# Project 2 README – The Postfix Desk Calculator

The Postfix Desk Calculator programs provides a simple desktop postfix calculator that supports 5 mathematical binary operations performed positive and negative whole numbers. The program can perform addition, subtraction, division, multiplication and modulation. Other commands can also be used to manipulate or view the stack that holds the numbers.

## Design

The program does not use any files for input or output. The program takes in numbers and characters and performs the appropriate action depending on each character or number input. The program uses a stack to hold numbers entered by the user and all operations are performed on numbers that have already been pushed onto the stack. The stack classes used are stack.cpp and stack.h files which have been provided specifically for this program. Another file, named dsexceptions.h, which defined the exceptions that would be thrown for Underflow, Overflow and Division by Zero.

The stack class included a private variable that indicated the height of the stack, named topOfStack. There are also several methods in the stack class. Overflow and Underflow exceptions are thrown as needed in the following methods.

The push method takes an int as a parameter and pushs the int onto the top of the stack and increments the private topOfStack variable by 1. The method is void and throws an Overflow exception if the stack is full. The pop method is the opposite of the push method, as it pops the most recently inserted item off the stack, which is also the item at the top of stack and decrements topOfStack by 1. The method is void and throws an Underflow exception if the stack is empty. Both methods are modifiers.

The top method returns the most recently inserted item without altering the stack. The method is void and does not throw any exceptions. The topAndPop method combines the pop and top method, as it pops off the most recently inserted item off the stack and returns the value. The method returns an int and throws an Underflow exception if the stack is empty. The top method is not a modifier, but the topAndPop method is.

The isEmpty method checks if the stack is empty or not. If the stack is empty, then the method returns true. If the stack is not empty, then the method returns false. The isFull method checks if the stack is full or not. If the stack is full, then the method returns true. If the stack is not full, then the method returns false. Both methods return Boolean values and are observers.

The makeEmpty method clears the stack of all values and is void, as it does not return anything. The method is a modifier and throws no exceptions.

The calculator program does not require spaces between operations or negative numbers but does require spaces to distinguish between separate positive numbers. Negative numbers were specified by replacing the negative sign with an underscore, because the – symbol was being used to represent the subtraction operation.

## Algorithm

1. Program is started by the user.
2. The program will continually accept lines of numbers and operations from the user.
3. Execute action(s) specified by user until user enters exit command.
   1. If user enters a number, push number onto calculator’s stack, if stack is not full.
      1. Multiple numbers can be entered, but they need to be separated by a space.
   2. If user enters an underscore followed by a number, push the negative of the number onto the calculator’s stack.
      1. Multiple negative numbers can be entered without a space separating the numbers.
   3. If user enters p, print the top value of the stack without altering the stack, then print a new line.
   4. If user enters n, print the top value of the stack and pop the top value off, then print a new line.
      1. Operation is not executed if stack is empty.
   5. If user enters f, print the contents of the stack without altering it. A new line is printed after each value.
   6. If user enters c, empty the stack.
   7. If user enters d, duplicate the top value and push it onto the stack.
      1. Operation is not executed if stack is full.
   8. If user enters r, swap the order of the top two values on the stack.
      1. Operation is not executed if stack has less than 2 values.
   9. If user enters +, pop the top two values of the stack and push the sum of the values onto the stack.
      1. Operation is not executed if stack has less than 2 values.
   10. If user enters -, pop the top two values of the stack and push the difference of the first value from the second value onto the stack.
       1. Operation is not executed, if stack has less than 2 values.
   11. If user enters \*, pop the top two values of the stack and push the product of the values onto the stack.
       1. Operation is not executed if stack has less than 2 values.
   12. If user enters /, pop the top two values of the stack and push the quotient of the second value by the first value onto the stack.
       1. Operation is not executed if stack has less than 2 values or if dividing by 0.
   13. If user enters %, pop the top two values of the stack and push the remainder of the division, that the / command would do, onto the stack.
   14. If the user enters a space, nothing happens.
   15. For all other characters not defined above, the program will alert the user of an invalid input and continue asking for another line of input.
   16. If user enters Ctrl + D, then the program will exit.