Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Fall 2016-2017

Homework 3–Stack and Queue

Due: 24/10/2016, 13:30
(Late submission penalty: -20%)

PLEASE NOTE:
Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!

Introduction
The aim of this homework is to get familiar with stacks and queues. You are asked to read a sequence of characters (only a-zA-Z, i.e., the ones in the alphabet) from standard input and determine whether they are in the requested format or not. The details will be explained in subsequent sections of this homework.

The datastructures
In this homework, you ***must*** use ***dynamic*** **stack** or ***dynamic*** **queue** to determine the correctness of sequences. The size of stack and queue must increase dynamically. The decision to use which structure (stack or queue) to implement is depends on the format of sequence. You are not allowed to use any other data structure, i.e., **no arrays and vectors**.You can use the same linked list code we introduced in the lectures, or the ones we used in the recitations as the base implementation.

The desired formats for the sequences:
There will be 2 formats of the sequences containing lower and upper case letters (**all the inputs will be valid letters; you don't need to check the input in this homework**). In both formats, the number of occurrences of a lower case letter, e.g., 'a', will be equal to the number of corresponding upper case **char**, e.g., 'A':

1. In the first format, each lower case character need to be matched with one of its corresponding upper case characters in the sequence (and vice versa). If a lower case **char** appears earlier than another lower case **char**, the former's matching upper case **char** should appear later than the latter's matching **char**. That is if there is "ab" in the

sequence, the **char**‘B’ must come before ‘A’ and after “ab”. Here are two examples of correct format:

abbcCghhdDHHGBBA
abBAcCdefgGgGggGGFED

2. The second format contains multiple consecutive sub-sequences of lower case and upper case characters; e.g., **abcABC**. Furthermore, the sub-sequences need to contain the same characters (lower and upper) in the same order as in the previous example **abcABC**. Multiple sub-sequences can be used to form the whole sequence. Here are two examples of this format:

aaaaAAAAabABaAbB
abcaABCAdDbaBA

Program flow and output

Your program should start with a prompt for the type of the sequence format (**1** for first sequence and **2** for second sequence format); y**our program must handle the cases where user enters an incorrect format**. When a valid type is given, the program should prompt for sequence string. According to which sequence format is chosen, the program should check the correctness of the sequence and display result of check in the output.The program should prompt for new sequence format and sequence again until user enter **Ctrl+Z**.

Sample runs

Below, we provide some sample runs of the program that you will develop. The italic and bold phrases are inputs taken from the user. You should follow the input order in these examples.

```
Please choose which sequence format are you going to search ? First or Second: sdf
You entered invalid input
Please choose which sequence format are you going to search ? First or Second: rew
You entered invalid input
Please choose which sequence format are you going to search ? First or Second: 1
Enter a sequence: abbcCghhdDHHGBBA
The input is in correct format
Please choose which sequence format are you going to search ? First or Second: 2
Enter a sequence: aaaaAAAAabABaAbB
The input is in correct format
Please choose which sequence format are you going to search ? First or Second: 1
Enter a sequence: abBAcCdefgGgGggGGFED
The input is in correct format
Please choose which sequence format are you going to search ? First or Second: 2
Enter a sequence: abcaABCAdDbaBA
The input is in correct format
Please choose which sequence format are you going to search ? First or Second: 1
Enter a sequence: aabcCABdDA
invalid format
Please choose which sequence format are you going to search ? First or Second: 1
Enter a sequence: abBAcCdefgGgGggGGFED
The input is in correct format
Please choose which sequence format are you going to search ? First or Second: 1
Enter a sequence: aaaaAAAAaAbBabAB
```

```
invalid format
Please choose which sequence format are you going to search ? First or Second: 2
Enter a sequence: aaAAbccaBCA
invalid format
Please choose which sequence you are going to search ? First or Second: ^Z
Enter a sequence: Press any key to continue . . .
```

Some Important Rules:
In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**.

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Name your cpp file that contains your program as follows:

"SUCourseUserName_YourLastname_YourName_HWnumber.cpp"

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw2.cpp

Do not add any other character or phrase to the file name. Make sure that this file is the latest version of your homework program. Compress this cpp file using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are**NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
CS204 Team (Leyli Javid Khayati, Kamer Kaya)