# 1st Year Project Report

## Specification

I aim to create a program which can evaluate and draw basic equations. It should be have several key features. First it must be able to take input from the user in string form. It should then be able to convert this string into a series of data structures which represent the equation. This should then be displayed to the user. These data structures should then be rearranged into an expression tree structure. The program should then draw the equation with a range from the user if given or the program should calculate a suitable range.

## Design

#### Input

The input function should prompt the user for input and return the input as a string. It should also display help text if prompted by the user.

#### String Splitter

The string splitter should receive the string to be split from the input function. It should then split the string into individual elements strings which should be stored sequentially and returned.

#### String Evaluator

The string evaluator will receive the split string elements and processes them into nodes. It should identify whether the current element should be a variable node (constants are treated as a coefficient to a variable with power zero), an operator node or a bracket node.

Variable will be identified by finding if all the characters of the element are alphanumerical. Determining if the element is an operator consists of finding whether the element is only one character in length and that character is from a list of accepted operator characters. Finding an open/close bracket node requires finding an element one character in length and consists of an open/close bracket character.

Once the entire sequence of elements has been processed the string evaluator will return the element sequence.

#### Bracket Evaluator

The bracket evaluator takes the processed equation and searches for open bracket nodes. Upon finding an open bracket node it will call the bracket node creator function which will create a bracket node to store the contents of the brackets. To find the content of the brackets the bracket node creator which will search the remainder of the equation until it finds either a close bracket node (where it will return the bracket contents) or the end of the equation (where it will return an error). If whilst searching through the remainder of the equation the bracket node creator finds another open bracket node it will call itself. Once the bracket evaluator has reached the end of the equation it will return the amended equation containing brackets.

#### Equation Output and Evaluation Success Confirmation

The equation output will print the equation contents, in order, to the user. It then asks the user whether it has interpreted the input correctly. If correctly evaluated, it will continue on to generating the tree structure. Otherwise it will ask the user to re input the equation and prompt them with help text.

#### Tree Structure Generator

Once the equation has been evaluated successfully the tree structure generator will organise the equation into a tree structure ready for calculation. It will first identify the operator with the lowest priority (this will be placed at the top of the tree as it will be calculated last). The equation will then be split around this operator, the left hand side of the operator will be stored in the left hand side of this operator node and the right side will be stored this way except in the right hand side node. This process will be repeated in operation priority order until all nodes have been processed. The tree structure generator will then return the now structured sequence of nodes.

#### Range Creation

#### Graphical Display