BLG 233E, Data Structures and Laboratory, Fall 2012-2013
Assignment #2
Asst. Prof. Dr. Gülşen Cebiroğlu Eryiğit
Res. Asst. Doğan Altan

Due: Friday, November 30th, 23:00 [Hard deadline. Ninova HW submission closes automatically. Late homeworks will NOT be accepted.]

Required submission components: **1)** your source code file (.cpp) and **2)** a soft copy of your report (in pdf format).

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 instructors 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 report, 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. For the most part, oral discussions between students about the algorithms used in a program or the program's design, is not considered cheating. In fact, these types of discussions are encouraged. 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".

HOMEWORK DESCRIPTION

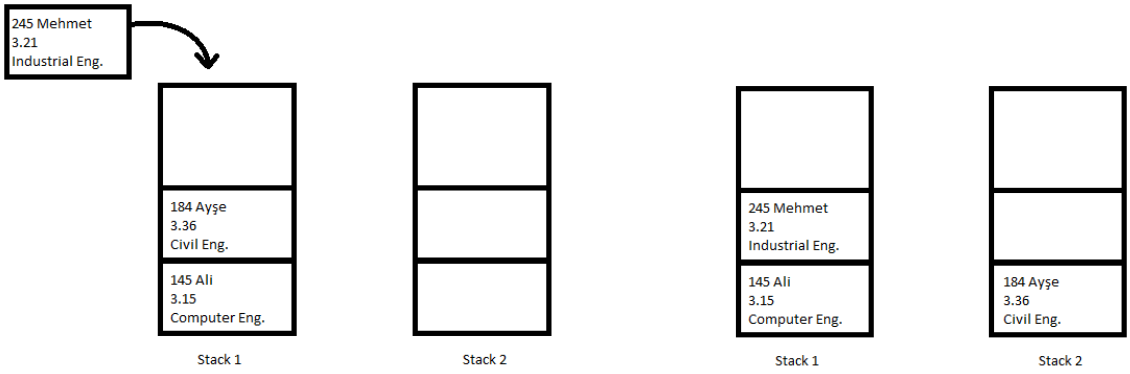
In this homework, you will implement a solution that uses stacks to sort student records according to GPAs in a university. An input file will be given which includes the necessary information for your struct (number, name, GPA, and department).

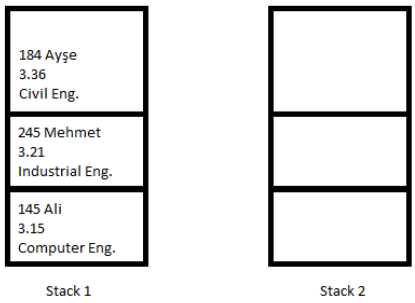
You need to use two stacks:

1) The program will add records in a sorted fashion to the first stack. The student record with the lowest GPA should be at the bottom of this stack, and the record with the highest GPA should be at the top.

2) You will use the second stack for temporarily storing records while adding a new record to the first stack.

When adding to the first stack, you should check the top record and pop the records from the first stack until the GPA of the top record is lower than that of the record to be added. You should push these popped records onto the second stack, and after pushing the record to the first stack, you should move the contents of the second stack to the first stack. For clarity, this situation is illustrated below.





(c)

Placing the record "Mehmet" on top of the stack would destroy the order, so we should first move the record "Ayşe" to Stack 2 because this record has a higher GPA. After adding "Mehmet" to Stack 1, "Ayşe" should be moved back to Stack 1.

At the end of the stack operations (when the stack is sorted), you should pop the records from Stack 1 and write them to the file "sortedRecords.txt."

You must also consider the situation where there are two students with the same GPA. Implement a solution for this situation, and explain how your algorithm deals with this situation in your report.

Your stack structs should look like the following:

```
struct Node
{
	/* your variables here */
	Node* next;
};

 struct Stack
{
	Node* top;
	/* your methods here */
};
```