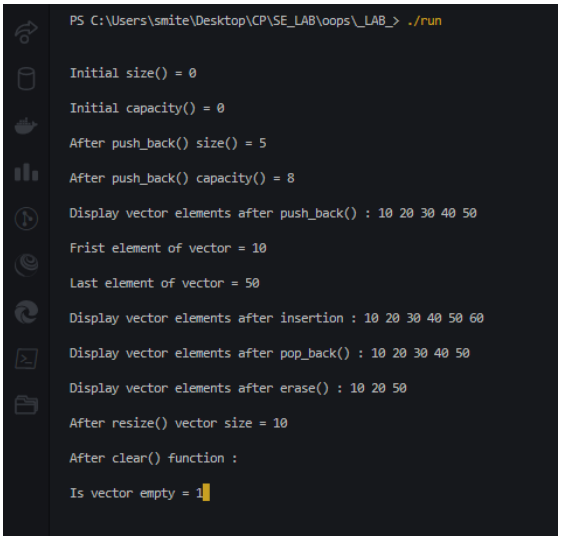
1)

```
// Vector Sequence Container
#include <iostream>
#include <conio.h>
#include <vector>
using namespace std;
void display(vector<int> &);
int main()
{
   vector<int> v;

   cout << "\n\nInitial size() = " << v.size();
   cout << "\n\nInitial capacity() = " <<
v.capacity();
   v.push_back(10);
   v.push_back(20);
   v.push_back(30);
   v.push_back(40);
   v.push_back(50);

   cout << "\n\nAfter push_back() size() = " <<
v.size();
   cout << "\n\nAfter push_back() capacity() = "
<< v.capacity();

   cout << "\n\nDisplay vector elements after
push_back() :";
   display(v);

   cout << "\n\nFrist element of vector = " <<
v.front();
   cout << "\n\nLast element of vector = " <<
v.back();

   vector<int>::iterator itr = v.begin();
   itr = itr + 5;
   v.insert(itr, 60);

   cout << "\n\nDisplay vector elements after
insertion :";
   display(v);


   v.pop_back();

   cout << "\n\nDisplay vector elements after
pop_back() :";
   display(v);


   v.erase(v.begin() + 2, v.begin() + 4);

   cout << "\n\nDisplay vector elements after
erase() :";
   display(v);
   v.resize(10);
   cout << "\n\nAfter resize() vector size = " <<
v.size();

   v.clear();

   cout << "\n\nAfter clear() function :";
   display(v);

   cout << "\n\nIs vector empty = " <<
v.empty();

   getch();
   return 0;
}
void display(vector<int> &v)
{
   for (int i = 0; i < v.size(); i++)
   {
      cout << " " << v.at(i);
   }
}
```

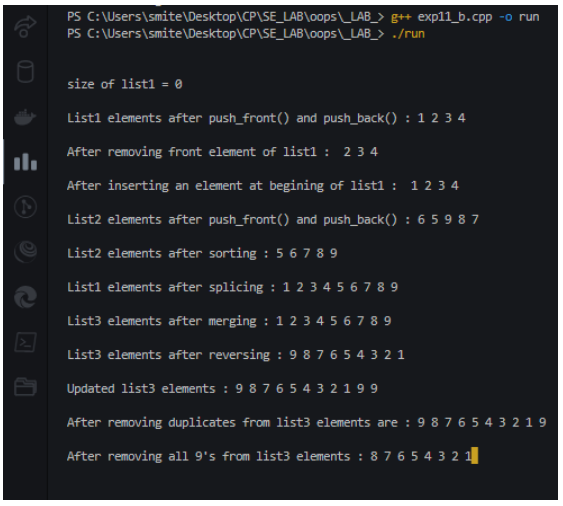
O/P



2)

```
// List sequence Container
#include <iostream>
#include <conio.h>
#include <list> //linear linked list
using namespace std;
void display(list<int> &); // display function
prototype
int main()
{
   list<int> list1; // empty list1 of zero length
   list<int> list2; // empty list2
   list<int> list3;

   cout << "\n\nsize of list1 = " << list1.size();

   list1.push_front(2);
   list1.push_front(1);
   list1.push_back(3);
   list1.push_back(4); // 1234

   cout << "\n\nList1 elements after
push_front() and push_back() :";
   display(list1);

   // Remove an element from front end
   list1.pop_front(); // same way pop_back()
   cout << "\n\nAfter removing front element of
list1 : ";
   display(list1); // 234

   // insert an element(1) at the begining of
list1
   list1.insert(list1.begin(), 1);
   cout << "\n\nAfter inserting an element at
begining of list1 : ";
   display(list1); // 1234

   // pushing elements in list2
   list2.push_front(5);
   list2.push_front(6);
   list2.push_back(9);
   list2.push_back(8);
   list2.push_back(7); // 65987

   cout << "\n\nList2 elements after
push_front() and push_back() :";
   display(list2);

   // sorting list2 elements
   list2.sort();

   cout << "\n\nList2 elements after sorting :";
   display(list2); // 56789

   // splice(insert) the elements of list2 at the
end of list1
   list1.splice(list1.end(), list2); // similarly
splice could be at begin() also

   cout << "\n\nList1 elements after splicing :";
   display(list1); // 123456789

   // merging list1 contents into list3
   list3.merge(list1);

   cout << "\n\nList3 elements after merging :";
   display(list3);

   // reverse a list
   list3.reverse();

   cout << "\n\nList3 elements after reversing
:";
   display(list3);

   list3.push_back(9);
   list3.push_back(9);

   cout << "\n\nUpdated list3 elements :";
   display(list3);

   // Removing duplicates from list3 elements
   list3.unique();

   cout << "\n\nAfter removing duplicates from
list3 elements are :";
   display(list3);
   // remove all 9's from list3
   list3.remove(9);

   cout << "\n\nAfter removing all 9's from list3
elements :";
   display(list3);
   getch();
   return 0;
}
void display(list<int> &v)
{
   list<int>::iterator p;
   for (p = v.begin(); p != v.end(); ++p)
      cout << " " << *p;
}
```

O/P :



3)
```
// Dequeu Sequence Container
#include <iostream>
#include <conio.h>
#include <deque>
using namespace std;
void display(deque<double> &); // display function prototype
int main()
{
   deque<double> d;

   // insert elements in d
   d.push_front(2.2);
   d.push_front(3.5);
   d.push_back(1.1); // 3.5 2.2 1.1

   cout << "\n\nDeque elements after insertion are as follows :";
   display(d);

   // pop_front() to remove front element
   d.pop_front(); // d.pop_back();

   cout << "\n\nDeque elements after pop_front() are as follows :";
   display(d);

   // using [] subsricpt operator to modify elements
   d[1] = 3.3; // 1.1 gets overwritten by 3.3

   cout << "\n\nDeque elements after subscript insertion using [] are as follows :";
   display(d);

   getch();
   return 0;
}
void display(deque<double> &d1)
{
   for (int i = 0; i < d1.size(); i++)
   {
      cout << " " << d1[i];
   }
}
```

```
PS C:\Users\smite\Desktop\CP\SE_LAB\oops\_LAB_> g++ exp11_c.cpp -o run
PS C:\Users\smite\Desktop\CP\SE_LAB\oops\_LAB_> ./run

Deque elements after insertion are as follows : 3.5 2.2 1.1

Deque elements after pop_front() are as follows : 2.2 1.1

Deque elements after subscript insertion using [] are as follows : 2.2 3.3
```

4)
```
// Standard Library queue adapter class
#include <iostream>
#include <conio.h>
#include <queue>
using namespace std;
int main()
{
   queue<double> q; // double type queue "q" created (initially empty)

   // push elements onto queue
   q.push(1.1); // 1.1  2.2  3.3
   q.push(2.2);
   q.push(3.3);

   cout << "\n\nPopping elements from queue : ";
   while (!q.empty())
   {
      cout << q.front() << " ";
      q.pop();
   }

   getch();
   return 0;
}
```

O/P :

```
PS C:\Users\smite\Desktop\CP\SE_LAB\oops\_LAB_> g++ exp11_d.cpp -o run
PS C:\Users\smite\Desktop\CP\SE_LAB\oops\_LAB_> ./run

Popping elements from queue : 1.1 2.2 3.3
```

5)
```
// Standard Library stack adapter class {can be implemented as vector,deque,list}
#include <iostream>
#include <conio.h>
#include <vector>
#include <list>
#include <deque>
#include <stack>
using namespace std;
template <class T>
void pushelement(T &s); // pushelement() function prototype
template <class T>
void popelement(T &s); // popelement() function prototype
int main()
{
   stack<int> dequestack; // stack with default underlying deque

   stack<int, vector<int>> vectorstack; // stack with underlying int vector

   stack<int, list<int>> liststack; // stack with underlying int list

   // push 10 elements on each of these stacks (i.e dequestack,vectorstack,liststack)
```

```
    cout << "\n\npushing elements onto
dequestack : ";
    pushelement(dequestack);
    cout << "\n\npushing elements onto
vectorstack : ";
    pushelement(vectorstack);
    cout << "\n\npushing elements onto liststack
: ";
    pushelement(liststack);

    cout <<
"\n\n-----------------------------------------------";

    cout << "\n\npopping element from
dequestack : ";
    popelement(dequestack);
    cout << "\n\npopping element from
vectorstack : ";
    popelement(vectorstack);
    cout << "\n\npopping element from liststack
: ";
    popelement(liststack);

    getch();
    return 0;
}
// Function definition for pushelement()
template <class T>
void pushelement(T &s)
{
    for (int i = 0; i < 10; i++)
    {
        s.push(i);
        cout << s.top() << " ";
    }
}
// Function definition for popelement()
template <class T>
void popelement(T &s)
{
    while (!s.empty())
    {
        cout << s.top() << " ";
        s.pop();
    }
}
```

O/P :