COMP2119A Introduction to Data Structures and Algorithms

Programming Assignment 1 **Due Date:** 7pm, Oct 22, 2019

Rules: discussion of the problems is permitted, but writing the assignment together is not (i.e. you are not allowed to see the actual solution of another student).

Course Outcomes

- **[O4]. Implementation**

[**O4**] In this assignment, you are requested write a program to solve four tasks, **A**, **B**, **C** and **D**, using ADT we learned during class.

Your program will be judged by a computer automatically, please follow the exact format. You should read from standard input and write to standard output. e.g, scanf/print, cin/cout.

Languages.

We only accept C/C++ programming languages.

Judging.

Please note that your solution is automatically judged by a computer.

Solutions that fail to compile or execute get 0 points.

We have designed 40 test cases, 10 for each task. Each test case is of 2.5 points.

The time limit for each test case is 1 second.

For each test case, you will get 2.5 points if your program passes it otherwise you will get 0.

Self Testing.

You should test your program by yourself using the provided sample input/output file.

The sample input/output is **different** from the ones used to judge your program, and it is designed for you to test your program by your own.

Note that your program should always use standard input/output.

To test your program in Windows:
1. Compile your program, and get the executable file, “main.exe”
2. Put sample input file, “input.txt”, in the same directory as “main.exe”
3. Open command line and go to the directory that contains “main.exe” and “input.txt”
4. Run “main.exe < input.txt > myoutput.txt”
5. Compare myoutput.txt with sample output file. You may find “fc.exe” tool useful.
6. Your output needs to be **exactly** the same as the sample output file.

To test your program in Linux or Mac OS X:
1. Put your source code “main.cpp” and sample input file “input.txt” in the same directory.
2. Open a terminal and go to that directory.
3. Compile your program by “g++ main.cpp -o main”

4. Run your program using the given input file, “./main < input.txt > myoutput.txt”
5. Compare myoutput.txt with sample output file.
6. Your output needs to be **exactly** the same as the sample output file.
7. You may find the **diff** command useful to help you compare two files. Like “diff -w myOutput.txt sampleOutput.txt”. The $-w$ option ignores all blanks (SPACE and TAB characters)

Note that myoutput.txt file should be **exactly** the same as sample output. Otherwise it will be judged as wrong.

Submission.

Please submit your source files through moodle.

You should submit four source files(**without** compression), one for each task.

Please name your files in format UniversityNumber-X.cpp for X in $\{A, B, C, D\}$, e.g. 1234554321-A.cpp.

Task A
Given an empty stack and some operations (*push* and *pop*) on the stack, print the popped element for each *pop* operation. When applying *pop* operation on an empty stack, output 'false' and then terminate the program. The number of operations is at most 1000. The elements of the stack are integers. All inputs in the test case are valid. (You don't need to consider the case when format of input is wrong.)

You should implement a stack by yourself without using built-in classes.

Input:
The input contains multiple lines, where each line contains one operation.

Output:
Print the results, seperated by spaces. Having a trailing space at the end of the line or not does not matter.

Examples:

1. Input:

   ```
   push 1
   push 2
   pop
   push 9
   pop
   ```

   Output:

   ```
   2 9
   ```

2. Input:

   ```
   push 3
   push 12
   pop
   pop
   pop
   push 4
   pop
   ```

   Output:

   ```
   12 3 false
   ```

Task B

Rearrange a singly linked list in such a way that all odd nodes together followed by the even nodes. Note that here we are talking about the node number and not the value in the nodes. Your program should run in $O(1)$ space complexity and $O(n)$ time complexity. The first node is considered odd, the second node even and so on. $1 \leq n \leq 1000$.

Note: You must refer to the folder 'sketch for B': do not change the code in main.cpp and ListNode.h, and you are only allowed to edit 1234554321-B.cpp. You just need to submit 1234554321-B.cpp at the end. (Use your own university number.)

Input:

The input contains two lines.
The first line contains an integer n – length of the list.
The second line contains n distinct integers, standing for the list.

Output:

Print the required list, containing n integers. Having a trailing space at the end of the line or not does not matter.

Examples:

1. Input:

   6
   1 5 3 7 8 11

   Output:

   1 3 8 5 7 11

2. Input:

   7
   2 1 3 5 6 4 7

   Output:

   2 3 6 7 1 5 4

Task C
Given a parentheses string s, compute the score of the string based on the following rule:

- If s is not balanced, the score is 0.
- () has score 1.
- AB has score A + B, where A and B are balanced parentheses strings.
- (A) has score 2 * A, where A is a balanced parentheses string.

A balanced string satisfies the following requirements:

- The number of '(' equals the number of ')' in the string.
- Counting from the first position to any position in the string, the number of '(' is greater than or equal to the number of ')' in the string. (For example, ')(' is not balanced.)

The length of s is at most 30.

Input:

The input is a parentheses string s.

Output:

Print the score of s.

Examples:

1. Input:
   (())

   Output:
   2
2. Input:
   (()(()))

   Output:
   6
3. Input:
   ((()())

   Output:
   0

Task D
Given n strings, output the number of distinct strings. The sum of length of strings is at most 10^6.

Input:
The input contains two lines:
The first line is an integer n, which is the number of strings in the second line.
The second line are n strings.

Output.

Output an integer – the number of distinct strings.

Examples.

1. Input:
   3
   abc abcc abd

   Output:

   3

2. Input:
   7
   aaa bbb abc bjyx wegdbs szdszd aaa

   Output:

   6

3. Input:
   5
   abc lgieong lgieong kkkk lgieong

   Output:

   3