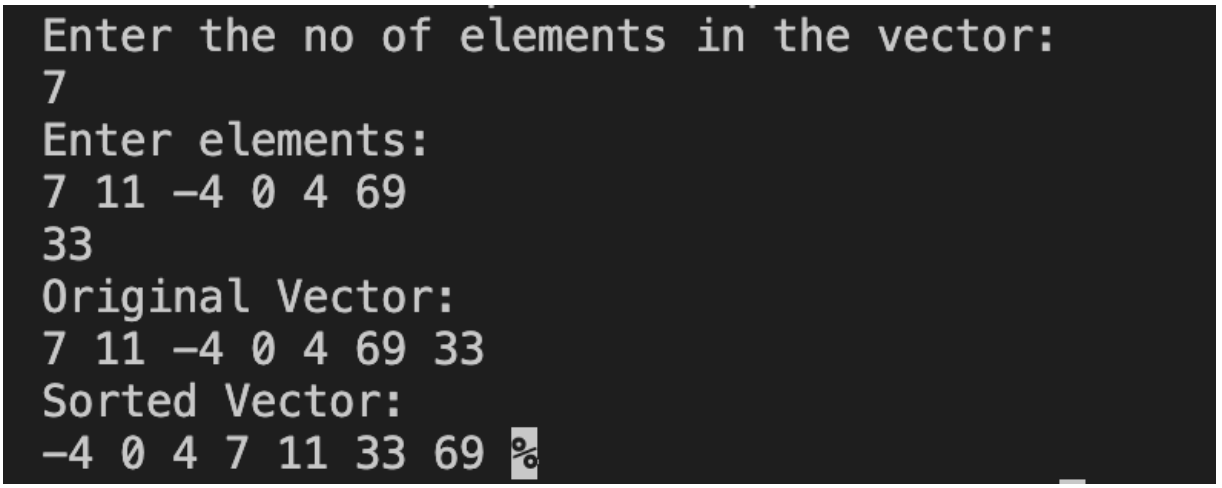
15B17CI371 – Data Structures Lab
ODD 2024
Week 6-LAB A
Practice Lab - STL

1. Use vectors to apply sorting to any array.

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
   vector<int>v1;
   cout<<"Enter the no of elements in the vector:\n";
   int n,k;
   cin>>n;
   cout<<"Enter elements:\n";
   for(int i=0;i<n;i++)
   {
       cin>>k;v1.push_back(k);
   }
   cout<<"Original Vector:\n";
   for(int i=0;i<n;i++)
   {
cout<<v1[i]<<" ";
   }
   sort(v1.begin(),v1.end());
```

```
    cout<<"\nSorted Vector:\n";
   for(int i=0;i<n;i++)
   {
cout<<v1[i]<<" ";
   }
   return 0;
}
```



2. Use STL to
   a. count the frequency of a particular value in a given array.
   b. erase a selected element in vector, shift and resizes the vector elements accordingly (after deletion of the selected element).
   c. erase duplicates in a given vector.
   d. find the distance between the first element and the maximum value within an array

```
#include <iostream>
#include<vector>
#include <algorithm>


using namespace std;
```

```
int main() {
//    int n,value;
//     n=4;
//     int arr[4];
//     cout<<"Enter  4 elements:"<<endl;
//     for(int i=0;i<4;i++)
//     {
//         cin>>arr[i];
//     }
//  cout<<"Enter element to get its frequency\n";
//  cin>>value;
//     int count = std::count(begin(arr), end(arr), value);


//     cout << "Frequency of " << value << " is: " << count << endl;


//     return 0;
// vector<int>v1;
//  cout<<"Enter the no of elements "<<endl;
//     int n,k,value;
//     cin>>n;
//     cout<<"Enter elements:"<<endl;
//     for(int i=0;i<n;i++)
//     {
//         cin>>k;v1.push_back(k);
//     }
//     cout<<"Original Vector:"<<endl;
//     for(int i=0;i<n;i++)
//     {
// cout<<v1[i]<<" ";
//     }
//     cout<<"\nEnter element to erase:\n";
//     cin>>value;
//     vector<int>::iterator it=find(v1.begin(),v1.end(),value);
//   v1.erase(it);
//    cout<<"Updated Vector:"<<endl;
//     for(int i=0;i<n-1;i++)
//     {
// cout<<v1[i]<<" ";
//     }
// vector<int>v1;
//  cout<<"Enter the no of elements "<<endl;
//     int n,k,value;
//     cin>>n;
```

```
//     cout<<"Enter elements:"<<endl;
//     for(int i=0;i<n;i++)
//     {
//         cin>>k;v1.push_back(k);
//     }
//     cout<<"Original Vector:"<<endl;
//     for(int i=0;i<n;i++)
//     {
// cout<<v1[i]<<" ";
//     }


// sort(v1.begin(),v1.end());
// v1.resize(distance(v1.begin(),unique(v1.begin(),v1.end())));
//  cout<<"\nUpdated Vector:"<<endl;
//     for(int i=0;i<(v1.size());i++)
//     {
// cout<<v1[i]<<" ";
//     }
int n,value;
   n=7;
   int arr[7];
   cout<<"Enter  7 elements:"<<endl;
   for(int i=0;i<7;i++)
   {
       cin>>arr[i];
   }
int*ptr =max_element(begin(arr),end(arr));
cout<<endl<<distance(begin(arr),ptr);


   return 0;
}
```

```
Enter  4 elements:
7 7 3 4
Enter element to get its frequency
7
Frequency of 7 is: 2
```

```
Enter the no of elements
7
Enter elements:
12 13 7 14 2 3 0
Original Vector:
12 13 7 14 2 3 0
Enter element to erase:
13
Updated Vector:
12 7 14 2 3 0 %
```

```
Enter the no of elements
7
Enter elements:
0 0 3 3 2 1 7
Original Vector:
0 0 3 3 2 1 7
Updated Vector:
0 1 2 3 7 %
```

```
 Enter  7 elements:
 1 2 3 4 7 5 6

 4%
```

3. Use **std::list (**class of the List container) to perform the following:

a. Finds the value of the first element in the list.
b. Finds the value of the last element in the list.
c. Adds a new element at the end of the list.
d. Removes the first element of the list, and reduces the size of the list by 1.
e. Inserts new elements in the list before the element at a specified position.
f. Returns the size of the list.
g. Removes all the elements from the list, which are equal to a given element.
h. Reverses the list.
i. Removes all duplicate consecutive elements from the list.
j. swap the contents of one list with another list.

```
#include <iostream>

#include <list>

using namespace std;

int main() {

list<int> lst;

lst.push_back(1);

lst.push_back(2);

lst.push_back(3);

lst.push_back(4);

lst.push_back(4);

cout << "originsl List:\n";

for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";

cout << endl;
```

```
cout << "First element: " << lst.front() << endl;
cout << "Last element: " << lst.back() << endl;
lst.push_back(6);
cout << "After adding 6 at the end: ";
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
lst.pop_front();
cout << "After removing the first element: ";
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
list<int>::iterator it = lst.begin();
advance(it, 1);
lst.insert(it, 10);
cout << "After inserting 10 at the 2nd position: ";
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
cout << "Size of the list: " << lst.size() << endl;
lst.remove(3);
cout << "After removing all elements equal to 3: ";
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
lst.reverse();
cout << "After reversing the list: ";
```

```
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
lst.unique();
cout << "After removing consecutive duplicates: ";
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
list<int> lst2;
lst2.push_back(7);
lst2.push_back(8);
lst2.push_back(9);
cout << "List 2:\n";
for (list<int>::iterator it = lst2.begin(); it != lst2.end(); ++it) cout << *it << " ";
cout << endl;
lst.swap(lst2);
cout << "After swapping with another list: ";
for (list<int>::iterator it = lst.begin(); it != lst.end(); ++it) cout << *it << " ";
cout << endl;
return 0;
}
```

```
originsl List:
1 2 3 4 4
First element: 1
Last element: 4
After adding 6 at the end: 1 2 3 4 4 6
After removing the first element: 2 3 4 4 6
After inserting 10 at the 2nd position: 2 10 3 4 4 6
Size of the list: 6
After removing all elements equal to 3: 2 10 4 4 6
After reversing the list: 6 4 4 10 2
After removing consecutive duplicates: 6 4 10 2
List 2:
7 8 9
After swapping with another list: 7 8 9
```

4. Use **std::map** Member Functions to

a. Find the number of elements in the map.
b. Add a new element to the map.
c. Removes the key-value ‘g’ from the map.

```
#include <iostream>
#include <map>

using namespace std;

int main() {
    map<char, int> myMap;

    // Take initial map size input
    int n;
    cout << "Enter the number of elements to insert initially: ";
    cin >> n;

    // Take initial elements input
    for (int i = 0; i < n; ++i) {
        char key;
```

```
        int value;
        cout << "Enter key and value (e.g., a 1): ";
        cin >> key >> value;
        myMap[key] = value;
    }

    // a. Find the number of elements in the map
    cout << "Number of elements in the map: " << myMap.size() << endl;

    // b. Add a new element to the map
    char newKey;
    int newValue;
    cout << "Enter new key and value to add (e.g., d 4): ";
    cin >> newKey >> newValue;
    myMap[newKey] = newValue;

    // c. Remove the key-value pair with a specific key
    char keyToRemove;
    cout << "Enter key to remove: ";
    cin >> keyToRemove;
    myMap.erase(keyToRemove);

    // Print the map to verify changes
    cout << "Map contents:\n ";
    map<char, int>::iterator it = myMap.begin();

    while (it != myMap.end()) {
        cout << "Key: " << it->first
             << ", Value: " << it->second << endl;
        ++it;
    }
    cout << endl;

    return 0;
}
```

```
Enter the number of elements to insert initially: 3
Enter key and value (e.g., a 1): c
1
Enter key and value (e.g., a 1): a 2
Enter key and value (e.g., a 1): t 3
Number of elements in the map: 3
Enter new key and value to add (e.g., d 4): c
4
Enter key to remove: c
Map contents:
 Key: a, Value: 2
Key: t, Value: 3
```

```
Enter the number of elements to insert initially: 4
Enter key and value (e.g., a 1): a 4
Enter key and value (e.g., a 1): b 7
Enter key and value (e.g., a 1): c 11
Enter key and value (e.g., a 1): d 17
Number of elements in the map: 4
Enter new key and value to add (e.g., d 4): e 22
Enter key to remove: c
Map contents:
 Key: a, Value: 4
Key: b, Value: 7
Key: d, Value: 17
Key: e, Value: 22
```