Foundation for Advancement of Science and Technology

National University of Computer & Emerging Sciences (NUCES),

School of Computing

Data Structures

Fall 2021

Lab 04

Learning outcomes:

In this lab you are expected to learn the following:

• Linked list using arrays
• String matching
• Regex
• Google testing

Task 1: *70 pts*

Design an application for **List** class using templates, with following data members:

- A pointer of type T
- A variable *capacity* of type int to hold the total capacity of the list
- A variable *counter* of type int which holds the current index of the list

Your class should be able to perform following operations:

- **insert(T item)**

  Inserts an item and increments the counter only if the list is not full

- **insertAt(T item, int index)**

  The function takes two parameters i.e., item of type T and an integer parameter which points to the position where the insertion is to be done. Make sure you do not overwrite values. For this you will have to increment the counter to create additional space for the item to be inserted.

- **insertAfter(T itemTobeInserted, T item)**

  The first parameter of this function represents the item to be inserted in the list, the second parameter represents the item already present in the list. You must search the item already present in the list and then insert the **itemtobeinserted** after **item**. Use the functions of your own code instead of replicating logic.

- **insertBefore(T itemTobeInserted, int item)**

  The first parameter of this function represents the item to be inserted in the list, the second parameter represents the item already present in the list. You must search the item already present in the list and then insert the **itemtobeinserted** before **item**. Use the functions of your own code instead of replicating logic, i.e.

- **isEmpty()**

  Checks if the *counter* variable is zero, i.e., the list is empty there are no elements.

- **isFull()**

  Checks if the *counter* variable has reached the total capacity of the **List**, which means no more insertions are possible.

- **remove(T item)**

Removes item passed through parameters. Make sure you update the *counter* after removing one item.

- **removeBefore(T item)**

You pass an item through parameter, then search for it in the array, then remove the item present before this item. i.e. if I have {1, 2, 3, 4} in array. I pass 3 to this function the resultant array should look like {1,3,4}. Make sure you update *counter* after removing one item.

- **removeAfter(T item)**

You pass an item through parameter, then search for it in the array, then remove the item present after this item. i.e. if I have {1, 2, 3, 4, 5} in array. I pass 3 to this function the resultant array should look like {1,2, 3, 5}. Make sure you update *counter* after removing one item.

- **search(T item)**

This function searches for the item passed through parameter. If item exists, return index number. But if you didn't get the item in the list, what are you going to return?

- **print()**

Print all the items of the List.

- **operator==(List& L)**

we can overload == to compare the items of two lists. If the lengths of both the lists aren't same, you don't have to go further to check for the items, i.e., the lists are already unequal.

- **reverse()**

Reverse all the elements of the List.

✗ Don't forget to allocate memory to the array through constructor.

Foundation for Advancement of Science and Technology FAST

National University of Computer & Emerging Sciences (NUCES), School of Computing

Task 2 **10+50pts**

You are hired to work on input validation of an application.

a. Your 1st task is to write a C++ code for password validation. A valid password must satisfy following conditions:
   1. Length of password should be more than 5 and less than 11
   2. Password must contain at least 1 digit, 1 uppercase Letter and 1 special character

b. Your 2nd task is to write a C++ code for email validation. A valid email must satisfy following conditions:
   1. Email must have the given format "prefix@domain.com".
   2. Domain must be a valid domain (must match with any domain in domain dataset).
   3. 1st char of prefix must be an alphabet
   4. prefix can have only 3 special characters (- , . , _)
   5. 2 consecutive special characters in prefix are not allowed
   6. Last character of a prefix can not be a special character