0.0.1 Substitute Algorithm

To actually plot the functions we need to be able to substitute values into the actual functions to know where to draw our points. Eventually we should be able to substitute multiple variables/constants into our function, but for now we will only substitute an x value into our function. This can be an extension for my next iteration.
Now algorithms **??** and **??** on page **??** traverse a tree and output a list (we will implement this using a stack during the actual coding). However the most significant operations and values, i.e. the actions we would do first according to **BIDMAS** [1] , are at the front of the list. This means that in a stack they would be at the bottom, which means they would be popped off last which is not what we want when we evaluate a function. While we can reverse the stack, this is ineffiecient and also defeats the purpose of using a stack.

However this is fixed by the need of this substitute algorithm. We will need to traverse this stack to replace the variables/constants with actual values anyway, so while we do this we can reverse the stack, fulfilling to purposes.

To accomplish this we will create a new stack, of the same height as the original, and then pop each value of the original stack, checking if it is a variable, and if it is substituting the value required. The value will then be pushed onto the new stack. This new stack will then be returned. We also will need to create a copy of the old stack and manipulate that so that we retain the original stack for other substitution calls.

Algorithm 1: Substitution Algorithm

```
1  function substitute(Double x):
2      Stack copy = postFixStack                                  // Copy the Post-Fix Stack
3      Stack substituteStack = new Stack (this.postFixStack.getHeight())     /* Create new Stack
        the same size to reverse it into */
4      for  i=0 to copy.getHeight() do
5          String pop = copy.pop()
6          if pop == "x" then
7              pop = x
8          substituteStack.push(pop)
9      end
10     return substituteStack
11 end
```

[1] **BIDMAS** stands for **Brackets**, **Indices**, **Division**, **Multiplication**, **Addition** and **Subtraction**. It signifies the order that we must do operations on an expression. It is sometimes called **BODMAS** where the **O** stands for **Orders**.