CSI 333 – Programming at the Hardware-Software Interface – Spring 2017  
Project II

The total grade for the assignment is 100 points.

You must follow the programming and documentation guidelines available in the Blackboard module Projects.

# Description:

You are required to write a C program to carry out *a strict-left-to-right* evaluation of an arithmetic expression consisting of integer constants 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and the operators +, −, ∗, and /. Here, the operator / denotes integer division; that is, the remainder is discarded. In a strict-left-to-right evaluation, there is no notion of precedence. For example, the value of the expression 6+ 4 ∗3 when evaluated in a strict-left-to-right fashion is 30. (Under usual precedence rules, where multiplication has higher precedence than addition, the value of the above expression would be 18.) Similarly, the value of the expression 6 + 4 ∗ 3/7 − 9 when evaluated in a strict-left-to-right fashion is −5.

Here is some additional information about the input.

1. Each input expression is terminated by the newline (’\n’) character.
2. There may be zero or more spaces between successive non-blank characters of an expression.
3. You may assume that the given expression is valid; that is, it satisfies all of the following conditions.
4. The expression consists only of integer constants, the operators +, −, ∗, / and spaces.
5. In particular, the expression won’t contain parentheses.
6. Each integer constant in the expression consists of *only one* decimal digit.
7. The expression begins with an integer constant, *without* any preceding sign.
8. In the expression, integer constants and operators alternate.

The outline of your C program is as follows.

1. Prompt the user for an expression.
2. Read the expression character by character and carry out a strict-left-to-right evaluation of the expression.
3. Print the value of the result obtained in Step 2 and *stop*.

Thus, each time your program is executed, it should handle just one expression. Bear in mind that your program reads its input from stdin and writes its output to stdout. No error checks are needed.

Notes:

* An expression consisting of a single digit integer constant, without any operators, is a valid expression.
* After each call to the function printf, include the following C statement: fflush(stdout);. Example: printf("Value = %d\n", result); fflush(stdout);.

# Electronic Submission:

Important Notes: ignoring any of the following rules will result in penalty or even ZERO grade for the project.

* 1. For Project 2 you must turn in the file **named** “**p2.c”**.
  2. At the top of each of your C source file the following information must appear in the form of comments:
  3. your name,
  4. your UNIX login id,
  5. the name of your lab instructor and
  6. the day and time of your lab class.
  7. Make sure that your programs compile and produce correct results on the UNIX machines (itsunix.albany.edu) supported by Information Technology Services (ITS) unit of UAlbany. Programs that cause compiler or linker errors on the ITS UNIX machines will NOT receive any credit.
  8. Using the turnin-csi333 program as discussed below is the ONLY acceptable way of submitting programming assignments in this course. You should NOT email the files to the instructor or to the TAs.
  9. Remember that you must submit only your C source files. DON'T turn in unnecessary files (e.g. object files with extension “.o” created by compiling C source files, executable files such as “a.out”, etc.).

To submit your files electronically, you must have the source files on one of the ITS UNIX machines. For this project the file p2.c must be in your working directory and you must be logged on to one of those machines to actually carry out the electronic submission.

To perform submission you should type the following command to the UNIX operating system:

/usr/local/bin/turnin-csi333 -c csi333 -p hw2 p2.c

After you issue the above command, the system responds with:

The sections of csi333 are:

FR\_1025

FR\_1235

FR\_0140

WE\_1235

Enter your section:

Depending upon the day and time of your weekly discussion section, you would type the appropriate section. For example, if your discussion class meets on Wednesdays at 12:35 PM, you would type WE\_1235 followed by the return key. The system will then respond with

Your files have been submitted to csi333, hw2 for grading.

In the above message, "hw2" refers to the name of the project that is currently active. If you submit your program during two days after due date the name of the project will be "hw2-late". Attempts to submit the program after the two day grace period will fail.

Additional information about the turnin program:

* 1. If you use the turnin command above again at a later time (before the deadline), then the files submitted previously would be completely replaced by the newly submitted files.
  2. At any time, you can obtain the names of the files that you have submitted to the current project using the following command:

/usr/local/bin/turnin-csi333 -c csi333 -v

# Some sample data to test your program:

Important Note: Some sample inputs that can be used to test your programs are given below. However, you should remember that when we compile and run your source files, we will use other data. Just because your programs work for the sample inputs given below, you shouldn't assume that they will work for all inputs. Therefore, you should test your programs thoroughly with other input values.

Some expressions and their values when evaluated in a strict left-to-right fashion are given below.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 6 | 6 |
| 9 + 5 \* 0 - 7 | -7 |
| 0+ 7 + 4 \* 5 - 9 | 46 |
| 9+9-7+4-2/2+4-6 | 4 |
| 7\*8-9\*4-5\*6-6/3+4 | 368 |

# Program Grading:

Programs will be graded using a script written by the TAs. The script will compile your source program, generate the executable version and run the executable on new test data. The TAs will grade the version that you submit; once the submission is closed, you won’t be allowed to make any changes to your program.

Points: 90 points for correctness and 10 points for structure/documentation.

# Example of program execution:

The following examples assume that the executable version of the program is in the file p2.out.

unix2> p2.out

Enter expression: 9\*2 - 5/3 -9

Value = -5

unix2>