16.216: ECE Application Programming

Spring 2012

Programming Assignment #4: A Simple Calculator

Due **Monday, 2/27/12**, 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 directly to Dr. Geiger (Michael_Geiger@uml.edu) as an e-mail attachment. Ensure your source file name is ***prog4_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.

For up to 10 extra credit points, submit a flowchart describing your planned solution by 11:59 PM on Wednesday, 2/22/12. Note that:

- No hints will be given or questions answered about the extra credit problems, either in person, through e-mail, or on the message board.
  - You should not post anything on the message board about the flowcharts—please do not ask or answer any questions.
- Late flowcharts **will** be accepted, but are subject to the same rules as late programs (-1 point for first late day, -2 points for second late day, etc.).
- Flowcharts are not eligible for regrade requests.
- Flowcharts may be hand drawn or generated using a program such as Microsoft Visio.
- Students submitting a hand drawn flowchart (or a hard copy of a computer-generated one) **must submit the flowchart in person by 3:00 PM** on whatever day they submit, while electronic submissions will be accepted until the end of each day.
- Your flowchart will be evaluated on:
  - Clarity: can I clearly understand the process you describe?
  - Detail: the more detail you provide, the better, as a more detailed flowchart will make it easier for you to write the program.
  - Correctness: do you describe the process in such a way that it will actually translate to a valid, running program? Does your flowchart meet all of the program requirements?

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 = -13.34`
- `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 and immediately exit under any of the following conditions:

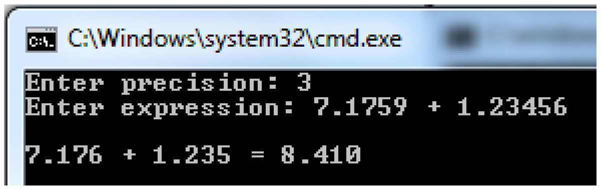
- Any of the inputs are incorrectly formatted and therefore cannot be read correctly using `scanf()`.
  - `scanf()` returns the number of values correctly read; you can store this number in a variable and check its value. Say you have the following line of code:

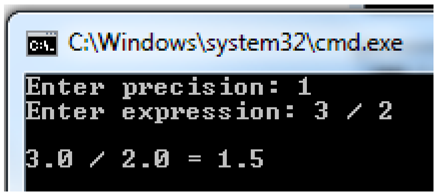
    ```
    nVals = scanf("%d %d %d");
    ```

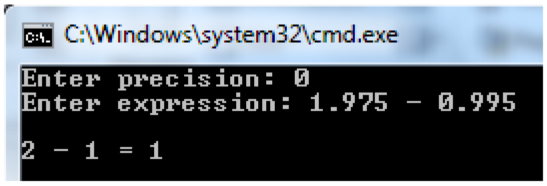
    If the user enters:
    - `1 2 3` → `nVals == 3`
    - `1 2 a` → `nVals == 2` (`'a'` is not part of a valid `int`)
    - `1.2 2 3` → `nVals == 1` (`'.'` is not part of a valid `int`, but `1` is read correctly)
    - `X1 2 3` → `nVals == 0` (`'X'` is not part of a valid `int`)
- The precision is not a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

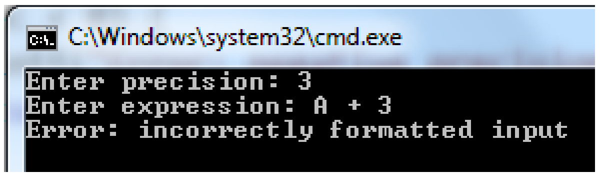
4. Test Cases

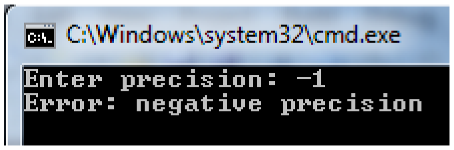
Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your lab, but will also generate additional cases that will not be publicly available. Note that these test cases do not cover all possible program outcomes. You should create your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


```
Enter precision: 3
Enter expression: 7.1759 + 1.23456

7.176 + 1.235 = 8.410
```


```
Enter precision: 1
Enter expression: 3 / 2

3.0 / 2.0 = 1.5
```


```
Enter precision: 0
Enter expression: 1.975 - 0.995

2 - 1 = 1
```


```
Enter precision: 3
Enter expression: A + 3
Error: incorrectly formatted input
```


```
Enter precision: -1
Error: negative precision
```

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.