Programming Assignment III

DUE: Sunday, April 27, 2014, 23:00. [Hard deadline. Ninova HW submission closes automatically. Late homeworks will NOT be accepted.]

NOTE: If you have any questions about the homework, contact the research assistant, Çağatay Koç, by email (kocca@itu.edu.tr) or in person (in Research Lab 3, EEB 4313).

NINOVA SUBMISSION: You should be aware that the Ninova system clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructor with your submission after the Ninova site submission has closed. If you have submitted to Ninova once and want to make any changes to your code, you should do so before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems to the Internet or to Ninova in the last few minutes are not valid excuses for being unable to submit. You should not risk leaving your submission to the last few minutes.

CHEATING: This is not a group assignment. It should be done individually. When a student receives information from another person about a program, it is considered cheating when the information is enough to precisely describe the code in a nontrivial part of the program. The most common example of cheating occurs when a student copies all or part of a program from another student and then changes the names of some of the program variables and functions. If cheating is discovered, a report will be made recommending a course grade of "VF".

Assignment Description

This assignment consists of **two parts:** (1) a prefix expression evaluator that uses a stack implemented on a doubly linked list and (2) a task priority queue implemented on a linked list. **You are required to submit both parts.**

1) Implement a **prefix expression evaluator** using a stack data structure. Your program should read expressions line-by-line from the input file and write the results to the output file. You are provided with an example input file, **"input1.txt",** and the expected output of this input file, **"output1.txt"**. In the table below, the contents of these files are given.

Sample Input File	Expected Output File
+ * 11 2 5 * + 4 3 3 - / 12 3 2 * 2 + 13 - 4 2	27 21 2 30

- The program should read the expressions from the input file (**"input1.txt"**), evaluate them, and write the result to the output file (**"output1.txt"**).
- You should implement your own **stack structure** with a **doubly linked list**.
- You should successfully **deallocate** all of the allocated memory before termination of your program.

- Make sure that **GNU C++ compiler (g++)** compiles your project and the program runs smoothly in Linux. You can use **ITU ssh server** to compile and test your application.

2) Implement a program that uses a **priority queue** to store tasks according to their priorities. The program should read commands from an input file and creates the appropriate output file. You are provided with an example input file, **"input2.txt",** and the expected output of this input file, **"output2.txt"**. In the table below, the contents of these files are given. Your program should read data from the input file (**"input2.txt"**) and interpret the given commands.

Sample Input File	Expected Output File
enqueue 8 A enqueue 5 B enqueue 10 C dequeue enqueue 9 D dequeue dequeue dequeue	C D A B

- The **priority queue** should be implemented with a **linked list**.
- The command **"enqueue priority task_name"** consists of an integer priority value and a task name. The **enqueue** operation should keep the linked list ordered by priority. The tasks with higher priority should dequeued first.
- The command **"dequeue"** should dequeue the task with the highest priority and write it to the output file (**"output2.txt"**).
- You should successfully **deallocate** all of the allocated memory before termination of your program.
- Make sure that **GNU C++ compiler (g++)** compiles your project, and the application runs in Linux smoothly. You can use **ITU ssh server** to compile and test your application.

SUBMISSION PROCEDURE:

1. The top lines of all the files in your project should be comments with the following information
   // Your Lastname, Your Firstname
   // Your Student ID
   // Programming Assignment 3
2. Write down the appropriate compiling and running codes to a readme.txt file (only if necessary).

3. Make sure that GNU C++ compiler (g++) compiles your project, and the application runs in Linux smoothly. This is important because we will evaluate your homework in Linux using g++.
4. After making sure that everything compiles smoothly, archive all files into a zip file.
5. Upload this file by **23:00 on Sunday, April 27, 2014** to Ninova. No late assignments will be accepted.

Your program will be run and graded on programming style and how it performs the required computation. With regard to programming style, we expect the following: (i) **comments and appropriate variable names** should be used to make your program more readable; (ii) appropriate prompts and messages for input and output should be given to the user of your program. In general, more emphasis will be placed on program clarity than on program speed or size.