CSC 322
Systems Programming
Lab Assignment 1
Fall 2017

I. Goal

In the first lab, we will develop a system program which provides mathematical commands using C language. It must be an errorless system program, in which user starts the program, enter commands, see the desired result, and finishes it if he or she wants to quit. For students who has the first time to use C language, this project will be a good initiator to learn a new programming language. Students who already know C language also will have a good opportunity to warm up their programming abilities and get prepared for the rest of labs.

II. Introduction

A. Basic Calculations

The system program provides four basic mathematical commands shown in the below table. Each command presents an operator, and requires operands as parameters. For instance, when user enters **sum 4 2**, the program returns the result **6** (= 4 + 2) in the next line. And then it will go to the next prompt waiting for user's next command. Note that each operator should have two operands, and if a command has more than 2 parameters, it is an incorrect syntax. The commands are following:

Command	Description
sum	Summation of parameters. For instance, **sum 4 2** calculates 4 + 2.
mul	Multiplication of parameters. For instance, **mul 4 2** calculates 4 x 2.
sub	Subtraction of parameters. The first parameter is minuend, and all other parameters are subtrahends. For instance, **sub 4 2** calculates 4 – 2.
div	Diving two numbers. It will allow only two parameters. The first parameter is dividend and the second parameter is divisor. For instance, **div 4 2** calculates 4 ÷ 2 = 2

B. Collatz Conjecture

The system program provides an advance mathematical command, which is called Collatz conjecture. The Collatz conjecture concerns what happens when we take any positive integer *n* and apply the following algorithm:

$$n = \begin{cases} n/2, & \text{if n is even} \\ 3 \times n + 1, & \text{if n is odd} \end{cases}$$

The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, if *n* = 19, the sequence is following:

19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

The command has one parameter. If it has more than two parameters, it is an incorrect syntax.

Command	Description
col	Finding collatz conjecture of parameter. For instance, **col 19** returns a collatz sequence.

C. Program Design

The system program **should run on Linux machine at *cs.oswego.edu***. Once it is compiled and performed, it draws user to its own computing environment by showing prompt following the format below.

```
yourID> $
```

Then user will enter a command and parameters if necessary. The syntax of the command is following.

```
command [parameter]*
```

Note that some commands such as basic calculations need multiple parameters. When each command finishes its own calculation, it will show the results on the screen as shown in the Figure 1, and the new prompt follows for the next command.

```
[jwlee@deepwater:~]$ cc lab1_jwlee.c –o lab1
[jwlee@deepwater:~]$ ./lab1
jwlee> $ sum 3 2
5
jwlee> $ mul 5 7
35
jwlee> $ sub 6 7
-1
jwlee> $ div 4 2
2
jwlee> $ col 19
19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
jwlee> $ col 35
35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
jwlee> $ bye
bye
[jwlee@deepwater:~]$
```

Figure 1. Example of project 1

To quit the program, user should enter command **bye** without any parameters. Then the program ends after a simple message *bye*, which notifies the end of the program. It has no parameter, and if it has any parameter, it is an incorrect syntax.

Command	Description
bye	Finishing the program.

D. Extra work

Each basic mathematical command in the original program can calculate only two operands, for instance, **sum 3 5** will return 8 (=3 + 5). In the real world, however you may be asked to calculate more complicated formula such as is 5 + (7 * (4 - 2)). Thus, you will enhance these commands so that they can work for more complicated calculation as an extra work. The example formula has three operators, four operands, and two pairs of parentheses. For calculating the example, you will enter the following command:

```
sum 5 (mul 7 (sub 4 2))
```

For simplicity, the system program will restrict the maximum number of operators, so there are no more than 7 operators.

This is not mandatory but optional. Therefore, it will give you 6 extra points, which is worth 30% of the assigned points. Those who have experiences of C before are strongly encouraged to do this extra work.

III. Requirements

A. Developing environment

The program should be **implemented in C only**. The program in another language will not be graded. And the program should be executable in CS Linux machine. The program which cannot be compiled or executed in the CS Linux machine will be considered as incomplete program, and will lose most points.

B. Handling exceptions

The program must detect errors while running. The errors may occur when user violates the syntax of the command or enters wrong inputs (character, more than 2 numbers, etc.). Also any mathematically incorrect calculations, for instance, **div 3 0**, must be detected. When the program detects any error cases, it should handle properly by giving some error messages to the user, and still be runnable. **The program, which fails to detect such errors and to properly handle them, will be taken off penalty points**.

C. Readability of source code

The source code should be easy to read with **good indentation** and **proper amount of comments**.

D. Creating a *readme* file

The program should be submitted along with a readme file which has information such as student ID, name, etc. It also may provide instruction to run this program if exists. Those who complete extra work need to describe their progress and implementation in one short paragraph for instructor to grade the extra work.

The source file and readme file should be named following the format below. **Those who are not following the naming rule will also be taken off penalty points**.

```
lab1_yourID#.c
lab1_yourID#_readme.txt
```

E. Submitting by due

The program should be submitted to Blackboard within the two weeks after the project is posted. **Due is September 21**. All submission **by 11:59 PM** on that day will be accepted without any penalty. On the due date, Blackboard may be suffering of too much network traffics and be unstable. There is no excuse about the issue, therefore you are strongly recommended to submit earlier than the due date.

IV. Grading

This project is assigned 20 points, which is 10% of the final grade. Groups who complete extra work will receive 6 additional points.

A. Grading criteria

The project will be graded by evaluating the requirement. Any missing and unsatisfiable criteria will take off points. The tentative and brief criteria are below.

- Compilation: 5 points (25%)
- Execution: 10 points (50%)
- Error detection & others: 5 points (25%)

B. Late penalty

Late submission will take off **10% per day per week** after due date. **Thus, submission after 10 weeks will not be accepted in any circumstances**.

V. Academic Integrity

Any dishonest behaviors will not be tolerated in this class. Any form of plagiarism and cheating will be dealt with according to the guidelines on the Academic Integrity Policy on-line at http://www.oswego.edu/integrity. For more information about university policies, see the following online catalog at:

http://catalog.oswego.edu/content.php?catoid=2&navoid=47#stat_inte_inte

Student who is against the honor code will not have any credits in this project.