Lab 1 – STL

Objective: Objective of this topic is to learn about Standard Template Library of C++. This knowledge will be required throughout the course to code easily.

Standard Template Library is a C++ library which contains some useful data structures like Stack, Queue, Vector, Priority Queue, Map etc. They come very handy while coding graph theory related codes.

Vector

Vectors are sequence container that can change size. Container is an object that holds data of same type. Sequence containers store elements strictly in linear sequence.

Vector stores elements in contiguous memory locations and enables direct access to any element using operator []. Unlike array, vector can shrink or expand as needed at run time. The storage of the vector is handled automatically.

Following is the way to declare vector type variable:
vector <DATA_TYPE> VARIABLE_NAME;

The vector operations that will be used frequently in this course are:
.push() - It is used to insert an element at the back of a vector.
.size() - It is used to find out the number of elements kept inside the vector.

To use vector we need to add **#include <vector>.**

Example: You are given a list of **N** numbers in a vector. Write a program to find out if summation of all odd numbers in the vector is **greater than** summation of all even numbers.

Sample Input	**Sample Output**
5 1 2 3 4 5 4 1 2 3 4	YES NO

Sample Code for Vector

```
#include <stdio.h>
#include <vector>

using namespace std;

int main() {
   int n;
   scanf("%d", &n);
   vector<int> v;

   for(int i = 0; i < n; ++i) {
      int a;
      scanf("%d", &a);
      v.push_back(a);
   }

   int odd = 0, even = 0;

   for(int i = 0; i < n; ++i) {
      odd += (v[i] % 2 == 0) ? 0 : v[i];
      even += (v[i] % 2 == 0) ? v[i] : 0;
   }

   if(odd > even) printf("YES\n");
   else printf("NO\n");
}
```

Map

Map is dictionary like data structure. It is a sequence of (key, value) pair, where only single value is associated with each unique key. It is often referred as associative array.

Following is the way to declare map type variable:
map <KEY_DATA_TYPE, VALUE_DATA_TYPE> VARIABLE_NAME;

An example of storing data in a **string, double** pair map is given below:
map <string, double> mp;
mp["ABC"] = 10.203;
mp["EFG"] = -5.360;
Now if we print mp["ABC"], then 10.203 will be printed.
And for mp["EFG"] it will be -5.360.

To use map we need to add **#include <map>.**

Example: You are given a list of **N** student names and their achieved numbers in final. You have to query a student name. If the student name and her/his number has already been taken as input then you will check if the number is >= 80. In that case print "A+". Otherwise print "Not A+". If the student's name has not been taken as input, print "NO STUDENT RECORD AVAILABLE". The sample IO is given below:

Sample Input	**Sample Output**
3 Tamim 50 Shakib 80 Mahmudullah 85 Mustafiz	NO STUDENT RECORD AVAILABLE

Sample Code for Map

```
#include <iostream>
#include <map>
using namespace std;

int main() {

  int n;
  scanf("%d", &n);

  map<string, int> mp;

  for(int i = 0; i < n; ++i) {
     string str;
     int a;
     cin >> str >> a;
     mp[str] = a;
  }

  string query;
  cin >> query;
```

```
    if(mp[query]) {
        if(mp[query] >= 80.0)
            cout << "A+" << endl;
        else
            cout << "NOT A+" << endl;
    }
    else {
        cout << "NO STUDENT RECORD AVAILABLE" << endl;
    }

    return 0;
}
```

Priority Queue

Priority queues are a type of container adaptors, specifically designed such that its **first element is always the greatest of the elements** it contains, according to some strict weak ordering criterion.

This context is **similar to a heap**, where elements can be inserted at any moment, and only the max heap element can be retrieved (the one at the top in the priority queue).

Priority queues are implemented as container adaptors, which are classes that use an encapsulated object of a specific container class as its underlying container, providing a specific set of member functions to access its elements. Elements are popped from the "back" of the specific container, which is known as the top of the priority queue.

Following is the way to declare priority queue type variable:
priority_queue <DATA_TYPE> VARIABLE_NAME;

Useful priority queue operations for this course are given below:
.empty() - Test whether container is empty
.top() - Access top element
.push() - Insert element
.pop() - Remove top element

To use priority queue we need to add **#include <queue>.**

Example: You are given a list of N numbers. You need to write a program such that the numbers will be stored in ascending order in a priority queue.

Sample Input	**Sample Output**
5 1 4 3 1 7	1 1 3 4 7

Sample Code for Priority Queue

```
#include <stdio.h>
#include <queue>

using namespace std;

int main() {
   int n;
   scanf("%d", &n);
   priority_queue<int> q;

   for(int i = 0; i < n; ++i) {
      int a;
      scanf("%d", &a);
      q.push(-a);
   }

   while(!q.empty()) {
      int u = q.top(); q.pop();
      printf("%d ", -u);
   }
}
```

Task:

1. Write a program to demonstrate usage of Vector.
2. Write a program to demonstrate usage of Map.
3. Write a program to demonstrate usage of Priority queue