**Sabancı University**

Faculty of Engineering and Natural Sciences

**CS204 Advanced Programming**

**Fall 2018-2019**

**Homework 3 – Stacks & Queues**

**Due: 17/10/2018 - 23:55**

(One day late submission penalty: -10%)

|  |
| --- |
| **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!** |

1. **Introduction**

The aim of this homework is to practice stack and queue operations. There will be two XML files containing student and lecture information. For a detail explanation of XML files, refer to <https://www.w3schools.com/xml/xml_whatis.asp>

The goal of the program is assigning courses to students according to their preferences, which are listed in the XML file. The assignment scheme will be using a Round-Robin strategy. Also, the XML file for the lectures will contain capacity information. When the capacity of a course is reached, the remaining students will fail to take that course. The details of the homework is described in the document.

1. **Data Structure**

In this homework you need to implement a stack for parsing the XML files. Since XML files are designed to be nested, using a stack is necessary for parsing information. You can refer to online sources to **get an idea** about how to parse an XML file. However, codes taken from online platforms will be punished. *Also, you should print out an error message if the XML format is invalid. (If the opening and closing tags are not matching)*

Second, the list of courses desired by a student will be held in a singly linked-list. You can use (and modify if you want) the struct definitions below to track **Student** information. Note that, using these structs is not mandatory and you can implement your own structs as well.

struct lectureNode{

string lectureName;

lectureNode \*next;

lectureNode()

{

lectureName = "";

next = NULL;

}

lectureNode(string name, lectureNode\* ptr=NULL)

{

lectureName = name;

next = ptr;

}

};

struct Student{

string name;

int id;

lectureNode \*lectures;

Student()

{

name = "";

id = 0;

lectures = NULL;

}

Student(string n, int i, lectureNode \*ptr = NULL)

{

name = n;

id = i;

lectures = ptr;

}

};

Finally, you need to use Queue structure to implement a Round-Robin course assignment system. In round-robin schemes the items are listed in a queue. The items are *dequeued* one-by-one and *enququed again* **if their job is not finished yet.** In the case of course assignment, if a student object still has desired courses in its lectures list, it will be enqueued again.

1. **Program Flow**

The program will start by prompting the file names for student XML file and lecture XML file. In case. As per usual, the names will be prompted again and again until a correct name is entered.

When the files are successfully opened, first the student XML file will be parsed and each student in the file will be *enqueued* to a student queue. While parsing the student XML file, each lecture will be added **at the end of the** lectures list inside the student node.

Once the student XML file is successfully parsed, the lectures XML file will be parsed and each lecture in the file will be added to a **linked-list** of lectures. Note that these lectures will also contain capacity information, so you might want to create another struct for these.

Finally, after both files are parsed and data structures are filled with necessary information, the assignment step will begin. Students will be assigned to courses in their lectures lists **starting from the head node**. If the capacity of a course is reached, the student won’t be assigned to that course and an appropriate message will be printed. Below you can find a sample run for the files given in the attachment.

1. **Sample Runs**

Please enter the name of the Student XML file: **asd.txt**

*Invalid file.*

Please enter the name of the Student XML file: **inputStu.txt**

Please enter the name of the Lectures XML file: **input.txt**

Invalid file

Please enter the name of the Lectures XML file: **inputLec.txt**

*CS201 is assigned to Mustafa Kemal Tas(16755)*

*CS204 is assigned to Kamer Kaya(18961)*

*CS204 is assigned to Mustafa Kemal Tas(16755)*

*CS408 is assigned to Kamer Kaya(18961)*

*CS303 is assigned to Mustafa Kemal Tas(16755)*

*CS303 can not be assigned to Kamer Kaya(18961)*

*CS517 can not be assigned to Kamer Kaya(18961)*

Press any key to continue . . .

Please enter the name of the Student XML file: **wrongStu.txt**

Please enter the name of the Lectures XML file: **inputLec.txt**

Invalid XML format!.. Exiting.

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

***Hw2\_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 (Mustafa Kemal Taş, Kamer Kaya)