Graphing Calculator Design Document

Authors (Group #8):

Kashaan Ali – 0815445

Radwa El Farmawy – 0863703

Yun Hao – 0840924

Dominic Lee – 0823091

Michael Oakes – 0798233

Matthew Weigel – 0833519

Table of Contents:

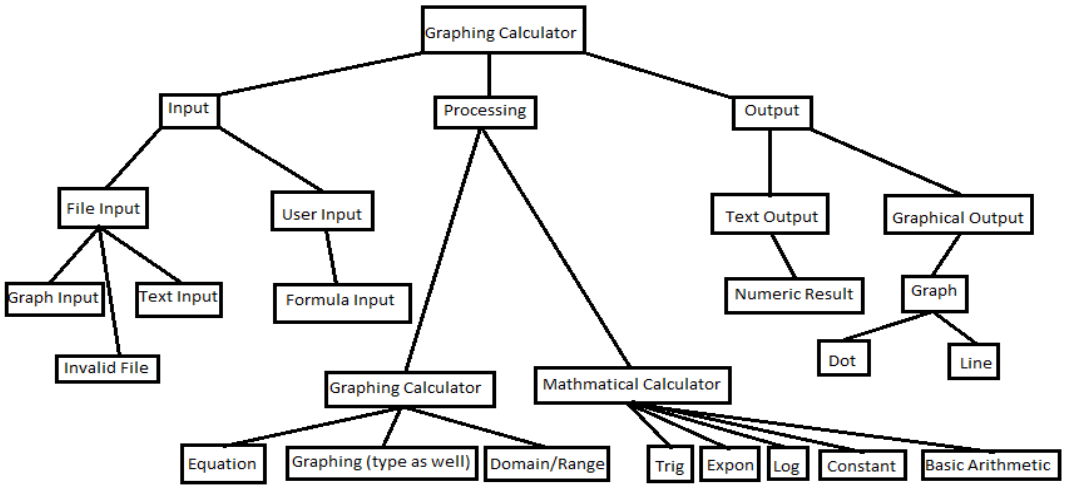
1.0 Introduction

1.1 Purpose

The purpose of this design document is to create a scientific calculator using top-down design in Python. It will perform the following:

- Ask for user input.
- Calculates input.
- Outputs the answer to a calculation or a graph function.

1.1 Architecture Diagram:



2.0 Data Structures

2.1 Global Data Structures

Integers are used to store the constants of cos, sin, tan, and PI. The user input is stored in the calculator_display global. In this global variable everything will be calculated inside and outputted afterwards.

Example:

5**2 = 25 5^2=25 is the user input will be changed into 5**2 = 25, sin,cos,tan will be changed to math.sin, math.cos, math.tan.

Perhaps make a table or chart of all the math algorithms we will be transforming like listed previously.

The user input is stored into the global variable calculator_display. This is used for the memory functions, trigonometric functions, and basic calculations like the example above.

The list "coords" is used to store the plot points for the graphing portion of the calculator. Then all the data inside the list is transferred throughout the program from one function to another.

Graphing:

- Entering a function and clicking “Graph” will let the program know that it is a graphing function that needs to be graphed.
- The equation of the graph such as y = mx + b
- The domain of the graph in the form: minimum < x < maximum.
- The coordinates and scale for the graph.

Example of a list that will be used in the program:

Graph:

y = mx+b minimum x maximum x “coords” scale

[“y = 1x+2 ”, “2”, “102”, “0,2”, “100-100”]

2.2 Internal Data Structures

Each button has its own function which inside includes a variable. The variable along with the function will be linked to TK in order for the button to be clickable.

Example of an operator button (divide):

<variable button name> = <Tk variable name> (properties of button [calculator_model.add_to_display('/')])

Memory has its own functions that do the following:

- Puts a value from the calculator into memory.
- Adds a value from the calculator with the existing value in memory.
- Subtracts a value from the calculator with the existing value in memory
- Displays the memory on window.
- Adds the value in memory to the variable.
- Clears the memory.

2.3 Temporary Data Structures

The filename of the inputted file is stored in a variable called calculate_file which has the parameters read_file and write_file. The read_file is the file directory for the input text file and the write_file is the file directory for the program to output to. These variables will be used to find the file path, allowing the program to open and read the information as well as create and write new information. This variable will be used to find the file path allowing the program to open and read the information.

3.0 Functions and Algorithms

This section of the program will briefly explain as to what options the user can choose from. Also, how this will be done in order to complete the process of entering a calculation with receiving an appropriate output.

3.1 User input from text file

3.1.1.1 Interface Description

init()

- Accepts user input through the GUI or via text file.
- Input into GUI is done manually by the user.
- Input via text file is done

3.1.1.2 Algorithm Model

1. Read input from input file or from GUI input by user.
2. Based on which ever option the user chooses for input either calculator_console_view.init() or calculator_gui_view.init() will be run

A error check method is used to check the input/output numbers can not exceed maximum value for type double and the equation must be correctly inputted in designated format or else prompt user to fix input mistake.

3.1.1.3 Restrictions

None

3.1.1.4 Local Data structures

None

3.1.1.5 Performance issues

None

3.1.1.6 Design Constraints

- The calculator must be command line oriented.

3.1.2.1 Interface Description

add_to_display()

- This function will perform all the necessary calculations for graphing and/or simple and complex calculations.
- This function will also check if the input uses python operators if not it will let the user know that they should input python operators.

3.1.3.1 Interface Description

calculate_file(read_filename, write_filename)
-read_filename is the stored name for the input file to read the equation and/or graph for processing.
-write_filename is the stored name for the output file to write the answers to the input equation.
- This function reads and writes to the given file by the user.
- If a write file is provided then write the result to a file otherwise print it out to the console.
- The function also checks to see whether the user has provided a text file or not. If not then print out an appropriate message saying that they have not included a file.

3.1.4.1 Interface description

calculator_console_view()

- This function will allow the user to enter any type of calculations and/or graph through text file mode.
- This function will branch onto calculator_model() and pass all the parameters to calculate any type of calculation.

3.1.4.2 Algorithm Model

1. Once the user begins running the program through console mode they will be allowed to graph and/or create any type of calculations of their choice.
2. Theconsole mode will open when the calculator_model() and calculator_display function is called and then all the other functions that are required for the GUI mode will also be called as explained above.

3.1.4.3 Restrictions

None

3.1.4.4 Local Data structures

None

3.1.4.5 Performance issues

None

3.1.4.6 Design Constraints

- The calculator must be command line oriented.

3.1.5.1 Interface description

calculate_file(read_filename, write_filename)

-read_filename is the stored name for the input file to read the equation and/or graph for processing.
-write_filename is the stored name for the output file to write the answers to the input equation.
- This function not only reads and writes to file but also calculates any calculation given by the user.

- The function will pass the equation to the calculate() and then receive the output as a return value as a.
- This will print out the output in the text file and on the command line.

3.1.5.2 Algorithm Model

- Receiving the answer will mean that the user can exit the program by clicking ‘x’ on the right hand side if they run the GUI model and if file model will mean that after receiving the answer it is safe to assume that the program has stopped unless they enter in a new equation.

3.1.5.3 Restrictions

None

3.1.5.4 Local Data structures

None

3.1.5.5 Performance issues

None

3.1.5.6 Design Constraints

- The calculator must be command line oriented.

3.2 GUI

The calculator will be presented on one GUI screen that will display both basic calculation options (ex. sin (30)) along with a graphing screen above the input box for the calculations. The scientific calculator window resizes depending on the graph function that the user inputs. So when the calculator is resized it does not displace the buttons. Pressing the close button on the either calculators GUI will terminate the program. Also, the max x-axis range will be set to a constant of 300 just so it does not displace the buttons on the screen.

3.2.1 GUI Components Algorithms

update_calculator(*args)

The variable args is the user's input file name for processing by the program.
As stated above, this function will make sure that the calculator_model is in sync with the data from the entry boxes to make sure they match. This is like the controller updating the model and the GUI view to be in synch.

draw_graph()

1. Draws the graph on the canvas.
2. Displays it afterwards on the GUI screen.

Buttons

init_calculator_widgets()

The buttons on the calculator will be named just like any other calculator with the appropriate names such as PI, sin, cos, tan, etc.

add_to_display()

This function simply adds the digits 0-9 and symbols accordingly onto the calculator. Buttons will be made using TK.

add_trig_display()

The function above will add all the appropriate trigonometric functions into their designated spot calling the variable trig_functions as a parameter.

Example of a button using TK

<button command name> = <Tk.Button(properties of the button: [function linking to the button command])>

Properties of button: such as the frame, text, command, and colour.

The above will apply to all the buttons.

Example Button

<button command name>():
<command code>
<button name> = <button creation and button command name>

GUI Restrictions

None

GUI Limitations

None

GUI Local Data structures

- Button names and button command functions.

GUI Performance issues

None

GUI Design Constraints

None

3.2.2 Scientific Calculator GUI Description

A window will pop up with a grid for graphing and clickable buttons containing symbols and digits underneath the grid.

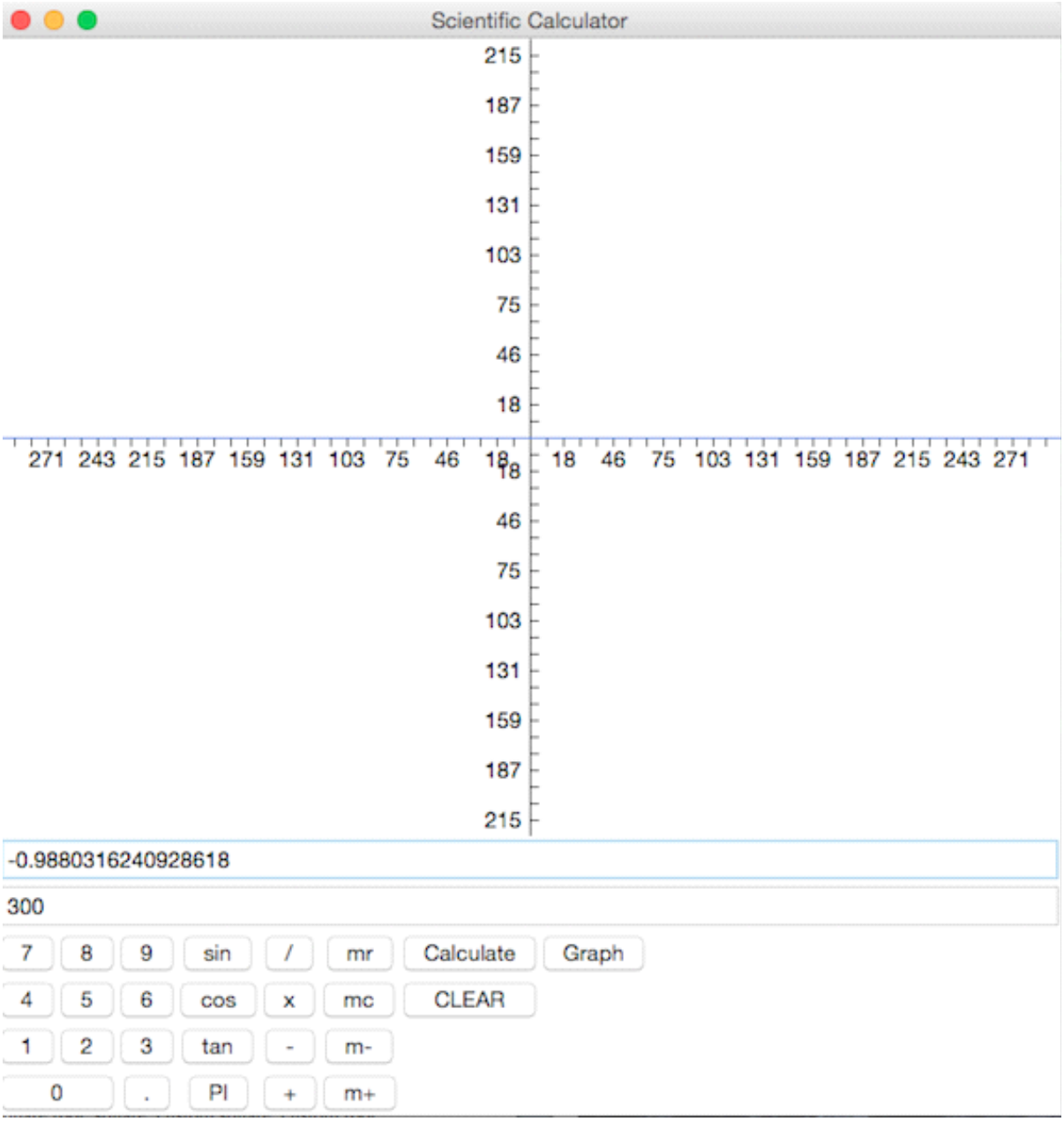
The buttons are arranged in the formation of a physical calculator.
Calculator Symbols and Numbers:

1) Advance Arithmetic (x^2, x^y, $\sqrt{x}$, $\sqrt[y]{x}$, sin, cos, tan)
2) Basic Arithmetic (×, ÷, −, +, =)
3) Constants (e, PI)
4) Numbers (0-9)
5) Positive/Negative Button (+/−)
6) Brackets

7) Memory Button (M+, MC, MR, M-)
8) Clear Button that clears the entire input

The user will be able to input any function as they wish which will in return output the answer they are looking for. Two boxes will be shown under the graph grid and above the clickable buttons. The first box is for the user input and the second box displays the maximum x-axis range, which is 300. The user will simply need to press the ‘calculate’ button in order to receive an answer.

Below is a screenshot of what the calculator GUI may look like in terms of simple calculating (sin 30):



3.2.2.1 Scientific Calculator GUI Algorithms

calculator_gui_view.init()

In this function the user will be able to run the calculator along with view the calculator with all the clickable buttons. Once the user clicks the ‘calculate’ button they will immediately receive the answer on the same screen as they entered their calculation. The calculation will be sent to the function calculate() which will return the answer to the their question. Everything will be done on one screen.

3.2.2.2 Scientific Calculator GUI Restrictions

- The buttons presented on the screen can only be used nothing else.

3.2.2.3 Scientific Calculator GUI Limitations

None

3.2.2.4 Scientific Calculator GUI Local Data structures

- Button names and button command functions.

3.2.2.5 Scientific Calculator GUI Performance issues

None

3.2.2.6 Scientific Calculator GUI Design Constraints

- The GUI screen will be in a certain position.

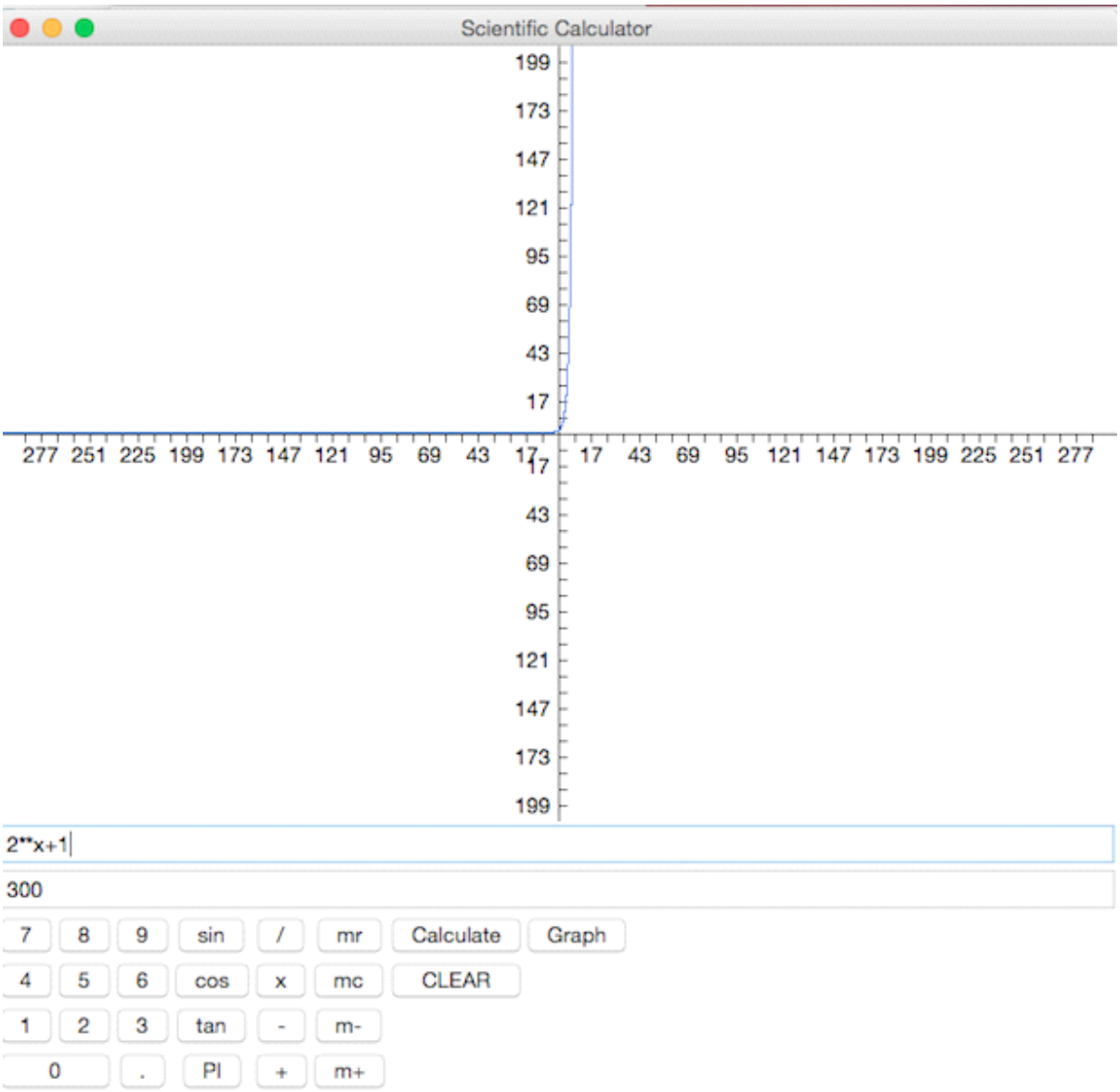
Graphing GUI Description

3.2.3.1 Graphing GUI Algorithms

init_graphing_widgets()

The function above runs the graphing portion of the calculator along with drawing the canvas. This will ask the user for input to draw any type of graph. This will be then sent to calculator_model where all the commands will be followed.

Below is a screenshot of what the calculator GUI may look like in terms of graphing and plotting:



3.2.3.2 Graphing Calculator GUI Restrictions

- Drawing the graph should not adjust the size of the calculator.

3.2.3.3 Graphing Calculator GUI Limitations

None

3.2.3.4 Graphing Calculator GUI Local Data structures

- The equation textbox prints out the information from equation variable.

3.2.3.5 Graphing Calculator GUI Performance issues

None

3.2.3.6 Graphing Calculator GUI Design Constraints

None

3.2.4.1 Interface description

calculator_gui_view.init()

- This function will allow the user to enter any type of calculations and/or graph through GUI mode.
- This function will branch onto Tk.mainloop() and begin the GUI program. From there all the functions inside init(); will begin to play their role such as drawing the graph and making calculations according to the user input.
- This will then call functions such as calculate(), add_to_display(), add_trig_display(), memory_add(), memory_subtract(), memory_clear(), memory_recall(), and clear_display().

3.2.4.2 Algorithm Model

1. Once the user begins running the program through GUI mode they will be allowed to graph and/or create any type of calculations of their choice.
2. The GUI mode will open when the Tk.mainloop() function is called and then all the other functions that are required for the GUI mode will also be called as explained above.

3.2.4.3 Restrictions

None

3.2.4.4 Local Data structures

None

3.2.4.5 Performance issues

None

3.2.4.6 Design Constraints

- The calculator must be command line oriented.

3.3 Memory

Memory Description
Depending on the memory command (refer to 2.2)

Memory Algorithms
3.3.1 memory_add()

a) Evaluates equation or value.
b) Adds it to the current memory value.

3.3.2 memory_substract()

a) Evaluates equation or value
b) Subtracts it to the current memory value.

3.3.3 memory_clear()

a) Sets the memory to zero.

3.3.4 memory_recall()

b) Checks to see what the last equation or value that is entered in the text box.

3.3.*.1 Memory Restrictions

None

3.3.*.2 Memory Local Data structures

None

3.3.*.3 Memory Performance issues

- The memory is an integer value. When putting it into the equation variable it needs to be converted into a string.

3.3.*.4 Memory Design Constraints

None

3.4 Backend/Hidden

These functions deal with the creation and calculations of graphs. The user will never input data directly to them rather data is passed to them via parameters

3.4.1.1 Interface description

calculate_file(read_filename, write_filename)

- This function makes sure that the file sent to it by its parameter meets all specifications of the program. If it does then the proper function will be called. If it does not then it will let the user know that they need to provide a file in order for the program to evaluate the calculation or else the program will not run until a file is provided.

3.4.1.2 Algorithm Model

If a file is provided the program will go onto check if it is an acceptable text file if not then the user will need to use a text file that is acceptable once they have read the prompt given by the user for not using an appropriate text file. Then the program will look to see whether the text file that is provided is in the same directory as the executable. Afterwards, it will check if the calculation that the user has typed is using

Python operators if not then the calculation will not be performed. If the calculation is correct then the user input is passed onto the calculator_model() and then off to calculate() which will display and calculate the input.

3.4.1.3 Restrictions

- Sys module is to be for checking file status.

3.4.1.4 Local Data structures

None

3.4.1.5 Performance issues

None

3.4.1.6 Design Constraints

None

3.4.2.1 Interface description

init_graphing_widgets(root), draw_graph()

This first function will create the graph depending on whatever function the user inputs. Local data structures in side this function will begin to do there role such as creating the graph depending on the user input such as the height, width, and resizing the graph depending on the function. The second function will draw the graph accordingly and has parameters passing from init_graphing_wdigets() to draw_graph().

3.4.2.2 Algorithm Model

1. The minimum and maximum x and y values will be taken into considered before drawing the graph.
2. The plotting data will be stored into two different lists called scaled = [] and cords = []
3. All the functions to make the graph will be run using TK and math module.

3.4.2.3 Restrictions

- The math module must be used.

3.4.2.4 Local Data structures

None

3.4.2.5 Performance issues

None

3.4.2.6 Design Constraints

None

3.4.3.1 Interface description

debug()

This function is used for testing purposes and making sure if any possible issues were to arise then they can be helpful to help fix any errors and/or warnings quickly. It will also let the user know whether they have entered an incorrect file and if it exists or not.

debug_flag can be toggled to ‘TRUE’ or ‘FALSE’ to enable debug messages when the program is running.

The function will be passed throughout the whole program where possible errors or warnings can be caused. A few being with file reading or writing, incorrect input, etc.

3.4.3.2 Algorithm Model

1. The debug flag will be enabled once the program begins to run in order to find any possible errors. This will help the user identify any input issues they may have come across and let them know how to fix them.
2. Check to see which mode the program is running on (console or GUI).

3.4.3.3 Restrictions

None

3.4.3.4 Local Data structures

None

3.4.3.5 Performance issues

None

3.4.3.6 Design Constraints

None

3.4.4.1 Interface description

usage()

This function is primarily used for error handling and outputting messages to the user to let the know that they have typed something incorrectly.

Example:

usage()
“Please provide a file to read calculation lines as the first parameter and a file for writing results to as the second parameter. “

3.4.4.2 Algorithm Model

1. The function will check to see if the files were in the correct directory if not then print out an appropriate message letting the user know.

3.4.4.3 Restrictions

None

3.4.4.4 Local Data structures

None

3.4.4.5 Performance issues

None

3.4.4.6 Design Constraints

None