CSC 122: Calculator II

Recursion

1 Introduction

In this lab you will be required to take either your completed code from Calculator I or the solution provided by the instructor and implement order of operations and some additional mathematical operations that can be solved using recursion. After completing this lab you will have a fully functioning calculator.

2 Objective

In this lab you will take the finished code from Calculator I and use recursion to implement order of operations and a few mathematical operations that are well suited for recursion. Such operations include square root and greatest common denominator. This lab will also take advantage of clever use of a basic data structure to implement a complex operation. Order of operations will be implemented by recursively creating stacks.

3 Activity

In this section the overall activity of this lab will be described with a bit of background information to prepare you for the implementation section that follows.

3.1 Background Information

For order of operations make sure that you obey the following order:

1. Parentheses
2. Exponents
3. Multiplication/Division (left to right)
4. Addition/Subtraction (left to right)

Part of this lab requires the use of a stack. A stack is a data structer that has the properties of last in, first out. This can be visually represented by a stack of papers on a desk in which the only piece of paper you can access is the top on. To get to the bottom of the stack you must ’pop’ off the top piece one at a time until you have reache the last piece. Conversely you can only ’push’ pieces back on top of the stack one at a time. This idea will be used extensively in evaluating expressions in the order specified above, but the actual implementation of a stack will be simplified with the use of Java’s built in Stack class. Please look at Oracle’s Java API’s for more information specific to the Java Stack class.

For the recursive math functions please research online about how to implement the specific required mathematical operations recursively. They are both really simple and finding examples should not be hard.

3.2 Implementation

In this section the implementation of this lab will be described in a manner such that you should be able to complete it with your current knowledge and a moderate amount of research.

3.2.1 Order of Operations

In this part of the lab you will be implementing order of operations in the calculator that you completed in the Calculator I lab. To do the will require the use of some stacks and some flags.

Initially you will need to create a stack and a flag associated with that stack. This stack will contain the initial numbers and operators that will be typed in by the user. The flag will be set to true when a multiplication or division operator is reached in the stack and then as soon as another number pushed onto the stack the current expression is popped off of the stack and its result is pushed back onto the stack. Elsewise if it is an addition or subtraction operator it will continue reading input until a multiplicaiton or division operator or a parentheses (this is a special case) is reached.

If a parentheses is reached you must initialize a new stack with a new flag and then push and pop the values into that stack as described above. This should be done until the close parentheses is reached, at which point all that should be in the stack is a single expression that should require order of operations to compute. This expression should be popped off, evaluated, and pushed back. Once all that is in the stack is a single value, it is popped off and that value is pushed onto the stack that called the stack that just closed out.

Exponents should be handled immediately unless parentheses are involved, in which the value inside the parentheses should be calculated and then the exponent should be calculated.

3.2.2 Recursive Math Functions

Many math functions such as square root and greatest common denominator can be calculated recursively. Recursion is the process where a function calls itself and passes a new value. After the last recursive call has been completed the functions then return their values back to the function that called them. For this lab you must implement square root and greatest common denominator using recursion. The best way to approach this is to find a way online to solve these problems using recursion and use that as an example for your own implementation.

4 Conclusion

Upon completion of this lab you should have a firm grasp on recursion and use of the basic stack. You should also be more comfortable working with GUIs and adding back-end functionality to a GUI, as well as have a fully functioning calculator.

5 Deliverables

You are required to turn in your entire project source code as a zip file through the means specified by the teacher. The project must successfully compile into an Android application. If the application does not compile, your project will not be graded. If the application does not run on a phone, it will receive minimal marks. Any function that does not work will result in a loss of points. A force close (crash) will also result in a deduction of points. Any known bugs should be documented in a readme file included with the project. This may allow for a smaller deduction of points depending on how serious the bugs are.