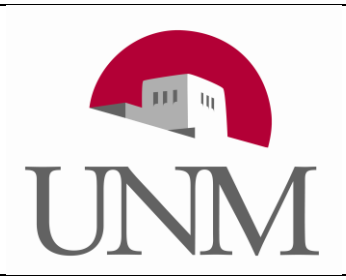

The University of New Mexico

CS 241—Data Organization

Lab 4: Postfix Calculator

Description:

Sections 4.2 and 4.3 of Kernighan and Ritchie build up various parts of a postfix calculator. Section 4.4 and 4.5 show how to break the program up into multiple files. Your task is to put it all together and make it work with a few extensions.

Write a program that reads from the standard input stream and evaluates each line of input as a postfix expression. Every constant must be read and stored as a double.

Your program must recognize the standard binary operators: **`'+'`**, **`'-'`**, **`'*'`** and **`'/'`**. All operators must be calculated in double precision. Thus, an input of:

```
1 3 /
```

evaluates to:

```
0.3333333333333333
```

All input numbers and operators must be space delimited.

Your program must recognize:

1) Input numbers with zero or one decimal place.
2) With or without a leading 0.
3) Numbers that are zero or positive (no leading symbol).
4) Numbers that are negative (with a leading **`'-'`**, followed immediately by a digit or decimal point). A **`'-'`**, followed by a space must always be interpreted as the binary minus operator.

Your program must produce one line of output for each line of input.

Each line of output must echo back each character of the input line followed by **`" = "`** and either the result of the expression formatted as **`"%.3f"`** or, if the expression contains an error, then **`"Error"`**.

Errors:

If a line contains an error, print **`" = Error"`** after echoing the input. A line is an error if it:

1) Contains a character that is not a digit, **`'+'`**, **`'-'`**, **`'*'`** and **`'/'`**.
2) Contains more than 200 characters including the spaces.

3) A number has more than one decimal point.
4) An operator is not preceded or not followed by a space.
5) A number has more than one negative symbol.
6) A negative symbol is not preceded by a space or not followed by a digit.
7) A number is out of range of a **`double`** (on cs.unm.edu).
8) The expression is not a legal post-fix expression:
   a) A binary operator is read when the stack does not contain two numbers.
   b) A '**`\n`**' or EOF is read when the stack is not empty.

Examples:

Input line	Output
`1 2 - 4 5 + *`	`1 2 - 4 5 + * = -9.000`
`0.1  0.02 + 0.003 + -1 +`	`0.1  0.02 + 0.003 + -1 = -0.877`
`1 2 + +`	`1 2 + + = Error`
`1 2 %`	`1 2 % = Error`
`1 2 3 +`	`1 2 3 + = Error`

Grading Rubric (total of 20 points)

[-2 points]: Program compiles with warnings on cs.unm.edu using /usr/bin/gcc with no options.

[-5 points]: Code does not follow the CS-241 standard. Note: to be compliant with the coding standard, you must change non-compliant parts of the book's code. For example, the book uses the global variable sp for the next free stack position. That is too short and too non-descriptive for our standard. Change sp to stackPosition, nextStackPosition, stackTop, topOfStack, stackIndex, stackIdx, stackPos or some other name more descriptive than sp.

[5 points]: The program is attached in Blackboard Learn as a .zip archive containing a single directory consisting of the five files: main.c, calc.h, getop.c, stack.c and getch.c with the structure shown in section 4.5 of the textbook.

The .zip archive must be named:

CS-241.postfix.*yourfirstname*.*yourlastname*.zip

The program must compile within a single directory with the command:

gcc *.c

[10 points]: Passes diff test with the 10 test cases given in postfix.in and postfix.out on the class website (+one point each).

[5 points]: Passes 5 unknown test cases (+one point each).