OOP LAB 15: Templates and STL

Question#01

Define a template class **Container** for storing different type of values e.g. int, double, float etc. This container would keep same type of values/elements at a time. The **Container** class should consist of the following member variables:

> **T * values;** A pointer to set of values it contains. This member points to the dynamically allocated array (which you will be allocating in constructor).
> **capacity;** An integer that shows total capacity of the container
> **counter;** An integer counter variable which increments upon every insertion operation and decrements upon removal of any element; representing total number of elements currently present in the container.

The container should consist of following functions:

- **constructor** with capacity as parameter
- **isFull** which will return true if counter reaches capacity otherwise false
- **insert** which will receive a value of some type (int/double/float) and insert into the container if it's not already full
- **search** which will return true if the container contains value passed as parameter
- **remove** which will remove a value, if exists, from the container and shifts all subsequent values one index back.
- **print** which will print all values contained in the container

Now create a test class to create three instances of Container for types: **int, float, double** and call all functions separately for each instance. Your output must be properly formatted.

Question#02

In this task you're required to extend the task#01 a little. Since in task #01 you created separate instances of Container (for int, double and float) and for each instance you called all functions. In this way the Line Of Code exceeded and duplicate code increased as well. Let's experience reusability feature of OOP. Create a template function before main function.

```
 void testContainer(Container<T> &thisContainer, T valueToBeInserted, T
increment, T valueToSearch, T valueToRemove)
```

The purpose of this function is to call each function of Container class such as: it would insert the value `valueToBeInserted` in the container at first then keeps on inserting value until it reaches its capacity. Each new insertion will be incremented by `increment.` Then print the existing values, search for `valueToSearch,` remove the value `valueToRemove and finally print the existing values after removal.` Now inside main function you just need to create a new instance of some type and call **testContainer** template function by passing all required arguments. The rest will be handled by template function.

Question#03

The STL **Containers** provide extended functionality. **Vector** is a simplest Container that is an array in actual, but with some additional features. **Iterators** are used to iterate the elements in container efficiently.

Create an int type vector **v1** and **push** 10 consecutive perfect squares in it using for loop. Create another vector **v2** and initialize it with elements of v1 in reverse order by using reverse iterators **rbegin** and **rend**. Now display the elements of both containers using **copy** function in STL separately for both. But before that initialize another iterator **ostream_iterator** to format the output. STL provides another powerful feature of **find** to locate an element in the container. Use this **find** function to locate 49 in your vector v1 and display index of the located element in the vector. Compare both vectors for equality using **equal** algorithm provided by STL and display the information whether both are equal or not. Now replace all elements with the value 16 in vector v1 with 100. For that you need to use **replace** algorithm. **Display** the final output for elements of the vector v1 after modification.

Note: Include iterator, algorithm and vector

Question#04

In this task you're required to implement an **integer type stack** in STL. Create a **menu** to manipulate stack with different functions. Get user choice for menu and based upon the input perform the corresponding function on stack. The Menu items and corresponding functions should be:

```
1. Insert Element into the Stack
2. Delete Element from the Stack
3. Display Size of the Stack
4. Display Top Element of the Stack
5. Exit
```

Any other input will be treated as wrong choice. Don't forget to **display** menu to the user first.

Note: In addition to above mentioned libraries include stack as well.

Good Luck!