Lab Assignment 4
(Week 2 – Lab B & Lab C)

Q1. Without running the code, analyze the correctness of following programs. If program is correct, obtain the output.

(a)
```
#include <iostream>
using namespace std;

template <class T>
void f1(T t1)
{
   static T x = t1;
   cout<<x;
}

int main()
{
   f1(5);
   return 0;
}
```

(b)
```
#include <iostream>
using namespace std;

template <class T>
class abc
{
   static T x;
public:
   void f1(T t1)
   {
      x = t1;
      cout<<x;
   }
};
int main()
{
   abc <int> o1;
   o1.f1(5);
   return 0;
}
```

(c)
```
#include <iostream>
using namespace std;

template <class T>
class abc
{
   static T x;
public:
   void f1(T t1)
   {
      x = t1;
      cout<<x;
   }
};
int abc::x;
int main()
{
   abc <int> o1;
   o1.f1(5);
   return 0;
}
```

(d)
```
#include <iostream>
using namespace std;

template <class T>
class abc
{
   static int x;
public:
   void f1(T t1)
   {
      x = t1;
      cout<<x;
   }
};
template <class T>
int abc<T>::x;
int main()
{
   abc <int> o1;
   o1.f1(5);
   return 0;
}
```

Q2. Implement a template class Sort capable to sort an array (int, float, or char type) using following sorting techniques: Selection sort, Bubble sort, Merge sort, Quick sort (recursive), Radix sort, and Bucket sort.

i.e. if an object O is defined as Sort <int> O and calls a member function as O.QuickSort(A), where A is an integer array then it will sort the array A using Quick sort and returns the sorted array

Q3. Vector is one of the often used containers (STL) in C++ and briefly defined as follows: Vectors are used when dynamic size arrays are needed. Like arrays, vector elements are placed in contiguous storage location so they can be accessed and traversed using iterators. To traverse the vector we need the position of the first and last element in the vector. Considering the various functionalities of Vector, create your own template class **MyVector** with data members and member functions such that following functionalities of Vector can also be performed by MyVector.

size():Returns the number of elements in the vector
resize(n): Resizes the container so that it contains 'n' elements
max_size(): Returns the maximum number of elements that the vector can hold
push_back(x): Stores/pushes the element, 'x' into a vector from the back
pop_back(): pop or remove the element from a vector from the back
assign(x, t): Assign new value as 'x' at the container specified position/index 't'
erase(t): Remove the element from the container specified position/index 't'
clear(): Remove all the elements of the vector container

Q4. Set is one of the often used containers (STL) in C++ and briefly defined as follows: Sets are containers which store only unique values and permit easy look ups. The values in the sets are stored in some specific order (like ascending or descending). Elements can only be inserted or deleted, but cannot be modified. We can access and traverse set elements using iterators just like vectors. Some of the important functionalities of Set are as follows:

begin(): Returns an iterator to the first element of the set. Operational time is O (1).

clear(): Deletes all the elements in the set and the set will be empty. Operational time is O (N).

empty(): Returns true if the set is empty otherwise false. Operational time is O (1).

end(): Returns an iterator pointing at next to the last element. Operational time is O (1).

erase(): Deletes a particular element or a range of elements from the set.

find(x): Searches for the element, 'x' and returns the iterator pointing to the element, if the element is found, otherwise it will return the iterator returned by end(). Operational time is O (log N).

insert(x): Inserts a new element, 'x'. Operational time is O(log N) where N is the size of the set.

size(): Returns the size of the set or the number of elements in the set. Operational time is O (1).

****Note:** N is the size of the set

Considering the above functionalities of Set, create your own template class **MySet** and facilitate it with data members and member functions so that all the above mentioned functionalities of Set can also be performed by MySet.

Q5. List is one of the often used containers (STL) in C++ and briefly defined as follows: List is a sequence container which takes constant time in inserting and removing elements. List in STL is implemented as Doubly Link List. The elements from List cannot be directly accessed. For example to access element of a particular position, you have to iterate from a known position to that particular position. Considering the various functionalities of List, create your own template class **MyList** with data members and member functions such that all the functionalities of List can also be performed by MyList.

Q6. Stacks are a type of container adaptors where addition (push) and removal (pop) of elements are performed from one end only, i.e. in other sense its working is of the nature "Last in First Out" or LIFO. Implement your own template class for stack as **MyStack** which encapsulates the object of vector class (Vector template) and performs following operations:

push(x): Adds/insert the element 'x' at the top of the stack
pop(): Deletes the top most element of the stack
top(): Returns a reference to the top most element of the stack
size(): Returns the size of the stack