# Tasks

**Question 1**

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

Private:

- 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

Public:

- A pointer of type T (T* values)

Your class should be able to perform the following operations:

- **bool insert(T item)**

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

- **bool 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.

- **bool 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.

- **bool 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.

- **bool isEmpty()**

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

- **bool isFull()**

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

- **bool remove(T item)**

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

- **bool 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.

- **bool 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.

- **int search(T item)**

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

- **void print()**

  Print all the items of the List.

- **bool 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.

- **void reverse()**

  Reverse all the elements of the List.

**Question 2**

You have to make a class:

class DynamicList {

private:

int size;

public:

int *arr;

//constructor that initializes the array using DMA

}

With the following member functions:

1. **void addEnd(int val)**
   Insert an Element at end of List

2. **void addStart(int val)**
   Insert an Element at start of list

3. **void addGivenP(int val, int pos)**
   Insert an Element at given position

4. **void delStart()**
   Remove an Element from the start of the list

5. **void delEnd()**
   Remove an Element from the end of the list

6. **void delGivenP(int pos)**
   Remove an Element from a given position

7. **int Next(int pos)**
   NEXT (p, L): Return the position following p on list L.

8. **int Previous(int pos)**
   Return the position preceding position p on list L
9. **void print()**
   Printing the List

10. **bool Isempty()**
    Returns True if List is empty else returns False

11. **void replace(int ind, int val)**

Replace (on the basis of index and value)
Replace any element of the List

**12. void clear()**

Deletes all the elements from the List

**13. Duplicate Elements**

Search for duplicate elements in list if duplicate Elements exist return True else return False