Assignment 02– Student Record Management System

Average expected estimated time to complete this assignment is 12 hours

- [1 HOUR] software requirement understanding
    - reading of the assignment document
    - understanding of problem statement
    - understanding of software inputs and outputs
- [2 HOURS] software design
    - identification of user-software interaction sequences
        - software menus
        - user inputs and
        - software flows
    - identification of required variables and data structures to be used
        - for input, processing, and output
        - identification of storage class specifications of required variables
    - modularization of software into required functions
        - proper naming of functions
        - identification of tasks to be performed into functions
        - identification of parameters and return types of functions
    - identification of user defined header files
        - names and functions to be placed in
    - flow sequence of main function and call of various user defined functions from it
- [6 HOURS] software coding
    - coding of user defined functions
    - placement of user defined functions into header file/s
    - coding of main function
- [2 HOURS] software testing
    - running software on various inputs
    - identification of software error and bugs
    - removal of software error and bugs
- [1 HOUR] software documentation
    - proper indentation of code
    - use of proper naming conventions
    - commenting of code

Objectives:

- This assignment will provide experience with usage of Singly and Doubly linked lists as the storage linear structure for Student Data.

Prerequisite Skills:

- Singly LinkedList
- Doubly LinkedList

Problem Statement:

Instructions:

In this Assignment a Student Record Management System be implemented. We will write code to manage the storage and organization of student data in a combination of doubly and singly linked lists. First of all, you will have to implement a doubly linked list which will contain the data about the students. Each node of this linked list will have the following structure.

```
struct StudentNode
{
      int rollNo;                // Roll number of the Student
      StudentData data;   // Rest of the Data (see below)
      StudentNode* prev;   // Link to previous node
      StudentNode* next;  // Link to the next node
};
```

The Student data (**StudentData)** is represented as a structure which contains two fields (name and a linked list of courses along with grades)

```
struct StudentData
{
      string name;   // name of the student
      CourseNode* head;  // Pointing to the head of courses/grades list

};
```

In above declaration 'head' is a pointer which will be pointing to the first node of the list of the courses. This list will contain course numbers (nodes) along with grades of a particular student.

```
struct  CourseNode
{
      string courseNumber;   // Number of the course e.g. IT101
      string grade;          // Grade in the course e.g. B-
      CourseNode* next;     // Link to the next node
};
```

The list of students (**StudentList**) will be a ***circular doubly linked list*** (with a dummy header node). It will have the following declaration. **This linked list should sort in increasing order of the student roll numbers.**

```
class StudentList
{
private:
	StudentNode head;
public:
	// methods
};
```

The doubly linked list (**StudentList)** should provide the following functions:

```
StudentList();
// constructor that will create an empty StudentList

~StudentList();
// Destructor to destroy (deallocate) StudentList

bool addStudent(int rollNo, const char* name);
// Adding a student record into the StudentList. Roll No must be unique. Function should
return false if
// roll number already exits

bool removeStudent(int rollNo);
// Remove the record of a particular student from the StudentList.Fucnction will return
true if student
// is removed successfully.


void updateStudent(int rollNo, const char* newCourse, const char* grade);
// update the record of a particular student (specified by rollNo) by adding the newCourse
(along with its grades) in the
// courses list. Insert new course at head



void displayStudent(int rollNo);
// Displaying the record of a particular student on the screen

void displayAllStudents(int order);
// Display the records of all the Students. if order=0, then this function will display
the student
// records in ascending order of the roll numbers,and if order=1 , then this function will
display the student records in descending
// order of the roll numbers
```

File(s) to be submitted:

The proper working solution of the Student Record management system. There should not be memory leak, dandling pointers, null pointer assignment errors or an Illegal memory access Exceptions. Try to handle all the exceptions and possible run time errors in proper way. You Driver program should be menu based allowing to call all above methods options and also exit the program option.

Deadline: November 19, 2023. (11:59pm)