Lab 3 – The arrayList class (modified 2/22)

Implementing Lists using Arrays

Task 1. The `arrayList` template:

Implement abstract data type list using arrays called `arrayList`. 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	Especially required if 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. If you choose to have no upper limit, mention it in a comment
`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. You can choose to also copy over the maxListSize parameter, or only copy the elements over if possible.

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.

Specifications

Below is a breakdown of task 1 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] Implemented array constructors using a default size of the array.

- [LP] Basic functions
    - isEmpty
    - isFull
    - listSize
    - maxListSize
    - print
    - isItemAtEqual
- [HP] (6/7 functions) OR (4/7 functions, dynamic memory management, and template implementation)
    - 1. insertAt
    - 2. insertEnd
    - 3. removeAt
    - 4. retrieveAt
    - 5. replaceAt
    - 6. clearList
    - 7. operator=
    - Used dynamic memory management in the constructor and destructors correctly.
    - Implemented template correctly
- [LP] All implemented functionalities are tested in main

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:

- `arrayList.h`
- `arrayList.cpp (if not using templates)`
- `lab3-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, February 29th, 11:59pm**. You are strongly advised to submit before Friday, February 23rd, 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."