Length > 0 Then Screenlbl.Text = Screenlbl.Text.Substring(0, Screenlbl.Text.Length - lengthToRemove) 'Show the contents of the screen without the last character(s)

If Cursorlbl.Location.X > 60 Then 'Moving cursor back one place

Select Case toSuperscript

Case True : Cursorlbl.Location = New Point(Cursorlbl.Location.X - 17, Cursorlbl.Location.Y)

Case False : Cursorlbl.Location = New Point(Cursorlbl.Location.X - 24, Cursorlbl.Location.Y)

End Select

End If

If Screenlbl.Text.Length = 0 Then 'Adjusting cursor to appropriate size and position if screen empty or last character is now superscript

toSuperscript = False

Cursorlbl.Location = New Point(Screenlbl.Location.X + 26, Screenlbl.Location.Y + 2)

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size)

ElseIf Not IsSuperscript(Screenlbl.Text(Screenlbl.Text.Length - 1)) And toSuperscript = True Then

toSuperscript = False

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size)

ElseIf IsSuperscript(Screenlbl.Text(Screenlbl.Text.Length - 1)) And toSuperscript = False Then

toSuperscript = True

Cursorlbl.Font = New Font("Calibri", 20)

End If

End Select

End Sub

Sub AddCharacter(ByVal character As String)

Classlbl.Hide()

For token = 0 To character.Length - 1 'Adding each character of the input

If Screenlbl.Text.Length < 30 Then

Select Case toSuperscript 'Checking whether to add the character in superscript or normal form

Case True

Screenlbl.Text &= Superscript(character(token))

Case False

Screenlbl.Text &= character(token)

End Select

End If

Next

Cursorlbl.Location = New Point(Cursorlbl.Location.X + (character.Length \* 8), Cursorlbl.Location.Y) 'Adjusting cursor to correct position

Cursorlbl.Location = New Point(Cursorlbl.Location.X + (character.Length \* 17), Cursorlbl.Location.Y)

End Sub

Private Sub Nlbl\_Click(sender As Object, e As EventArgs) Handles N0lbl.Click, N1lbl.Click, N2lbl.Click, N3lbl.Click, N4lbl.Click, N5lbl.Click, N6lbl.Click, N7lbl.Click, N8lbl.Click, N9lbl.Click, pilbl.Click, elbl.Click, Decimallbl.Click, Dividelbl.Click, Addlbl.Click, Multiplylbl.Click, Minuslbl.Click, CloseBracketlbl.Click, OpenBracketlbl.Click

Dim labelClicked As Label = CType(sender, Label)

AddCharacter(labelClicked.Text)

End Sub

Private Sub Funclbl\_Click(sender As Object, e As EventArgs) Handles Sinlbl.Click, Coslbl.Click, Tanlbl.Click, lnlbl.Click, loglbl.Click

Dim labelclicked As Label = CType(sender, Label)

AddCharacter(labelclicked.Text & "(")

End Sub

Private Sub Sqrtlbl\_Click(sender As Object, e As EventArgs) Handles Sqrtlbl.Click

toSuperscript = False

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size)

AddCharacter("√(")

End Sub

Private Sub ANSlbl\_Click(sender As Object, e As EventArgs) Handles ANSlbl.Click

AddCharacter("(" & previousAnswer & ")")

End Sub

Private Sub Equalslbl\_Click(sender As Object, e As EventArgs) Handles Equalslbl.Click

DisplayAns()

End Sub

Private Sub Clearlbl\_Click(sender As Object, e As EventArgs) Handles Clearlbl.Click

Clear()

End Sub

Private Sub Powerlbl\_Click(sender As Object, e As EventArgs) Handles Powerlbl.Click

SuperscriptClick()

End Sub

Sub Clear()

Screenlbl.Text = ""

calculation = ""

Cursorlbl.Location = New Point(Screenlbl.Location.X + 26, Screenlbl.Location.Y + 2) 'Resetting cursor postion

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size) 'Resetting cursor size

toSuperscript = False

Classlbl.Hide()

End Sub

Sub DisplayAns()

calculation &= Screenlbl.Text 'Stores calculation for sharing

Dim result As String

Try

result = RemoveE(Calculate(Screenlbl.Text))

If result = "i" Then 'Displays appropriate error if calculation returns specified error

MsgBox("MATH ERROR.",, "COOLCULATOR")

calculation = calculation.Substring(0, calculation.Length - Screenlbl.Text.Length)

ElseIf result = "s" Then

MsgBox("SYNTAX ERROR.",, "COOLCULATOR")

calculation = calculation.Substring(0, calculation.Length - Screenlbl.Text.Length)

Else

Screenlbl.Text = result

calculation &= "=" & Screenlbl.Text & " " 'Stores answer for sharing

previousAnswer = Screenlbl.Text

Cursorlbl.Location = New Point(Screenlbl.Location.X + 26 + (Screenlbl.Text.Length \* 24), Screenlbl.Location.Y + 2) 'Resetting cursor postition

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size) 'Resetting cursor size

toSuperscript = False

Classlbl.Show()

End If

Catch ex As Exception

MsgBox("SYNTAX ERROR.",, "COOLCULATOR") 'Returns appropriate error if calculation fails

calculation = calculation.Substring(0, calculation.Length - Screenlbl.Text.Length) 'Removing calculation for sharing as it resulted in an error

End Try

End Sub

Sub SuperscriptClick()

If Screenlbl.Text.Length > 0 Then

If Not toSuperscript Then 'Adjusting cursor size to the current superscript state

Cursorlbl.Font = New Font("Calibri", 20)

Else

Cursorlbl.Font = New Font("Calibri", Screenlbl.Font.Size)

End If

toSuperscript = Not toSuperscript

End If

End Sub

Private Sub Classlbl\_Click(sender As Object, e As EventArgs) Handles Classlbl.Click

CLASSfrm.Show()

End Sub

Private Sub Equationlbl\_Click(sender As Object, e As EventArgs) Handles Equationlbl.Click

EQUATIONMAKERfrm.Show()

End Sub

Private Sub Simplelbl\_Click(sender As Object, e As EventArgs) Handles Simplelbl.Click

Hide()

SIMPLEfrm.Show()

End Sub

Sub COOLCULATORfrm\_Close(sender As Object, e As FormClosedEventArgs) Handles Me.FormClosed

SIMPLEfrm.Close()

End Sub

End Class

## CLASSfrm CLASS

Option Strict On

Imports System.IO

Public Class CLASSfrm

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

Sharelbl.Text = "SHARE YOUR ANSWER" : NewClasslbl.Text = "CLICK FOR NEW CLASS"

End Sub

Private Sub Sharelbl\_Click(sender As Object, e As EventArgs) Handles Sharelbl.Click

SHAREfrm.Show()

End Sub

Private Sub NewClasslbl\_Click(sender As Object, e As EventArgs) Handles NewClasslbl.Click

NewClasslbl.Cursor = Cursors.WaitCursor

If NewClasslbl.Text = "CLICK FOR NEW CLASS" Then

Try

File.WriteAllText("Class.txt", " ") 'Erase all contents of text file

Upload() 'Upload empty text file

NewClasslbl.Text = "NEW CLASS CREATED."

Sharelbl.Text = "SHARE YOUR ANSWER"

Catch ex As Exception 'Catch errors due to network issues

NewClasslbl.Text = "CLICK FOR NEW CLASS"

Sharelbl.Text = "ANSWER SHARED."

MsgBox(ex.Message) 'Display appropriate message for the error

End Try

End If

NewClasslbl.Cursor = Cursors.Hand

End Sub

Private Sub Resultslbl\_Click(sender As Object, e As EventArgs) Handles Resultslbl.Click

RESULTSfrm.Show()

End Sub

End Class

## SHAREfrm CLASS

Option Strict On

Imports System.IO

Imports System.Text.RegularExpressions.Regex

Public Class SHAREfrm

Private Sub Sharebtn\_Click(sender As Object, e As EventArgs) Handles Sharebtn.Click

If Nametxt.Text <> "" And Match(Nametxt.Text, "^\d\d[A-Z][a-z]+[A-Z][a-z][a-z]$").Success Then 'Check if the username is in the form 16ReedLew

Sharebtn.Cursor = Cursors.WaitCursor

Try

Download() 'Download the text file

Dim fileText As String() = File.ReadAllLines("Class.txt") 'Read the contents of the text file into an array

Array.Resize(fileText, fileText.Length + 2) 'Add two lines to the results

fileText(fileText.Length - 2) = Nametxt.Text 'Set the first line to the username

fileText(fileText.Length - 1) = calculation 'Set the second name to their calculation

File.WriteAllLines("Class.txt", fileText) 'Write the whole array to the text file

Try

Upload() 'Upload the new text file

CLASSfrm.Sharelbl.Text = "ANSWER SHARED."

Catch ex As Exception 'Catch any erros due to network issues

CLASSfrm.Sharelbl.Text = "SHARE YOUR ANSWER"

MsgBox(ex.Message,, "COOLCULATORS") 'Display appropriate message for the error

End Try

CLASSfrm.NewClasslbl.Text = "CLICK FOR NEW CLASS"

Sharebtn.Cursor = Cursors.Hand

Close() 'Close form after sharing is complete

Catch ex As Exception 'Catch any erros due to network issues

MsgBox(ex.Message,, "COOLCULATOR") 'Display appropriate message for the error

End Try

Else

MsgBox("Please enter your school username.",, "COOLCULATOR") 'Ask user to change the format of their username

End If

End Sub

End Class

## RESULTSfrm CLASS

Option Strict On

Imports System.IO

Public Class RESULTSfrm

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

Cursor = Cursors.WaitCursor

Try

Download() 'Download the current text file

Dim fileText As String() = File.ReadAllLines("Class.txt") 'Read the contents of the text file to an array

If fileText.Length > 1 Then 'Checking if the file has at least one student (one student uses two lines of the file)

VScrollBar.Hide()

NamesListbox.Items.Clear()

For token = 1 To fileText.Length - 1 'Add all names to the ListBox

If CDbl(token) Mod 2 = 1 Then NamesListbox.Items.Add(fileText(token))

Next

NamesListbox.SelectedIndex = 0 'Set result to the first student

Else

MsgBox("No results yet.",, "COOLCULATOR") 'Display appropriate message if class results are empty

Close()

End If

Catch ex As Exception 'Catch any errors due to network issues

MsgBox(ex.Message) 'Display appropriate message for the error

Close()

End Try

Cursor = Cursors.Hand

End Sub

Private Sub NamesListbox\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles NamesListbox.SelectedIndexChanged

Calculationslbl.Text = ""

Dim fileText As String() = File.ReadAllLines("Class.txt") 'Read contents of the text file to an array

Dim token As Integer = 0

Dim temporary As Integer = 0

Do Until token >= fileText((2 \* NamesListbox.SelectedIndex) + 2).Length - 2 'Add character by character from the line in the array

Do Until fileText((2 \* NamesListbox.SelectedIndex) + 2)(token) = " " 'Add until a space is reached

Calculationslbl.Text &= fileText((2 \* NamesListbox.SelectedIndex) + 2)(token)

If (token - temporary) > 22 Then Calculationslbl.Text &= vbNewLine : temporary = token 'If the character has reached the end of the form then start a new line

token += 1

Loop

Calculationslbl.Text &= vbNewLine 'Start a new line

token += 1

temporary = token 'Set the current character to the start of the line

Loop

If Calculationslbl.Size.Height > 319 Then

VScrollBar.Show() 'Show the scroll bar if the calculations go off the screen vertically

Else

VScrollBar.Hide()

End If

End Sub

Private Sub VScrollBar\_Scroll(sender As Object, e As ScrollEventArgs) Handles VScrollBar.Scroll

Calculationslbl.Location = New Point(Calculationslbl.Location.X, CInt(25 - ((2 \* Calculationslbl.Size.Height) / 319) \* VScrollBar.Value))

End Sub

End Class

## GRAPHINGfrm CLASS

Option Strict On

Imports System.Windows.Forms.DataVisualization.Charting.SeriesChartType

Public Class GRAPHINGfrm

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

COOLCULATORFfrm.Hide()

Tangentbtn.Show() : Tangentupdown.Show() : Tangentxlbl.Show()

Try

Dim test As Double = CDbl(Calculate(ReplaceX(Differentiate(ConvertToMaximaForm(userEquation)), CStr(startValue))))

Catch ex As Exception

Tangentbtn.Hide() : Tangentupdown.Hide() : Tangentxlbl.Hide() 'Hiding all tangent related controls if the derivative cannot be found

End Try

Graph.Titles.Add("Your line:") 'Dynamically editing the graph

Graph.ChartAreas.Clear()

Graph.ChartAreas.Add("Default") 'Dynamically creating a chart area

With Graph.ChartAreas("Default").AxisX

.MajorGrid.Enabled = False

.MinorGrid.Enabled = False

.Crossing = 0

End With

With Graph.ChartAreas("Default").AxisY

.MajorGrid.Enabled = False

.MinorGrid.Enabled = False

.Crossing = 0

End With

With Graph 'Dynamically creating a series on the graph for the plot

.Series.Clear()

.Series.Add("Plot")

.Series("Plot").Color = Color.RoyalBlue

.Series("Plot").ChartType = Line

.Series("Plot").IsVisibleInLegend = False

End With

Dim y As String

Dim previousX, previousY, gradient As Double

Dim currentSeriesIndex As Integer = 2

If HasTrigFunction(userEquation) Then

stepvalue = ((endValue - startValue) / 1000) \* piValue 'If a trig function is involved, the step value is in phase with pi

Else

stepvalue = ((endValue - startValue) / 1000)

End If

For x = startValue To endValue Step stepvalue

y = Calculate(ReplaceX(userEquation, RemoveE(CStr(x)))) 'Calculating the y by subsituting the x in the user's equation

If y <> "i" Then gradient = (CDbl(y) - previousY) / (x - previousX) 'Does not calculate the gradient if the result is a math error

With Graph

If y <> "i" AndAlso (gradient < 100000 And gradient > -100000) Then 'Ignores the point if it is a math error

.Series(.Series.Count - 1).Points.AddXY(x, y) 'Add the point

ElseIf y <> "i" Then 'Dynamically creating a new series if the gradient is too high, most likely meaning an asymptote

.Series.Add("Plot" & currentSeriesIndex)

.Series("Plot" & currentSeriesIndex).Color = Color.RoyalBlue

.Series("Plot" & currentSeriesIndex).ChartType = Line

.Series("Plot" & currentSeriesIndex).IsVisibleInLegend = False

currentSeriesIndex += 1

End If

End With

If y <> "i" Then

previousX = x

previousY = CDbl(y)

End If

Graph.ChartAreas("Default").AxisX.RoundAxisValues()

Graph.ChartAreas("Default").AxisY.RoundAxisValues()

Next

End Sub

Private Sub Tangentbtn\_Click(sender As Object, e As EventArgs) Handles Tangentbtn.Click

Dim tangentX As Double = Tangentupdown.Value 'Setting the x value of the tangent to the user's input

Dim tangentY As Double = CDbl(Calculate(ReplaceX(userEquation, RemoveE(CStr(tangentX))))) 'Calculating the y by subsituting the x in the user's equation

Dim gradient, previousX, previousY, y As Double

gradient = CDbl(Calculate(ReplaceX(Differentiate(ConvertToMaximaForm(userEquation)), CStr(tangentX)))) 'Calculating the gradient by subsituting the x in the derivative of the equation

With Graph

Try

.Series.Add("Tangent")

Catch ex As Exception

.Series.RemoveAt(1) 'If there is already a tangent series, it is removed and a new one is added and created dynamically

.Series.Add("Tangent")

End Try

.Series("Tangent").Color = Color.Red

.Series("Tangent").ChartType = Line

.Series("Tangent").IsVisibleInLegend = True

.Series("Plot").IsVisibleInLegend = True

.ChartAreas("Default").AxisY.CustomLabels.Clear()

.ChartAreas("Default").AxisX.CustomLabels.Clear()

End With

For x = startValue To endValue Step ((endValue - startValue) / 100)

y = (gradient \* (x - tangentX)) + tangentY 'Calculating the y at each point using the mathematical formula: y-y1=m(x-x1)

Graph.Series("Tangent").Points.AddXY(x, y) 'Add the point to the series

If (x > 0 And previousX < 0) Or (x < 0 And previousX > 0) Then 'If the line has crossed the y axis, a label at that point is added

Graph.ChartAreas("Default").AxisY.CustomLabels.Add(previousY, y, CStr(Math.Round((previousY + y) / 2, 3)))

End If

If (y > 0 And previousY < 0) Or (y < 0 And previousY > 0) Then 'If the line has crossed the x axis, a label at that point is added

Graph.ChartAreas("Default").AxisX.CustomLabels.Add(previousX, x, CStr(Math.Round((previousX + x) / 2, 3)))

End If

previousY = y

previousX = x

Next

Graph.ChartAreas("Default").AxisX.RoundAxisValues()

Graph.ChartAreas("Default").AxisY.RoundAxisValues()

End Sub

Sub GRAPHINGfrm\_Closed(sender As Object, e As FormClosedEventArgs) Handles Me.FormClosed

COOLCULATORFfrm.Show()

End Sub

Function HasTrigFunction(ByVal userEquation As String) As Boolean

If userEquation.Contains("sin") Or userEquation.Contains("cos") Or userEquation.Contains("tan") Then 'Checking for normal trig function

Return True

ElseIf userEquation.Contains("ˢⁱⁿ") Or userEquation.Contains("ᶜᵒˢ") Or userEquation.Contains("ᵗᵃⁿ") Then 'Checking for a superscript trig function

Return True

Else

Return False

End If

End Function

End Class

## SESfrm CLASS

Option Strict On

Public Class SESfrm

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

EQUATIONMAKERfrm.Hide()

startValue = 0

endValue = 0

End Sub

Private Sub Startupdown\_ValueChanged(sender As Object, e As EventArgs) Handles Startupdown.ValueChanged

startValue = Startupdown.Value

End Sub

Private Sub Endupdown\_ValueChanged(sender As Object, e As EventArgs) Handles Endupdown.ValueChanged

endValue = Endupdown.Value

End Sub

Private Sub stepupdown\_ValueChanged(sender As Object, e As EventArgs) Handles stepupdown.ValueChanged

stepvalue = stepupdown.Value

End Sub

Private Sub Graphbtn\_Click(sender As Object, e As EventArgs) Handles Graphbtn.Click

If startValue < endValue Then 'The start value must be less than the end value

Try

Hide()

GRAPHINGfrm.Show()

Catch ex As Exception

MsgBox("GRAPHING ERROR.")

End Try

Else

MsgBox("Start value must be less than End value.",, "COOLCULATOR") 'Ask the user to reinput the start and end values

End If

End Sub

Private Sub Tableitbtn\_Click(sender As Object, e As EventArgs) Handles Tableitbtn.Click

If startValue < endValue And stepvalue > 0 Then

Hide()

TABLEfrm.Show()

ElseIf startValue >= endValue Then

MsgBox("Start value must be less than End value.",, "COOLCULATOR") 'Ask the user to reinput the start and end values

ElseIf stepvalue = 0 Then

MsgBox("Step value must be more than zero for a table.",, "COOLCULATOR") 'Ask the user to input a step value

End If

End Sub

End Class

## EQUATIONMAKERfrm CLASS

Option Strict On

Public Class EQUATIONMAKERfrm

Dim toSuperscript As Boolean = False

Dim currentLabel As Label

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

currentLabel = GraphTableExpressionlbl 'Set first label used for input as the graph/table one

SolverCheck.Checked = False

Cursorlbl.Location = New Point(GraphTableExpressionlbl.Location.X + 26, GraphTableExpressionlbl.Location.Y + 2) 'Set cursor to default position

Cursorlbl.Font = New Font("Calibri", GraphTableExpressionlbl.Font.Size)

KeyPreview = True

Solvebtn.Hide() : LHSlbl.Hide() : RHSlbl.Hide() : Equalslbl.Hide() : LHSbtn.Hide() : RHSbtn.Hide() 'Hide all solving-related buttons and labels

End Sub

Private Sub EQUATIONMAKERfrm\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles Me.KeyPress

Select Case e.KeyChar 'Compare the inputted character with valid characters

Case "\*"c : AddCharacter("×")

Case "0"c, "1"c, "2"c, "3"c, "4"c, "5"c, "6"c, "7"c, "8"c, "9"c, "/"c, "+"c, "-"c, "("c, ")"c, "."c : AddCharacter(e.KeyChar)

Case "^"c

If currentLabel.Text.Length > 0 Then

If Not toSuperscript Then

Cursorlbl.Font = New Font("Calibri", 20) 'Adjust cursor to superscript size

Else

Cursorlbl.Font = New Font("Calibri", GraphTableExpressionlbl.Font.Size) 'Adjust cursor to superscript location

End If

toSuperscript = Not toSuperscript

End If

Case ChrW(Keys.Back)

Dim lengthToRemove As Integer = 1

If currentLabel.Text.Length >= 3 Then 'Changing length of temporary string if there is a function

Select Case currentLabel.Text(currentLabel.Text.Length - 2)

Case "s"c, "ˢ"c, "n"c, "ⁿ"c : lengthToRemove = 4

End Select

End If

If currentLabel.Text.Length > 0 Then currentLabel.Text = currentLabel.Text.Substring(0, currentLabel.Text.Length - lengthToRemove) 'Show the contents of the screen without the last character(s)

If Cursorlbl.Location.X > 60 Then 'Moving cursor back one place

Select Case toSuperscript

Case True : Cursorlbl.Location = New Point(Cursorlbl.Location.X - 17, Cursorlbl.Location.Y)

Case False : Cursorlbl.Location = New Point(Cursorlbl.Location.X - 24, Cursorlbl.Location.Y)

End Select

End If

If GraphTableExpressionlbl.Text.Length = 0 Then 'Adjusting cursor to appropriate size and position if screen empty or last character is now superscript

toSuperscript = False

Cursorlbl.Location = New Point(GraphTableExpressionlbl.Location.X + 26, GraphTableExpressionlbl.Location.Y + 2)

Cursorlbl.Font = New Font("Calibri", GraphTableExpressionlbl.Font.Size)

ElseIf Not IsSuperscript(GraphTableExpressionlbl.Text(GraphTableExpressionlbl.Text.Length - 1)) And toSuperscript = True Then

toSuperscript = False

Cursorlbl.Font = New Font("Calibri", GraphTableExpressionlbl.Font.Size)

ElseIf IsSuperscript(GraphTableExpressionlbl.Text(GraphTableExpressionlbl.Text.Length - 1)) And toSuperscript = False Then

toSuperscript = True

Cursorlbl.Font = New Font("Calibri", 20)

End If

Case "x"c, "X"c : AddCharacter("x")

End Select

End Sub

Sub AddCharacter(ByVal character As String)

Dim expressionLength As Integer = 10

If currentLabel.Equals(GraphTableExpressionlbl) Then expressionLength += 20 'Increase length limit to the label if the user is typing to the graph/table equation

For token = 0 To character.Length - 1

If currentLabel.Text.Length < expressionLength Then

Select Case toSuperscript 'Checking whether to add the character in superscript or normal form

Case True

currentLabel.Text &= Superscript(character(token))

calculation &= Superscript(character(token))

Case False

currentLabel.Text &= character(token)

calculation &= character(token)

End Select

End If

Next

Cursorlbl.Location = New Point(Cursorlbl.Location.X + (character.Length \* 8), Cursorlbl.Location.Y) 'Adjusting cursor location

Cursorlbl.Location = New Point(Cursorlbl.Location.X + (character.Length \* 17), Cursorlbl.Location.Y)

End Sub

Private Sub Sinlbl\_Click(sender As Object, e As EventArgs) Handles Sinlbl.Click, Coslbl.Click, Tanlbl.Click

Dim labelClicked As Label = CType(sender, Label)

AddCharacter(labelClicked.Text & "(")

End Sub

Private Sub xlbl\_Click(sender As Object, e As EventArgs) Handles xlbl.Click

AddCharacter("x")

End Sub

Private Sub SolverCheck\_CheckedChanged(sender As Object, e As EventArgs) Handles SolverCheck.CheckedChanged

If SolverCheck.Checked Then 'Hide all graph/table related buttons and labels if solver is checked

GraphTablebtn.Hide() : fxlbl.Hide() : GraphTableExpressionlbl.Hide()

Solvebtn.Show() : LHSlbl.Show() : RHSlbl.Show() : Equalslbl.Show() : LHSbtn.Show() : RHSbtn.Show()

Cursorlbl.Hide()

currentLabel = LHSlbl

Else

Solvebtn.Hide() : LHSlbl.Hide() : RHSlbl.Hide() : Equalslbl.Hide() : LHSbtn.Hide() : RHSbtn.Hide()

GraphTablebtn.Show() : fxlbl.Show() : GraphTableExpressionlbl.Show()

Cursorlbl.Show()

Cursorlbl.Location = New Point(GraphTableExpressionlbl.Location.X + 26, GraphTableExpressionlbl.Location.Y + 2)

currentLabel = GraphTableExpressionlbl

End If

End Sub

Private Sub LHSbtn\_Click(sender As Object, e As EventArgs) Handles LHSbtn.Click

currentLabel = LHSlbl

End Sub

Private Sub RHSbtn\_Click(sender As Object, e As EventArgs) Handles RHSbtn.Click

currentLabel = RHSlbl

End Sub

Private Sub GraphTablebtn\_Click(sender As Object, e As EventArgs) Handles GraphTablebtn.Click

userEquation = GraphTableExpressionlbl.Text

Try

Dim calculationTest As String = Calculate(ReplaceX(userEquation, "1"))

If calculationTest = "i" Then 'Displays appropriate error if calculation returns specified error

MsgBox("MATH ERROR.",, "COOLCULATOR")

ElseIf calculationTest = "s" Then

MsgBox("SYNTAX ERROR.",, "COOLCULATOR")

Else

userEquation = GraphTableExpressionlbl.Text

SESfrm.Show()

End If

Catch ex As Exception

MsgBox("SYNTAX ERROR",, "COOLCULATOR")

End Try

End Sub

Private Sub Solvebtn\_Click(sender As Object, e As EventArgs) Handles Solvebtn.Click

Try

MsgBox(SolveEquation(LHSlbl.Text, RHSlbl.Text),, "COOLCULATOR") 'Attempt to solve the equation

Catch ex As Exception

MsgBox("CANNOT SOLVE.",, "COOLCULATOR") 'Display appropriate message if an error occurs in the solving

End Try

End Sub

Function SolveEquation(LHS As String, RHS As String) As String

Dim fx As String = "(" & LHS & ")-(" & RHS & ")"

Dim dfx As String = Differentiate(ConvertToMaximaForm(fx)) 'Set dfx to the derivative

Dim tolerance As Double = 0.0001

Dim maxIterations As Integer = 1000

Dim x As Double = 1

Dim x1 As Double

Dim i As Integer = 0

Dim fxValue, dfxValue As Double

Do Until i >= maxIterations 'Iterate x1 = x - (fx/dfx) according to the Newton-Raphson method

fxValue = CDbl(Calculate(ReplaceX(fx, CStr(x))))

dfxValue = CDbl(Calculate(ReplaceX(dfx, CStr(x))))

x1 = x - (fxValue / dfxValue)

If Math.Abs(x1 - x) < tolerance Then Return "x = " & x1 'Checking if the difference between them is less than the tolerence

x = x1

i += 1

Loop

Return "NO SOLUTION FOUND." 'Return appropriate message if a solution is not found

End Function

End Class

## TABLEfrm CLASS

Option Strict On

Public Class TABLEfrm

Dim RowArray(CInt((endValue - startValue) / stepvalue)) As Row

Dim isDescending As Boolean = False

Structure Row

Dim xValue As Double

Dim fxValue As String

Dim originalIndex As Integer

End Structure

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

SESfrm.Hide()

CreateRowArray()

End Sub

Private Sub xlistbox\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles xlistbox.SelectedIndexChanged

Select Case isDescending 'Selecting the index from the start or end of the table depends on if it is in descending order

Case True : Contentslbl.Text = RowArray(RowArray.Length - 1 - xlistbox.SelectedIndex).fxValue

Case False : Contentslbl.Text = RowArray(xlistbox.SelectedIndex).fxValue

End Select

End Sub

Private Sub Sort\_Click(sender As Object, e As EventArgs) Handles AscArrlbl.Click, DescArrlbl.Click, Originallbl.Click

Dim labelClicked As Label = CType(sender, Label)

QuickSort(RowArray, 0, RowArray.Length - 1) 'Sort the table for the binary search

xlistbox.Items.Clear() 'Empty ListBox for reinputting

If labelClicked.Text = "↓" Then 'For ascending, sorted values are added in order

For y = 0 To RowArray.Length - 1

xlistbox.Items.Add(RowArray(y).xValue)

Next

isDescending = False

ElseIf labelClicked.Text = "↑" Then 'For descending, sorted values are added in reverse

For y = RowArray.Length - 1 To 0 Step -1

xlistbox.Items.Add(RowArray(y).xValue)

Next

isDescending = True

Else

CreateRowArray() 'For original sort, the original values are created again

End If

xlistbox.SelectedIndex = 0 'Reset to first index of the ListBox

End Sub

Sub QuickSort(ByRef RowArr() As Row, ByRef left As Integer, ByRef right As Integer)

Dim newBound As Integer = Partition(RowArr, left, right) 'Setting the new end to the list to be sorted in an integer

If left < newBound - 1 Then QuickSort(RowArr, left, newBound - 1) 'Recall the quick sort with the adjusted list using the new bound

If newBound < right Then QuickSort(RowArr, newBound, right)

End Sub

Function Partition(ByRef RowArr() As Row, ByRef left As Integer, ByRef right As Integer) As Integer

Dim i As Integer = left

Dim j As Integer = right

Dim temporary As Row

Dim pivot As Double = CDbl(RowArr(CInt(Math.Truncate((i + j) / 2))).fxValue)

While i <= j

While CDbl(RowArr(i).fxValue) < pivot 'Searching for a row smaller than the pivot

i += 1

End While

While CDbl(RowArr(j).fxValue) > pivot 'Searching for a row greater than the pivot

j -= 1

End While

If i <= j Then

temporary = RowArr(i) 'Swapping the rows when the row below the pivot is greater than one above the pivot

RowArr(i) = RowArr(j)

RowArr(j) = temporary

i += 1

j -= 1

End If

End While

Return i 'Return where the rows were swapped

End Function

Sub CreateRowArray()

Dim rowIndex As Integer = 0

Dim calculationResult As String

For x = startValue To endValue Step stepvalue

With RowArray(rowIndex)

.xValue = Math.Round(x, 7) 'Set the x value to each x

calculationResult = Calculate(ReplaceX(userEquation, RemoveE(CStr(x))))

If calculationResult <> "i" Then

.fxValue = CStr(Math.Round(CDbl(calculationResult), 9)) 'Set the fx to the calculation of with the x

Else

.fxValue = "NaN"

End If

End With

rowIndex += 1

Next

xlistbox.Items.Clear() 'Empty the ListBox

For rowIndex = 0 To RowArray.Length - 1

xlistbox.Items.Add(RowArray(rowIndex).xValue) 'Add each row's x to the ListBox

Next

xlistbox.SelectedIndex = 0 'Reset to the first index

isDescending = False 'It is now in the orginal order and not descending

End Sub

Function BinarySearch(target As Double) As String

Dim left As Integer = 0

Dim sortedRowArray As Row() = CType(RowArray.Clone, Row()) 'Cloning the row array into a new array

For n = 0 To sortedRowArray.Length - 1

sortedRowArray(n).originalIndex = n 'Setting the current index as an original index before sorting

Next

QuickSort(sortedRowArray, 0, sortedRowArray.Length - 1) 'Sort the array

Dim right As Integer = sortedRowArray.Length - 1

Dim middle As Integer

While left <= right

middle = (left + right) \ 2 'Find the middle value

If CDbl(sortedRowArray(middle).fxValue) = target Then 'If the value has been found then return its original index

Select Case isDescending

Case False : Return CStr(sortedRowArray(middle).originalIndex) 'Returning the orginal index will select the correct position in the unsorted array

Case True : Return CStr(sortedRowArray.Length - 1 - sortedRowArray(middle).originalIndex)

End Select

ElseIf CDbl(sortedRowArray(middle).fxValue) < target Then 'If the target has not been found, adjust the left and right bounds for the next iteration

left = middle + 1

Else

right = middle - 1

End If

End While

MsgBox("NOT FOUND.",, "COOLCULATOR")

Return "NOT FOUND." 'If a value was not found, an appropriate message is returned

End Function

Private Sub Searchbtn\_Click(sender As Object, e As EventArgs) Handles Searchbtn.Click

Dim searchResult As String = BinarySearch(CDbl(Searchtxt.Text))

If IsNumeric(searchResult) Then 'If a correct value was found, select the correct corresponding row index in the table

xlistbox.SelectedIndex = CInt(searchResult)

End If

End Sub

End Class

# TESTING

Test formatting:

* Test Number
* Description
* Type and Description of Test Data/Input
* Expected Outcome
* Actual Outcome
* Result

## SIMPLE-STYLE CALCULATOR MODE

* 1.1
* Press character “4” key on the keyboard while the screen is empty.
* **TYPICAL**, Keypress = 4 Key
* The character “4” is shown on the screen.
* A picture containing text

  Description automatically generatedA picture containing diagram

  Description automatically generatedAs expected:
* **PASS**
* 1.2
* Deleting the “4” character currently on the screen by using the backspace button.
* **TYPICAL**, Keypress = Backspace Key
* The screen returns to being empty.
* A picture containing text

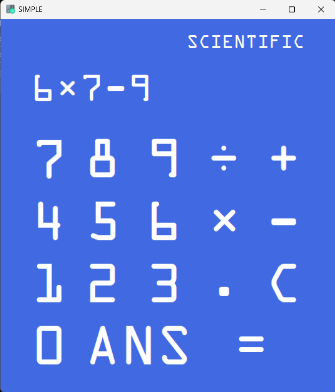
  Description automatically generatedA picture containing diagram

  Description automatically generatedAs expected:
* **PASS**
* 1.3
* Using the keyboard to enter an invalid character to the empty screen.
* **ERRONEOUS**, Keypress = g Key
* The keypress is ignored, and no character is added to the screen, meaning it remains empty.
* A picture containing text

  Description automatically generatedAs expected:
* **PASS**
* 1.4
* Using an on-screen button to input a valid character to the empty screen.
* **TYPICAL**,Button Click = 5 button
* The character “5” is shown on the screen.
* A picture containing text

  Description automatically generatedA picture containing text

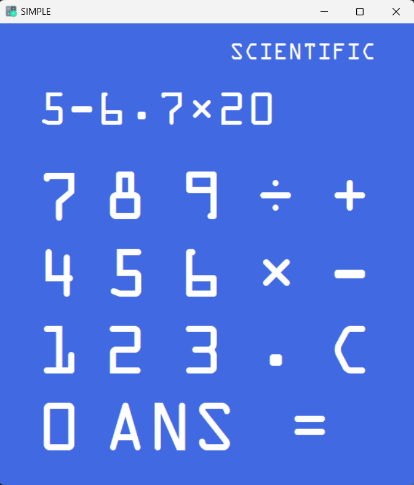
  Description automatically generatedAs expected:
* **PASS**
* 1.5
* Entering and calculating a valid expression.
* **TYPICAL**, Expression = 6\*7-9
* The correct answer “33” is returned and shown on the screen.
* A picture containing text

  Description automatically generatedAs expected:
* **PASS**
* 1.6
* Entering and calculating a syntactically invalid expression.
* **ERRONEOUS**, Expression = 4+6-
* A message box is shown displaying “SYNTAX ERROR.”
* Diagram

  Description automatically generatedAs expected:
* **PASS**
* 1.7
* Entering and calculating a mathematically invalid expression.
* **ERRONEOUS**, Expression = 4/0
* A message box is shown displaying “MATH ERROR.”
* Diagram

  Description automatically generatedAs expected:
* **PASS**
* 1.8
* Clearing the screen to start a new calculation.
* **TYPICAL**, Button Click = C button
* A picture containing text

  Description automatically generatedAll contents of the screen emptied.
* A picture containing text

  Description automatically generatedAs expected:
* **PASS**
* 1.9
* Entering an calculating a valid expression in reverse BIDMAS order, including a decimal number.
* **TYPICAL**, Expression = 5-6.7\*20
* The correct answer “-129” is returned and shown on the screen.
* A picture containing text

  Description automatically generatedAs expected:
* **PASS**
* 1.10
* Switching to the scientific calculator mode.
* **TYPICAL**, Button Click = SCIENTIFIC Button
* The scientific calculator window opens.
* As expected:

A picture containing text

Description automatically generated

Text

Description automatically generated

* **PASS**

## SCIENTIFIC CALCULATOR MODE

* 2.1
* Using keyboard input for numbers and their powers, and on-screen buttons for functions.
* **TYPICAL**, Expression = 42+sin(5.6)
* The expression is shown on the screen, with the cursor adjusting position for the indices.
* Text

  Description automatically generatedText

  Description automatically generatedText

  Description automatically generatedAs expected:
* **PASS**
* 2.2
* Pressing the backspace key to delete a single number, then pressing it once more to delete a whole function.
* **TYPICAL**, Key Press = Backspace Key x2
* The character “5” should be removed, then the four characters in the “sin(“ function should be removed all at once.
* Text

  Description automatically generatedText

  Description automatically generatedAs expected:

Text

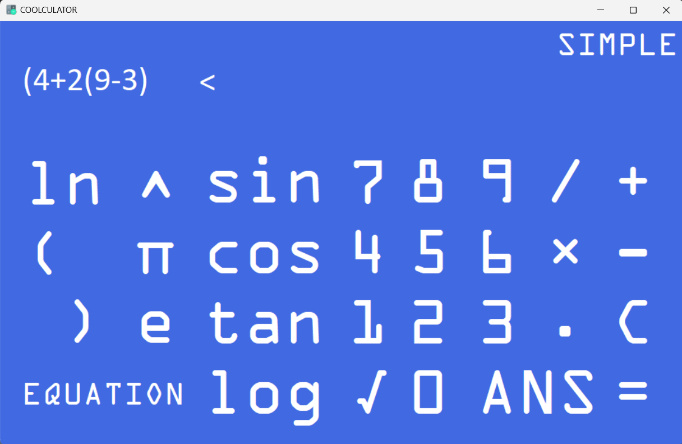
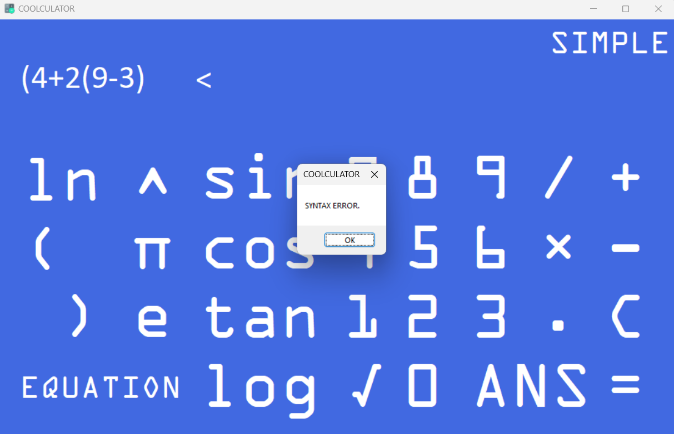
Description automatically generated

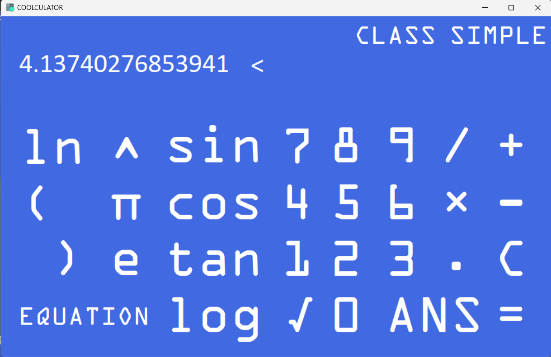
* **PASS**
* 2.3
* Entering and calculating a number with a power.
* **TYPICAL**, Expression = 675
* The correct answer “1350125107” displayed on the screen.
* Text

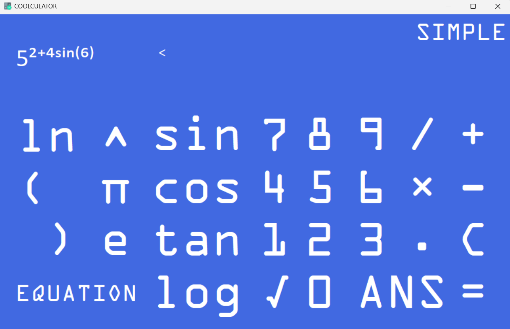
  Description automatically generatedText

  Description automatically generatedAs expected:
* **PASS**
* 2.4
* Entering and calculating an expression which uses brackets and nested brackets.
* **TYPICAL**, Expression = (4.2+8)/((2(3+6)+(sin(60))))
* The correct answer “0.6894529207…” displayed on the screen.
* Text

  Description automatically generatedText

  Description automatically generatedAs expected:
* **PASS**
* 2.5
* Attempting to calculate an expression with less close brackets than open brackets.
* **ERRONEOUS**, Expression = (4+2(9-3)
* An error message is displayed saying “SYNTAX ERROR.”
* As expected:

* **PASS**
* 2.6
* Entering and calculating an expression which has a nested expression in a power.
* **TYPICAL**, Expression = 52+4sin(6)
* The correct answer “4.137402769…” is shown on the screen.
* As expected:



* **PASS**
* 2.7
* Entering and calculating an equation with complex equations, forms and BIDMAS order.
* **TYPICAL**, Expression = 4\*5+(6.2)5-3tan(6)5\*5+4π\*ln(5)
* The correct answer “9201.58…” is shown on the screen.
* Text

  Description automatically generatedText

  Description automatically generatedAs expected:

* **PASS**
* 2.8
* Attempting to square root a negative number.
* **ERRONEOUS**, Expression = √(-10)
* An error message is displayed saying “MATH ERROR.”
* Text

  Description automatically generatedAs expected:

Text

Description automatically generated

* **PASS**
* 2.9
* Switching to the simple calculator mode.
* **TYPICAL**, Button Click = SIMPLE Button
* A picture containing text

  Description automatically generatedThe simple calculator window opens.
* As expected:

Text

Description automatically generated

* **PASS**
* 2.10
* Switching to the equation mode.
* **TYPICAL**, Button Click = EQUATION Button
* The equation mode window opens.
* Graphical user interface

  Description automatically generatedAs expected:

Text

Description automatically generated

* **PASS**

## CLASS MODE

* 3.1
* Resetting and emptying the class.
* **TYPICAL**, Button Click = CLICK FOR NEW CLASS Button
* A wait cursor appears while the text file is emptied and then uploaded to the server, and the screen shows a message saying a new class was created.
* As expected:



Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* **PASS**
* 3.2
* Opening the mode to share an answer.
* A screenshot of a phone

  Description automatically generated with medium confidence**TYPICAL**, Button Click = SHARE YOUR ANSWER Button
* The sharing mode window opens.
* Graphical user interface, text

  Description automatically generatedAs expected:
* **PASS**
* 3.3
* Opening the mode to see the class’s results.
* **TYPICAL**, Button Click = CLASS RESULTS Button, Results = 16ReedLew\_9+1=10
* The class results mode window opens.
* A picture containing text

  Description automatically generatedGraphical user interface, text

  Description automatically generatedAs expected:
* **PASS**
* 3.4
* Attempting to open the class’s results mode when there are no current results stored on the server.
* **ERRONEOUS**, Button Click = CLASS RESULTS Button, No Results
* An error message is displayed telling the user there are no results yet.
* Graphical user interface

  Description automatically generatedAs expected:
* **PASS**

## SHARING AN ANSWER MODE

* 4.1
* Attempting to enter a completely invalid username.
* **ERRONEOUS**, Username = Lewis Reed
* An error message is displayed asking the user to enter their school username.
* Graphical user interface, text, application

  Description automatically generatedAs expected:
* **PASS**
* 4.2
* Attempting to give an invalid username with letters in the wrong case.
* **BOUNDARY**, Username = 16rEEDlEW
* The same error message is displayed asking the user to re-enter their username.
* **NOT AS EXPECTED:** The answer and username are erroneously shared anyway.

**EXPLANATION:** After debugging this part of the program, I found the issue to be that an incorrect REGEX pattern was being used which didn’t check for cases of the characters: “^\d\d([A-Z]|[a-z])+$”. After replacing this pattern for the correct one which includes capital letters for the start of the names, I ran a corrective test.

* **FAIL**
* 4.2 **CORRECTIVE**
* Attempting to give an invalid username with letters in the wrong case.
* **BOUNDARY**, Username = 16rEEDlEW
* The same error message is displayed asking the user to re-enter their username.
* As expected:Graphical user interface, text, application

  Description automatically generated
* **PASS**
* 4.3
* Entering a valid username and sharing an answer.
* **TYPICAL**, Username = 22WhiteSim, Calculation = 5+5=10
* A wait cursor appears while the answer is added to the text file and the text file is uploaded. The window is closed, and the class mode window shows the message “ANSWER SHARED.”
* As expected:

Graphical user interface, text, email

Description automatically generatedGraphical user interface

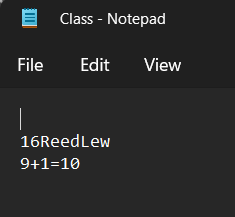
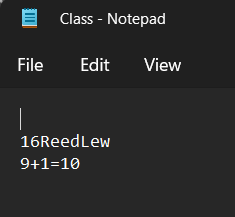
Description automatically generated

![A picture containing text

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAsAAAASCAMAAAC6q9RHAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAABLUExURf7+/gAAAKysrLa2tL68vW9vb/r79e/v7wkNDxkZG4uLi0pMQvv5+vb48fP78HZocOfh4w8AAJSUlCUiJj4HIwALAEMVNIuUiAAAAJwJVQIAAAAZdFJOU////////////////////////////////wABNAq3AAAACXBIWXMAACHVAAAh1QEEnLSdAAAAY0lEQVQYV1WO4QrAIAiET8kc9WNQY3v/N52aG3Rl93EahU3EXGwxifEnz1c7nKAq4GTbcQDHcmIfNGrogAI1Zqqjlfq8meekJ7p6Ho+k7aUypCRPu9ucr/wZ0cRNYzwu41/AC42GAhYLJyH8AAAAAElFTkSuQmCC)

Graphical user interface, text

Description automatically generated



Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* **PASS**

## VIEW CLASS’S RESULTS MODE

* 5.1
* Loading the mode and viewing a set of two results.
* **TYPICAL**, Results = 16ReedLew\_5+5=10\_4+6=10 - 22WhiteSim\_11-1+10
* The window is loaded with the first student (16ReedLew) selected.
* A picture containing text

  Description automatically generatedAs expected:
* **PASS**
* 5.2
* Selecting another student in the list of class results.
* **TYPICAL**, Index in ListBox Click = 22WhiteSim
* The second name becomes highlighted, and the contents of the label become the corresponding calculation.
* Text

  Description automatically generated with medium confidenceAs expected:
* **PASS**
* 5.3
* Viewing results with multiple calculations that extend past the screen vertically.
* **TYPICAL**, Results = 1+1=2\_2+(2)=4\_4+(4)=8\_8+(8)=16\_16+(16)=32\_32+(32)=64\_64+(64)=128\_128+(128)=256\_256+(256)=512\_512=512\_512+(512)=1024\_1024+(1024)=2048\_2048+(2048)=4096\_4096+(4096)=8192\_8192+(8192)=16384\_16384+(16384)=32768\_32768+(32768)=65536\_65536+(65536)=131072\_131072+(131072)=262144\_262144+(262144)=524288
* A scroll bar appears allowing the user to scroll through the results to view the ones displayed beyond the edge of the window.
* Text

  Description automatically generatedText

  Description automatically generatedAs expected:
* **PASS**

## EQUATION BUILDER MODE

* 6.1
* Switching to the solving an equation mode.
* **TYPICAL**, Checkbox Click = SOLVER Checkbox
* Every object to do with the graph/table mode is hidden, and every object to do with the solver mode is shown.
* Graphical user interface, text, application

  Description automatically generatedA screenshot of a computer

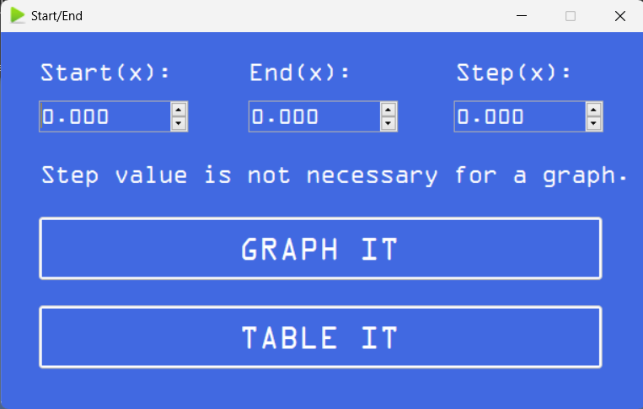
  Description automatically generated with medium confidenceAs expected:
* **PASS**
* 6.2
* Using keypresses and two button clicks to enter an expression.
* **TYPICAL**, Expression = 43sin(x)
* The expression is shown on the screen with the cursor adjusting for the power.
* Graphical user interface, application

  Description automatically generatedGraphical user interface, application

  Description automatically generatedGraphical user interface, text, application

  Description automatically generatedAs expected:
* **PASS**
* 6.3
* Attempting to proceed with the graph/table mode with a syntactically invalid equation.
* **ERRONEOUS**, Expression = 7.3++67), Button Click = GRAPH/TABLE Button
* An error message should be displayed saying “SYNTAX ERROR.”
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**
* 6.4
* Attempting to proceed with the graph/table mode with a mathematically invalid equation.
* **ERRONEOUS**, Expression = 4/0, Button Click = GRAPH/TABLE Button
* An error message should be displayed saying “MATH ERROR.”
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**
* 6.5
* Proceeding with the graph/table mode with a valid equation.
* **TYPICAL**, Expression = 43sin(x), Button Click = GRAPH/TABLE Button
* The window to enter the user’s start/end/step values is shown.
* As expected:

Graphical user interface, application

Description automatically generated

* **PASS**
* 6.6
* Attempting to solve the equation with no input.
* **ERRONEOUS**, Empty Equation
* An error message is displayed saying “CANNOT SOLVE.”
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**
* 6.7
* Attempting to solve an equation without an x value.
* **ERRONEOUS**, Equation = 45=7
* An error message is displayed saying “NO SOLUTION FOUND.”
* As expected:
* **PASS**
* 6.8
* Attempting to solve a quadratic equation with no solutions.
* **ERRONEOUS**, Equation = 5x2+3x+6=0
* An error message is displayed saying “NO SOLUTION FOUND.”
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**
* 6.9
* Attempting to solve a mathematically invalid equation.
* **ERRONEOUS**, Equation = 78/0=4x
* An error message is displayed saying “CANNOT SOLVE.”
* A screenshot of a computer

  Description automatically generated with medium confidenceAs expected:
* **PASS**
* 6.10
* Attempting to solve a trigonometry equation with no solutions (sin(x) equals a value >1).
* **ERRONEOUS**, Equation = sin(x)=5
* An error message is displayed saying “NO SOLUTIONS FOUND.”
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**
* 6.11
* Solving a valid equation involving trigonometry (sin(x) equals a positive value <1).
* **TYPICAL**, Equation = sin(x)=0.7
* The correct answer “0.775397…” is displayed in a message box.
* Graphical user interface, application

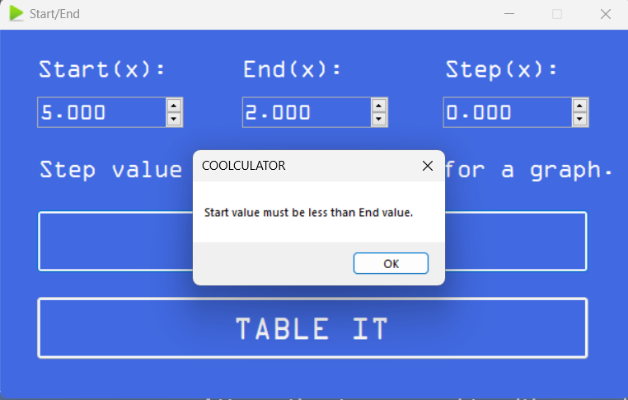
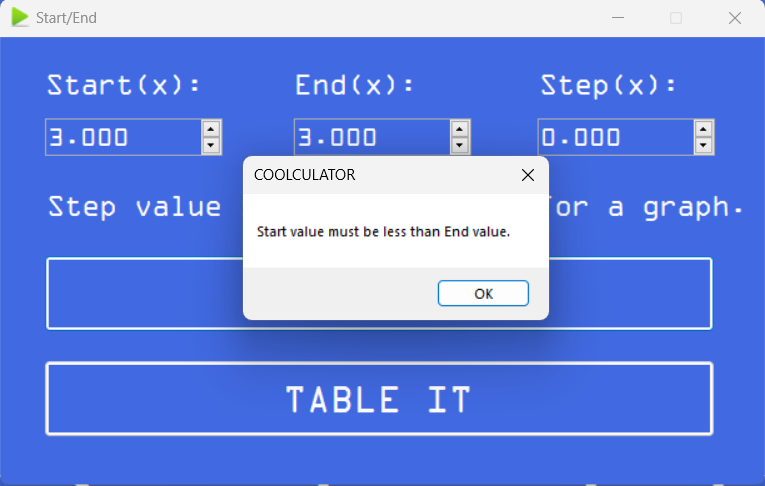
  Description automatically generatedAs expected:
* **PASS**
* 6.12
* Solving a valid equation involving trigonometry (sin(x) equals the maximum value; 1).
* **BOUNDARY**, Equation = sin(x)=1
* The correct answer (pi/2) “1.570729…” is displayed in a message box.
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**
* 6.13
* Solving an equation involving x with a power (quadratic).
* **TYPICAL**, Equation = 9x2+15x+3=0
* The correct answer “-0.232408…” is displayed in a message box.
* Graphical user interface, application

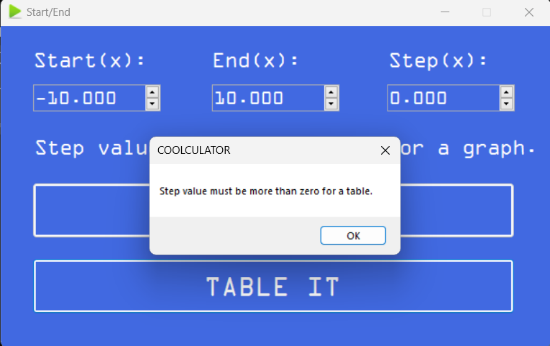
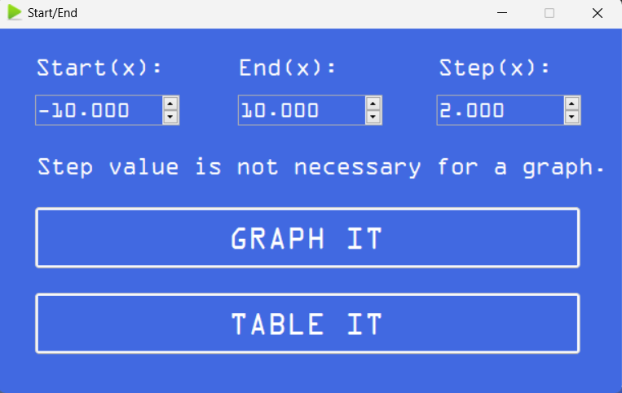
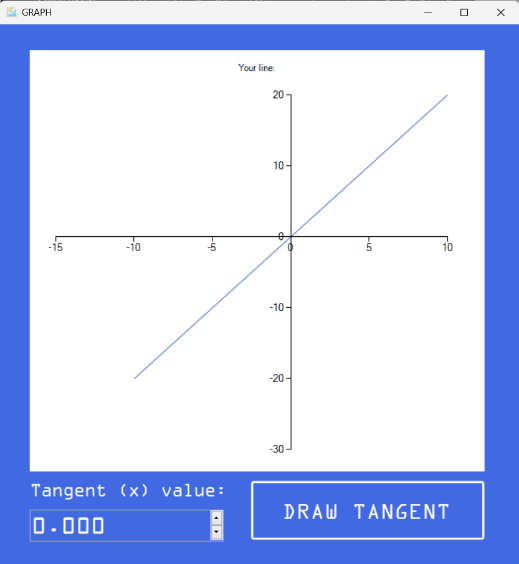
  Description automatically generatedAs expected:
* **PASS**
* 6.14
* Solving an equation with complex left and right hand sides, including the x located in a power.
* **TYPICAL**, Equation = (4.2+3)3\*6=74.5-6x
* The correct answer “0.089…” is displayed in a message box.
* Graphical user interface, application

  Description automatically generatedAs expected:
* **PASS**

## START/END/STEP ENTERING MODE

* 7.1
* Attempting to proceed to either mode with the start value greater than the end value.
* **ERRONEOUS**, Start(x) = 5, End(x) = 2
* An error message is displayed telling the user that the end value must be greater than the start value.
* As expected:
* **PASS**
* 7.2
* Attempting to proceed to either mode with the start value equal to the end value.
* **BOUNDARY**, Start(x) = 3, End(x) = 3
* An error message is displayed telling the user that the end value must be greater than the start value.
* As expected:
* **PASS**
* 7.3
* Attempting to proceed to the table mode with the step value as zero.
* **ERRONEOUS**, Step(x) = 0
* An error message is displayed telling the user that the step value must be more than zero for a table.
* **NOT AS EXPECTED:** The program tries to erroneously build the table with a step value of zero and crashes.

**EXPLANATION:** After debugging this part of the program, I found the issue to be that the program would assume the user had put an appropriate step value if they chose the table option. However, since this causes a complete program break, I realised this issue needed to be addressed. After editing the code so that the program would run a check on the validity of the step value input, I ran a corrective test.

* **FAIL**
* 7.3 **CORRECTIVE**
* Attempting to proceed to the table mode with the step value as zero.
* **ERRONEOUS**, Step(x) = 0
* An error message is displayed telling the user that the step value must be more than zero for a table.
* As expected:
* **PASS**
* 7.4
* Proceeding to either mode with valid input.
* **TYPICAL**, Start(x) = -10, End(x) = 10, Step(x) = 2
* The table mode should open and create a table using all the data inputted, while the graph mode should open and draw a graph using the data inputted but ignoring the step value.
* As expected:
* **PASS**

## GRAPHING MODE

* 8.1
* Drawing a simple y=mx+c graph.
* **TYPICAL**, Equation = f(x)=2x+4, Start(x) = -5, End(x) = 5
* The correct line with gradient 2 and y-intercept 4 should be drawn on the graph.
* Chart, scatter chart

  Description automatically generatedAs expected:
* **PASS**
* 8.2
* Drawing a quadratic graph.
* **TYPICAL**, Equation = f(x)=2x2+5x+7, Start(x) = -10, End(x) = 10
* The correct graph with no solutions should be drawn on the graph
* Chart

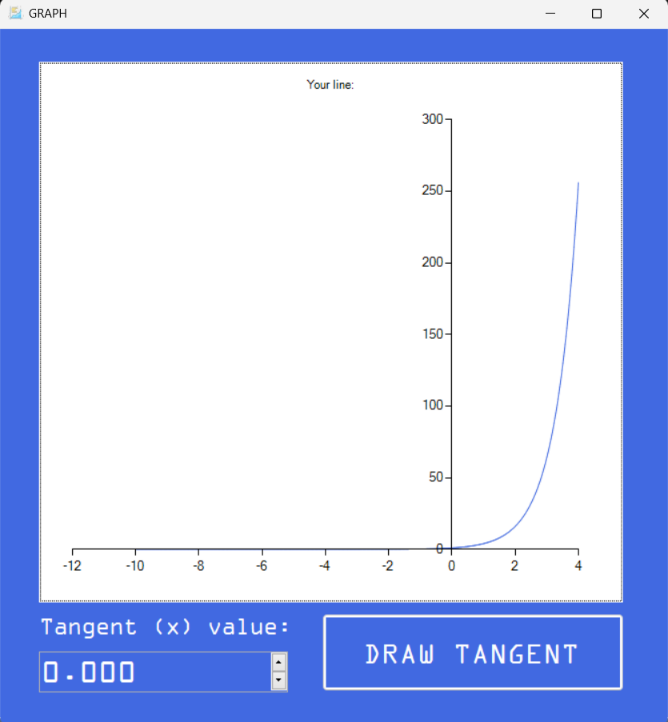
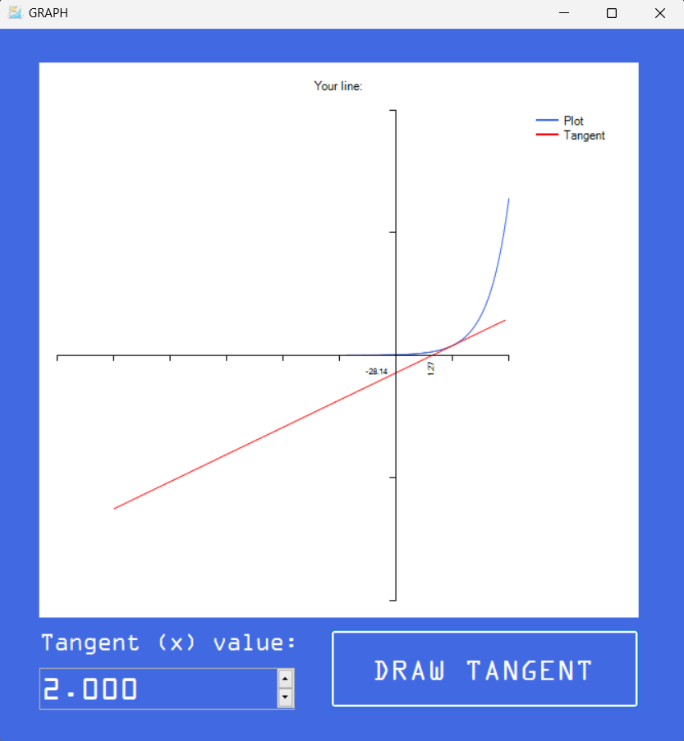
  Description automatically generatedAs expected:
* **PASS**
* 8.3
* Drawing a tangent on the quadratic graph.
* **TYPICAL**, Tangent(x) = -7
* A straight line tangent only touching the graph at x=-7 should be drawn, as well as its intercepts with the axis being labelled.
* Chart, line chart

  Description automatically generatedAs expected:
* **PASS**
* 8.4
* Drawing a trigonometric graph.
* **TYPICAL**, Equation = f(x)=sin(x), Start(x) = -10, End(x) = 10
* The correct line - a sine wave - is drawn on the graph.
* As expected:

Graphical user interface

Description automatically generated

* **PASS**
* 8.5
* Drawing a tangent to the trigonometric graph.
* **TYPICAL**, Tangent(x) = 3.15
* A straight line tangent only touching the graph at x=3.15 should be drawn, as well as its intercepts with the axis being labelled.
* Chart, line chart

  Description automatically generatedAs expected:
* **PASS**
* 8.6
* Drawing an exponential graph
* **TYPICAL**, Equation = f(x)=4x, Start(x) = -10, End(x) = 4
* The correct exponential line is drawn.
* As expected:
* **PASS**
* 8.7
* Drawing a tangent to the exponential graph.
* **TYPICAL**, Tangent(x) = 2
* A straight line tangent only touching the graph at x=2 should be drawn, as well as its intercepts with the axis being labelled.
* As expected:
* **PASS**
* 8.8
* Drawing a reciprocal graph.
* **TYPICAL**, Equation = f(x)=1/x, Start(x) = -3, End(x) = 3
* The correct reciprocal lines are drawn, with the graph starting a new series when the gradient becomes unreasonably high.
* A picture containing diagram

  Description automatically generatedAs expected:
* **PASS**
* 8.9
* Drawing a graph using an equation with complex forms and BIDMAS order.
* **TYPICAL**, Equation = f(x)=(4x+3)4x+3\*sin(6x)/3, Start(x) = -0.75, End(x) = 0
* The correct line is drawn on the graph.
* Diagram

  Description automatically generatedAs expected:
* **PASS**
* 8.10
* Showing that the option to draw a tangent does not appear when the Maxima software is unable to differentiate the equation.
* **TYPICAL**, Equation = f(x)=(4x+3)4x+3\*sin(6x)/3, Start(x) = -0.75, End(x) = 0
* The graph is drawn without any objects related to tangent drawing shown as the gradient cannot be calculated.
* Diagram

  Description automatically generatedAs expected:
* **PASS**

## TABLE BUILDING MODE

* 9.1
* Building a table using a simple equation.
* **TYPICAL**, Equation = f(x)=5x+3.5, Start(x) = -5, End(x) = 5, Step(x) = 1
* The correct x values are shown in a ListBox and the corresponding f(x) value when an x value is clicked.
* Graphical user interface

  Description automatically generatedGraphical user interface

  Description automatically generatedAs expected:
* **PASS**
* 9.2
* Building a table using an equation with complex forms and BIDMAS order.
* **TYPICAL**, Equation = f(x)=(4x+3)4x+3\*sin(6x)/3, Start(x) = -0.75, End(x) = 0, Step(0.075)
* The correct x values are shown in a ListBox and the corresponding f(x) value when an x value is clicked.
* Graphical user interface

  Description automatically generatedGraphical user interface

  Description automatically generatedAs expected:
* **PASS**
* 9.3
* Building a table where the input results in a mathematical error in a row of the table.
* **ERRONEOUS**, Equation = f(x)=5/x, Start(x) = -5, End(x) = 5, Step(x) = 1, Row Where x = 0
* Every row resulting in a valid answer is displayed, but the f(x) value for x = 0 shows “NaN” meaning Not a Number.
* Graphical user interface

  Description automatically generatedGraphical user interface, text

  Description automatically generated with medium confidenceAs expected:
* **PASS**
* 9.4
* Building a table with the maximum number of rows using the start/end/step inputs.
* **BOUNDARY**, Equation = f(x)=5x+2, Start(x) = -100, End(x) = 100, Step(x) = 0.01
* The correct x values are shown in a ListBox and the corresponding f(x) value when an x value is clicked.
* Graphical user interface

  Description automatically generated with medium confidenceA picture containing graphical user interface

  Description automatically generatedAs expected:
* **PASS**
* 9.5
* Sorting the f(x) values of a polynomial table in ascending order.
* **TYPICAL**, Equation = f(x)=2x2+3x+1, Start(x) = -10, End(x) = 10, Step(x) = 0.5, Sort = ↓
* The x values are reordered so their corresponding f(x) values are in ascending order.
* Graphical user interface

  Description automatically generated with medium confidenceAs expected:

A picture containing graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

* **PASS**
* 9.6
* Sorting the f(x) values of a polynomial table in descending order.
* **TYPICAL**, Equation = f(x)=2x2+3x+1, Start(x) = -10, End(x) = 10, Step(x) = 0.5, Sort = ↑
* The x values are reordered so their corresponding f(x) values are in descending order.
* A picture containing chart

  Description automatically generatedGraphical user interface

  Description automatically generatedAs expected:

A picture containing chart

Description automatically generated

* **PASS**
* 9.7
* Sorting the f(x) values of a polynomial table in the original start/end/step order.
* **TYPICAL**, Equation = f(x)=2x2+3x+1, Start(x) = -10, End(x) = 10, Step(x) = 0.5, Sort = ↑
* The x values are reordered so their corresponding f(x) values are in the original start/end/step order.
* A picture containing graphical user interface

  Description automatically generatedA picture containing chart

  Description automatically generated As expected:
* **PASS**
* 9.8
* Searching for an f(x) value (0) to locate the corresponding x value (solution to the quadratic).
* **TYPICAL**, Equation = f(x)=2x2+3x+1, Start(x) = -10, End(x) = 10, Step(x) = 0.5, Search Value = 0
* The x value with a corresponding f(x) equalling 0 is selected.
* A picture containing chart

  Description automatically generatedAs expected:
* **PASS**

# EVALUATION

## COMPARISON OF PROJECT AGAINTS REQUIREMENTS

1. Once the COOLCULATOR loads up, it must go straight to the COOLCULATOR screen where calculations can be made, rather than a menu or something similar, as to save the user hassle and time in a lesson. They will be presented with:

**MET – The first screen shown to the user is the simple calculator mode, great for when a quick calculation is needed. Also, the scientific calculator, used for more complex calculations in a maths lesson, is only one click away.**

**POSSIBLE IMPROVEMENT – If the user prefers to see the scientific calculator, as maybe they are more familiar with complex maths functions than others, the user could choose to see this first.**

* Numerical buttons showing simply with the number in a white, calculator-style font and on a consistent COOLCULATOR-blue background. When they are clicked, the value of the button is involved in the calculation.

**MET – The theme of the numerical buttons is clean and simple. When each one is clicked, their value is added to the contents of the screen of the current calculator mode.**

* Operator buttons with formatting consistent with numerical buttons. However, when any of these buttons are clicked, their corresponding operator will be called for the current calculation.

**MET – The theme of the operator buttons is consistent with the numerical buttons. When each one is clicked, their designated character is added to the contents of the screen of the current calculator mode.**

* A CLASS button will show once a calculation has been completed. This will be clicked when the user wants to upload or view other’s answers.

**MET – Every time the user completes a calculation, they have the option to go to the class mode. The button is hidden when it is not being used in order to minimise scree clutter.**

**POSSIBLE IMPROVEMENT – The option to share answers with the class could be implemented for the simple calculator mode too, for example in use of primary school maths classes.**

* More buttons for scientific functions with consistent format. These buttons will include:
  + Indices – The user will click this button after typing in a value, then type a number, in superscript, to be the power of that value.

**MET – The user is given a button with the ^ character, as this character is typically used for indices notation. Once this button is pressed, the cursor is moved up and made smaller to indicate the user is now inputting characters and functions in superscript.**

* + Trigonometric functions (sin, cos and tan) – The user will type a value after clicking this button. They will end the function by inserting a close bracket after their value.

**MET – sin, cos and tan buttons are shown and “sin(“, “cos(“ and “tan(“ are added to the contents of the screen respectively. This allows the user to neatly type in their value for the function and end it with a bracket.**

* + Constants – These will be used in the same way as the numerical buttons, where the value will be of the constant clicked.

**MET – The user is given pi and e buttons to use in their calculations, with the buttons working the same as usual number ones.**

* + Logarithmic functions – This will work similarly to the trigonometric functions, however the user will write a value when they want to find the result of its log base-10 or natural log.

**MET – log and ln buttons are shown, and “log(“ and “ln(“ are added to the contents of the screen respectively. The log function returns the log base-10 result of the user’s value, where the ln function returns the natural log (log base-e) of the user’s value.**

1. Also, the COOLCULATOR will display an extra ‘equation’ button to access more complex modes. The next screen will consist of modes which will allow users to:

**MET – At any time, from the scientific calculator mode, the user has the option to click equation button and enter the equation building mode.**

**POSSIBLE IMPROVEMENT – The button could also be shown in the simple calculator mode, giving the user the option to access the equation mode. However, this need would be a rare occurrence as the user will most likely be on the scientific mode already before requiring the equation mode.**

* Make tables – This involves the user giving an equation, start, end and step, then the COOLCULATOR displays a table showing the index and the corresponding value in an appropriate format.

**MET – The COOLCULATOR is able to make large tables with complex equations. It handles errors well and displays the tables in a consistent, clean theme. The tables can also be sorted very efficiently using the quick sort, as well as having an f(x) value searched for very quickly using a binary search.**

* Graph sketcher – The same input to the table maker will be implemented here, but instead of displaying the data in the form of a table with rows and columns, the data will be shown across a graph.

**MET – The COOLCULATOR is able to draw ascetically pleasing graphs using a variety of complex equations, giving it the ability to create the recognisable curves from A-level maths, as well as unusual arbitrary shapes, correctly. It uses a consistent theme and consistent colours to the rest of the program. In addition to this, where the equation can be differentiated, a tangent can be drawn at any x point, where the gradient is calculated at that point and the y-y1=m(x-x1) equation is used to build another tangent series on the graph.**

* Solve algebraic equations – The user will be able to enter an expression involving an ‘x’ and equal it to 0, and then solve the equation, finding a value for x.

**MET – The user can even solve an equation even if it doesn’t equal zero. Instead, the user types in a left-hand side and right-hand side to an equation. The program simply takes the RHS away from the LHS and equals it to zero. Then the Newton-Raphson method is used to solve for x.**

1. There should be a friendly user-interface. The font needs to be ascetically pleasing while being easy to read. The calculator should also have multiple input methods, making it as efficient as possible for the user to enter their maths problems:

**MET – The calculator does indeed have multiple input methods. It also has a friendly-looking, calculator-style font used for on screen objects. However, a more familiar typing font is used for the input screen to differentiate it from buttons.**

* Using on-screen buttons – All functions will be on-screen, meaning the user can call any function they need by using touchscreen and/or their cursor to click the appropriate buttons.

**MET – Every mode and function can be found on the screen of the COOLCULATOR, meaning the user does not need to use a keyboard at all if they would prefer.**

* Keyboard input – Some functions will be able to be typed in using certain keys on a keyboard, if an appropriate key can be allocated to that function, such as:

**MET – However, some inputs may be easier for the user to input via their keyboard.**

**POSSIBLE IMPROVEMENT – Even though this was not a requirement, it may have been even easier if the user could input trigonometric and logarithmic functions using their keyboard, like the ‘s’ key displaying ‘sin(‘, although the ‘c’ character is already used and the ‘l’ key could mean ‘log(‘ or ‘ln(‘.**

* Number row – All numbers and decimal points can be inputted from the keyboard by pressing the corresponding number key and the ‘full stop’ key respectively.

**MET – All number keys work in adding their corresponding numerical value to the contents of the current calculator screen, as well as decimal points.**

* ‘A’ or ‘a’ input – Either of these keys will call the answer function which will make the next numerical value equal to the last answer.

**MET – The previous answer, surrounded by an open and close bracket, is added to the contents of the screen when an ‘a’ key is pressed.**

* ‘C’ or ‘c’ input – Either of these keys will call the clear function, which will clear the whole calculation entered.

**MET – The contents of the screen, and the calculation to be shared with the class, are emptied when a ‘c’ key is pressed.**

* ‘+’, ‘-‘, ‘\*’ or ‘x’ or ‘X’, ‘/’, ‘=’ or Enter key, in – Any of these keys will call their corresponding operator for the current calculation.

**MET – Each operator key will add its corresponding character to the contents of the screen.**

* ‘^’ – This key can allow the user to enter the superscript mode, allowing them to type the power to their numerical values.

**MET - The ‘^’ key will enter the user into superscript typing mode, shown by a cursor on the screen, and allow the user to type the power to a number.**

1. The user must be able to upload their answers to an online format with consistent fonts and colours, and share their answers with the class. This will be achieved once the CLASS button is pressed. They should have three options:

**MET – Whenever a calculation has been completed, the CLASS button shows, giving the user the option to share the answer they just achieved, or view others’.**

* Upload – Be able to enter their names, along with their inputted calculation and outputted answer.

**MET – The option to enter their school username appears in a simple sharing mode, where they can upload their calculation to the class on the server.**

* Download – Retrieve the class’s results straight from online and display the answers with their corresponding student’s names in an easy-to-read table or list format.

**MET – The user also the option to review the class results, downloading the class file and reading it locally.**

* Create a new class – The user should be able to clear the results and create a new blank class where students can input their results.

**MET – Finally, the user can empty the class file on the server, refreshing and starting a new class.**

1. For any extra modes or tasks which take more room on the screen, the program should open new forms over the COOLCULATOR to keep the formatting neat, avoiding overlapping text or taking too much room on the screen. This will apply to any of the extra, complex modes in addition to the CLASS screen used to upload or view other’s answers.

**MET – Each mode has a dedicated window, with no-longer-necessary, previously open windows closing behind accordingly. This keeps the screen clean, as well as avoiding cramped or over-lapping content on each window.**

1. Error messages must be showed, describing the error at hand. This includes “Math Error” as seen on the original Casio calculator, which occurs when the calculation is not mathematically possible (e.g. √(-2) = Math Error).

**MET – When an input is syntactically invalid, the user is given a message saying, “SYNTAX ERROR.”, while when an input is mathematically invalid, they are given a message saying “MATH ERROR.”. Also, around the whole program, errors are caught and informed to the user with according messages, like in the solving equation mode with either “CANNOT SOLVE.” or “NO SOLUTION FOUND.”**

1. Buttons and user interface must be kept as simple and legible as possible. This includes only indicating ‘sin’ rather than ‘sin(‘ or ‘√’ rather than ‘√(‘.

**MET – Similar to the Casio calculator, the function buttons are displayed as simply the function (e.g. ‘log’, ‘cos’), where when clicked, what is actually inputted is ‘log(‘ and ‘cos(‘, as to make it clearer for where the user should type the input to their function, while keeping screen objects clean and looking less cluttered.**

1. As the COOLCULATOR will be used on another device’s screen, the utility of the limited space should be maximised. This means the COOLCULATOR and all its external modes will have as much room of the window taken up as possible while still being user-friendly. This also means formatting needs to be extra neat and minimalistic, like not having a ‘box’ for the COOLCULATOR and stretching it out to the edges of the window itself.

**MET – Every window has its contents stretching to the edges, with their sizes not being unnecessarily large. The COOLCULATOR uses only 2 colours which are bright and distinctive, meaning it is excellent at being differentiated from other potential windows the user has on their screen. In addition to this, boxes around content are not used in the program unless necessary, creating a fluid design within the windows.**

## Graphical user interface, text Description automatically generatedUSER FEEDBACK

## FINAL ADDITIONAL REQUIREMENT FOR POSSIBLE IMPROVEMENT

Based on the criticisms mentioned, here are some additional requirements that could enhance the calculator program:

1. Output Formatting Options: The program should include a few output formatting options to allow the user to display the output in different formats, such as scientific notation, fraction display mode, or comma-separated numbers. This would make the output more visually appealing and easier to interpret.
2. Memory and History Function: The program should include a memory and history function that allows users to save and recall previous calculations. This feature would be especially helpful for users who need to perform repetitive calculations or who want to compare the results of different calculations.
3. Customizable Interface: The program should allow users to customize the interface to their preferences. Users could choose different colour schemes, font sizes, or even the layout of the buttons. This would make the program more user-friendly and personalized.
4. Accessibility Options: The program should include accessibility options such as text-to-speech or high contrast mode. This would allow users with visual impairments or other disabilities to use the program more easily.

By incorporating these additional requirements, the calculator program would become even more versatile, user-friendly, and accessible to a wider range of users.

1. Price taken from <https://www.casio.co.uk/catalog/product/view/id/1047/s/fx-991ex/category/154/> on 21/09/2022. [↑](#footnote-ref-2)
2. <https://education.casio.co.uk/download-emulator-cw> [↑](#footnote-ref-3)