HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

BBM203 PROGRAMMING LAB.
ASSIGNMENT 2

Subject : Data Structures and Algorithms (Stack, C++)
Submission Date : 20.07.2017
Deadline : 29.07.2017, 23:59 pm
Programming Language : C++
Advisors : Asst. Prof. Dr. Adnan ÖZSOY, R. A. Cemil ZALLUHOĞLU

1. INTRODUCTION / AIM:

In this experiment you're expected to gain knowledge on advanced C++ topics. Prior knowledge on basic C++ syntax and class declarations is assumed.

2. BACKGROUND INFORMATION

2.1. C++: A MULTI-PARADIGM LANGUAGE

C++ is a general-purpose computer programming language. Since the 1990s, C++ has been one of the most popular commercial programming languages.

It is a multi-paradigm language supporting procedural programming (like C, Pascal, and other structured imperative languages), data abstraction (i.e. private members), object-oriented programming, and generic programming (i.e. templates).

Its template mechanism is so powerful that, in the recent years C++ has been “rediscovered”, even to surprise
its developers.

2.2. C++ STANDARD LIBRARY

The original C++ standard library contained the C standard library and a new I/O streams library, but lacked any standard base classes or templates. Later SGI's STL, which contains a set of class and function templates has been incorporated in the C++ standard. C++ standard library is still being extended.

There have been changes to inclusion and usage of library items. All the items are grouped in the “std” name space. And, the header names do no longer contain any suffix (no .h, .hpp or .hxx) and C standard library header names are preceded by “c” (e.g.: cmath).

3. EXPERIMENT

In this experiment, your aim will be to create a mathematical expression converter and evaluator. You have to implement addition, subtraction, multiplication, division and power operations.

When there is a division by zero, the result should be NaN (not a number) and any operation with at least one NaN operator should result as NaN.

Your program should evaluate the arithmetic expressions given in a custom stacked notation and write the results to the output file. For I/O operations, you should use I/O streams library of C++.

Your program should **convert the given arithmetic expression,** described in the following sections, **to prefix, infix, and postfix expressions.**

The arithmetic expressions can be evaluated using the **stack data structure**. When evaluating expressions, you can use **separate stacks for operators and numbers.**

An arithmetic expression in the custom stacked notation consists of N operators after N+1 numbers. The algorithm to evaluate expressions in the custom stacked notation is as follows:

1. Take an operator from the end of the operator sequence, and two numbers from the end of the number sequence, and perform the operation according to the operator (addition, subtraction …).
2. Add the result of the operation to the end of the number sequence.
3. Return to step 1 until there are no more operators left.

Here is an example of an arithmetic expression and how it looks after each iteration:

Input	3 4 7 5 + * -
	3 4 2 + *
	3 8 +
Output	11

3.1. EXECUTION

The name of the compiled executable program should be “ncalc”. Your program should read input/outputs file names from the command line, so it will be executed as follows:

ncalc [input file name] [output1 file name][output2 file name] [output3 file name] [output4 file name]

You can see sample inputs and outputs in ftp site. The program must run on DEV (dev.cs.hacettepe.edu.tr) UNIX machines. So make sure that it compiles and runs in one of the UNIX lab. If we are unable to compile or run, the project risks getting zero point. It is recommended that you test the program using the same mechanism on the sample files (provided) and your own inputs. You must compare your own output and sample output. If your output is different from the sample, the project risks getting zero point, too.

3.2. INPUT/OUTPUT FORMAT

In the input file, every line will represent a separate arithmetic expression. There will be whitespace (spaces and/or tabs) between operators and numbers. You do not need to check the input for syntax or semantic errors, every line will be a meaningful expression. An example input file can be given as follows:

Input	3 4 7 5 + * -

Your program should write the results of the expressions to the output file so that each line in the output file will contain the result of the expression in the corresponding line of the input file. Given the example input file above, the output file should be:

Output1 (Prefix)	+ 3 * 4 - 7 5
Output2 (Infix)	3 + 4 * (7 - 5)
Output3 (Postfix)	3 4 7 5 - * +
Output4	11

3.3. DESIGN EXPECTATIONS

You must perform a proper object oriented design for your solution. Writing spaghetti, or even structured code, without using Object Orientation or C++ features could render your entire assignment "unacceptable" and make it subject to huge (if not complete) grade loss.

4. EVALUATION

4.1. REQUIRED FILES

You should create and submit a ZIP archive in the following structure for evaluation. An invalid structured archive will cause you partial or full score loss.

You are required to submit a Makefile[1], which will be used to compile your program to generate the ncalc executable.

Directory	Files	Description
Source	*.cpp, *.h, Makefile	Program source/header files and Makefile
Report	*.pdf	Your report (Only pdf format is accepted)

4.2. REPORTS

- **Report will be 30 points.**
- Please explain your design, data structures and algorithms
- Give necessary details without **unnecessary clutter**
- **Guide: ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/FormatForLabReports.doc**

4.3 GRADING

Your report score will be calculated as follows

$$Report\ Final = Report\ \frac{Score * Exec\ Score}{70}$$

LAST REMARKS:

- The output of your program will be graded **automatically**. Therefore, any difference of the output (even a smallest difference) from the sample output will cause an error and you will get 0 from execution. Keep in mind that a program that does not work 100% right is a program that works wrong.
- Regardless of the length, use **UNDERSTANDABLE** names to your variables, classes and functions.
- Write **READABLE SOURCE CODE** block
- **You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ Deadline is: 23:59 pm. No other submission method will be accepted.**
- Do not submit any file via e-mail related with this assignment.
- **SAVE** all your work until the assignment is graded.

---

[1] Make file example http://mrbook.org/tutorials/make/

- The assignment must be original, **INDIVIDUAL** work. Duplicate or very similar assignments are both going to be punished. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms or source codes.
- Copying without citing will be considered as cheating. Citing does not make it yours; cited work will get 0 from respective section
- You can ask your questions through course's Piazza Group and you are supposed to be aware of everything discussed in the Piazza.

ONLINE RESOURCES

- Wikipedia C++ Entry: http://en.wikipedia.org/wiki/C_plus_plus
- SGI STL Docs: http://www.sgi.com/tech/stl/
- libstdc++ Docs: http://gcc.gnu.org/onlinedocs/libstdc++/latest-doxygen/
- About.Com STL Tutorials: http://cplus.about.com/od/stl/
- C++ FAQ: http://www.parashift.com/c++-faq-lite/

REFERENCES

1. Fundamentals of Data Structures, Ellis Horowitz
2. C++: The Complete Reference, Herbert Schildt