Data Structures and Algorithms Lab

Lab 09 **Marks 10**

Instructions

- Work in this lab individually. Follow the best coding practices and include comments to explain the logic where necessary.
- You can use your books, notes, handouts, etc. but you are not allowed to borrow anything from your peer student.
- Do not use any **AI** tool for help; doing so will be considered cheating and may result in lab cancellation and possible disciplinary action.
- Test your program thoroughly with various inputs to ensure proper functionality and error handling.
- Show your work to the instructor before leaving the lab to get some or full credit.

Student Management System with Linked List Operations

You are required to implement the **StudentList** class, which stores students in **unsorted order**. Your class declarations should look like this:

```
class Student
{
    friend class StudentList;

private:
    int id;             // ID of the student
    string name;        // Name of the student
    float cgpa;         // CGPA of the student
    Student* next;      // Pointer to the next node in the list

public:
    Student(int id, string name, float cgpa, Student* next = nullptr);    // Constructor
    void studentDetails() const;     // Displays the student information
};

class StudentList
{
private:
    Student* head;      // Start of the list
    Student* cursor;    // Current item in the list

public:
    StudentList();      // Constructor
    ~StudentList();     // Destructor

    // Member functions
    void insert(const Student& newStd);    // Inserts a new student
    void remove(int id);                   // Removes a student by ID
    void search(float cgpa) const;         // Searches for students by CGPA
    void replace(const Student& newStd);   // Replaces or appends a student
    bool isEmpty() const;                  // Checks if the list is empty
    void gotoBeginning();                  // Moves cursor to the beginning
    void gotoEnd();                        // Moves cursor to the end
    bool gotoNext();                       // Moves cursor to the next item
    bool gotoPrior();                      // Moves cursor to the previous item
    Student getCursor() const;             // Returns the student at cursor
    void showStructure() const;            // Displays the list structure
};
```

Specifications for Member Functions

Each member function should follow the given behavior:

1. **Insert**
   - Inserts **newStd** into the list. If the list is not empty, insert **newStd** after the cursor. Otherwise, insert **newStd** as the first item. Moves the cursor to **newStd**.
   - **Time Complexity**: $O(1)$

2. **Remove**
   - Removes the student with the given **id**. If the cursor points to the removed student, move the cursor to the next student. If the removed student was at the end, move the cursor to the beginning.
   - **Time Complexity**: $O(N)$
3. **Search**
   - Searches for student(s) with the specified **cgpa** and displays their details. If no match is found, display an appropriate message.
   - **Time Complexity**: $O(N)$
4. **Replace**
   - Replaces a student based on **id**. If no matching **id** exists, appends **newStd** at the end. Cursor points to **newStd** after the operation.
   - **Time Complexity**: $O(N)$ (search for id) or $O(1)$ (if appending at the end)
5. **isEmpty**
   - Checks if the list is empty.
   - **Time Complexity**: $O(1)$
6. **gotoBeginning**
   - Moves the cursor to the first student in the list.
   - **Time Complexity**: $O(1)$
7. **gotoEnd**
   - Moves the cursor to the last student in the list.
   - **Time Complexity**: $O(N)$
8. **gotoNext**
   - Moves the cursor to the next student if it is not at the end. Returns true if successful, otherwise false.
   - **Time Complexity**: $O(1)$
9. **gotoPrior**
   - Moves the cursor to the previous student if it is not at the beginning. Returns true if successful, otherwise false.
   - **Time Complexity**: $O(N)$ (since the list is singly linked and requires traversal)
10. **getCursor**
    - Returns the student object currently pointed to by the cursor.
    - **Time Complexity**: $O(1)$
11. **showStructure**
    - Displays all students in the list. If the list is empty, outputs "Empty list".
    - **Time Complexity**: $O(N)$

Input File Specifications

Your program should take student data from a text file named **input.txt**. The file format is as follows:

```
<id>
<name>
<cgpa>

<id>
<name>
<cgpa>

...
```

Each student's data is separated by a blank line.

Main Function

- Read data from the input file and populate the **StudentList**.
- Demonstrate the functionality of all member functions with appropriate test cases.