EECE.2160: ECE Application Programming

Spring 2017

Programming Assignment #3: A Simple Calculator

Due **Friday, 2/10/17**, 11:59:59 PM

1. Introduction

In this assignment, you will work with C conditional statements to implement a simple calculator program. You will also work with basic output formatting when printing your results.

2. Deliverables

Submit your source file by uploading it directly to your Dropbox folder. Ensure your source file name is ***prog3_calc.c***. You should submit only the .c file. Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

3. Specifications

Input: Your program should prompt the user to enter the following:

- The desired precision for the result and operands
- A simple arithmetic expression of the form *a op b*, where *a* and *b* are operands and op is one of the following operators: `+, -, *, /`. Examples include:
  - `5 + 3`
  - `-0.777 * 17.175`
  - `22 / 11`
  - `-1.2 - -3.4`

Output: Given a valid expression, your program should calculate the result and reprint the entire expression as well as its result, using the desired precision. If the precision is 2, the expressions listed above will produce the following output:

- `5.00 + 3.00 = 8.00`
- `-0.78 * 17.18 = -13.40`
- `22.00 / 11.00 = 2.00`
- `-1.20 - -3.40 = 2.20`

See Section 4: Test Cases for more sample program runs.

Error checking: Your program should print an error under any of the following conditions:

- Any of the inputs are incorrectly formatted and therefore cannot be read correctly using `scanf()`.
  - `scanf()` returns the number of values correctly read; you can store this number in a variable and check its value. Say you have the following line of code:

    ```
    nVals = scanf("%d %d %d");
    ```

    If the user enters:
    - `1 2 3` → `nVals == 3`
    - `1 2 a` → `nVals == 2` (`'a'` is not part of a valid `int`)
    - `1.2 2 3` → `nVals == 1` (`'.'` is not part of a valid `int`, but `1` is read correctly)
    - `X1 2 3` → `nVals == 0` (`'X'` is not part of a valid `int`)
- The precision is not a valid value (must be ≥ 0).
- The user tries to divide by 0.
- The operator entered is not a valid operator.

4. Hints and Tips

Variable precision: As discussed in class, precision controls the number of digits shown after the decimal point when printing `float` or `double` values. While we looked at examples using a constant value to specify precision, an integer variable can be used as well. To do so, an asterisk `*` should be used in place of the precision, as shown in the example below: `p` is the precision and the variable `v` is being printed:

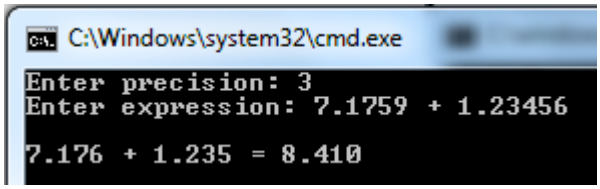
```
printf("%.*lf", p, v);
```

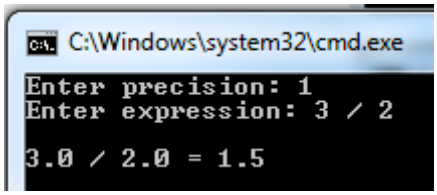
Note that if multiple values are printed, you must specify the precision for each one. The example below shows how to print three different variables—`v1`, `v2`, and `v3`—using the variable `prec` to control the precision of each:

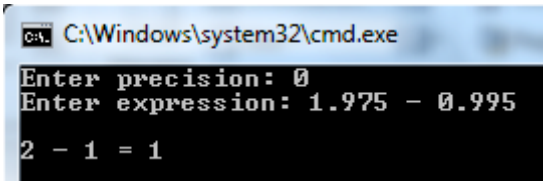
```
printf("%.*lf %.*lf %.*lf", prec, v1,
          prec, v2, prec, v3);
```

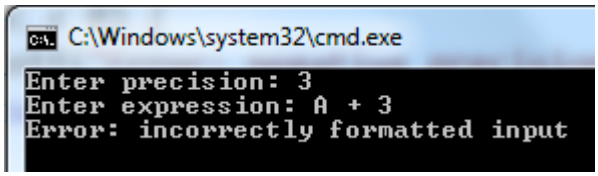
5. Test Cases

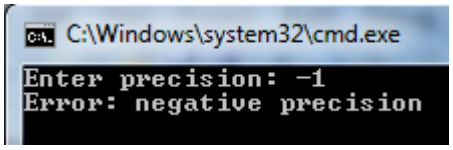
Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your lab, but will also generate additional cases that will not be publicly available. Note that these test cases do not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.











Remember, if you are using Visual Studio, to get your program to terminate with a message saying, “Press any key to continue …”, use the **Start Without Debugging** command (press Ctrl + F5) to run your code.

If you need to insert extra code at the end of your program to get that program to pause when executing (for example, an infinite loop or the system("pause") function), please remember to comment out that code prior to submitting your program.