Lab 4 – Linked Lists

Implementing Lists using Linked Lists

Task 1. The `linkedList` template:

Implement abstract data type list using singly linked lists called `linkedList`. Functionalities desired are as follows:

Function	Description
Constructors	Decide if you need to use any parameters. A list is empty when it is initialized
Destructors	Required since you use dynamic memory management
`bool isEmpty() const`	Checks if list is empty
`bool isFull() const`	Checks if list is full
`int listSize() const`	Returns the size of the list
`int maxListSize() const`	Returns the maximum possible size of the list.
`void print()`	Prints the elements of the list on the console
`bool isItemAtEqual(int, elemType)`	Checks if the item at position matches the 2nd parameter. You may need to check that the position is smaller than the size of the list.
`void insertAt(int, elemType)`	Inserts 2nd parameter at position specified in the 1st parameter. You may need to check that the position is smaller than the size of the list.
`void insertEnd(elemType)`	Inserts object to end of the list
`void removeAt(int)`	Removes object at position. You may need to check that the position is smaller than the size of the list.
`int retreiveAt(int)`	Retrieves object at position. You may need to check that the position is smaller than the size of the list.
`void replaceAt(int, elemType)`	Replaces object at position with 2nd parameter. You may need to check that the position is smaller than the size of the list.
`void clearList()`	Empties the list
`operator=`	Overload the assignment operator.

Here, `elemType` is the type of the members of the list. In a given list, all elements are of the same type. You should use template implementation to enable functionality to have lists storing different types of objects dynamically.

Task 2. The `sortedLinkedList` template:

Sorted lists are a special type of list that stores elements in a sorted order, i.e., each element should always be less than its next element and smaller than its previous element.

Implement abstract data type list using singly linked lists called `sortedLinkedList`. Functionalities desired are as follows:

Function	Description
Constructors	---
Destructors	---
`bool isEmpty() const`	Checks if list is empty
`bool isFull() const`	Checks if list is full
`int listSize() const`	Returns the size of the list
`int maxListSize() const`	Returns the maximum possible size of the list
`void print()`	Prints the elements of the list on the console
`void insert(elemType)`	Inserts element into the sorted linked list. Note that the position to insert is not provided. All the elements in the list are required to be sorted in ascending order.
`void remove(elemType)`	Removes an element from the sorted linked list
`bool isItemInList(elemType)`	Returns true if the 1st parameter is in the list
`elemType retreiveAt(int)`	Retrieves object in the position of the 1st parameter
`void clearList()`	Empties the list
`operator=`	Overload the assignment operator

Here, `elemType` is the type of the members of the list. In a given list, all elements are of the same type. You should use template implementation to enable functionality to have lists storing different types of objects dynamically. You may reuse your code from Task 1.

Specifications

Below is a breakdown of the tasks labeled [LP] and [HP]. If you complete ALL the LP components satisfactorily, you will receive a grade of "low pass" on the lab. If you complete ALL the LP components and the HP components mentioned below satisfactorily, you will receive a grade of "high pass":

- [LP] Task 1 functionalities implemented as a template using dynamic memory management correctly.
- [LP] All task 1 implemented functionalities are tested in main.
- [HP] Complete the following:
  - Task 2 functionalities implemented as a template using dynamic memory management correctly.
  - [Optional] Detail the operators that should be overloaded in the class that is passed as elemType, the template parameter.

If you do not meet the criteria for a "low pass", the submission will be marked as "revision needed".

What to submit:

Your final submission will need to have the files as follows:

- `linkedList.h`
- `sortedLinkedList.h (if you complete Task 2)`
- `Notes (if you complete the optional HP task)`
- `lab4-cmpe126.cpp`

NOTE: You can look for help on the Internet but refrain from referencing too much. Please cite all your sources in your `Notes` file.

When to submit:

Submit your lab before **Thursday, March 14th, 11:59pm**. You are strongly advised to submit before Friday, March 1st, 11:59pm.

When you submit your assignment, you automatically agree to the following statement. If you do not agree, it is your responsibility to provide the reason.

"I affirm that I have neither given nor received unauthorized help in completing this homework. I am not aware of others receiving such help. I have cited all the sources in the solution file."