```
#include <iostream>
#include<algorithm>
#include<vector>
#include<array>
#include<deque>
#include<list>
#include<stack>
#include<queue>
#include<set>
#include<map>

using namespace std;
int main() {
  cout << "Hello World!\n";
  cout << endl;
```

Array:

```
 //creating an array of int type and of size 4
array<int,6> a ={10,20,30,40,50,60};

//printing the indexes of the array
 for(int i=0;i<a.size();i++)
   {
     cout<<i<<" ";
   }cout<<endl;

//printing the elements of the array
 for(int i=0;i<a.size();i++)
   {
     cout<<a[i]<<" ";
   }cout<<endl;


//to get the size of the array
int size = a.size();
cout<<"the size of the array is: " << size << endl;

 //for accessing the element present at the second index,we use at(<index_number>) operation

  cout<<"element present at the second index is: "<<a.at(2)<<endl;

 //for checking whether the array is empty or not
 //we use empty() operation
 //this function returns boollean value i.e.either true or false
 cout<<"empty or not-> "<<a.empty()<<endl;

//if we want to get the first and last element of the array,we use front() and back() operation.

cout<<"the first element of the array is-> "<<a.front()<<endl;

cout<<"the last element of the array is-> "<<a.back()<<endl;
```

```
0 1 2 3 4 5
10 20 30 40 50 60
the size of the array is: 6
element present at the second index is: 30
empty or not-> 0
the first element of the array is-> 10
the last element of the array is-> 60
```

Vector:

```
//creating a new vector of int type
     vector<int>v;

  //inputing elements in the vector using push_back() operation

    v.push_back(1);
    v.push_back(5);
    v.push_back(6);
    v.push_back(12);
    v.push_back(15);
    v.push_back(20);
//to print the element of the vector
  for(int i:v)
    {
      cout<< i <<" ";
    }cout<<endl;

  //if we want to see the size of the vector,we use size() operation

  cout<<"size of the vector is: "<<v.size()<<endl;

  //if we want to view the capacity of the vector,we use capacity() operation
cout<<"capacity of the vector is: "<<v.capacity()<<endl;

  //if we want to access the element present at second index,we use at(<index>)

  cout<<"the element present at second index is: "<<v.at(2)<<endl;

  cout<<endl;

//if we want to get the first and last element of the vector,we use front() and back() operation.

cout<<"the first element of the vector is-> "<<v.front()<<endl;

cout<<"the last element of the vector is-> "<<v.back()<<endl;

cout<<endl;

//we can take out the last element using pop_back() operation;
 cout<<"before pop the vector is "<<endl;
  for(int i:v)
    {
      cout<< i <<" ";
    }cout<<endl;
```

```
v.pop_back();
  cout<<"after pop the vector is:"<<endl;
    for(int i:v)
    {
      cout<< i <<" ";
    }cout<<endl;
```

//if we want to clear the vector,in this case we use clear() operation

//Note-> when we use clear operation ,the size of the vector becomes zero not the capacity.

```
cout<<"before clear the size of the vector is: "<<v.size()<<endl;
cout<<"before clear the capacity of the array is: "<<v.capacity()<<endl;

v.clear();

cout<<"after clear the size of the vector is: "<<v.size()<<endl;
cout<<"after clear the capacity of the array is: "<<v.capacity()<<endl;
```

Deque:

//also known as Doubly Ended Queue
//random access possible

```
//creating a deque of int type
deque<int>d;
```

//inputing elements in the doubly ended queue
//we use push_back() and push_front() operation

```
d.push_back(20);
d.push_back(30);
d.push_back(40);
d.push_back(50);
d.push_back(60);
d.push_front(10);
```

//for printing the elements of the deque

```
for(int i:d)
  {
    cout<< i << " ";
  }cout<<endl;
```

//if we want to remove the elements from front and back in this case we use pop_front() and pop_back() operations respectively.

```
  d.pop_back();
  cout<<endl;
  for(int i:d)
  {
    cout<< i << " ";
  }cout<<endl;

 d.pop_front();
  cout<<endl;
  for(int i:d)
  {
    cout<< i << " ";
  }cout<<endl;


//if we want to access the element present at any index then we use at() function.
cout<<"The zeroth Index element is: "<<d.at(0)<<endl;
cout<<"The first Index element is: "<<d.at(1)<<endl;
cout<<"The second Index element is: "<<d.at(2)<<endl;
cout<<"The third Index element is: "<<d.at(3)<<endl;
cout<<"The fourth Index element is: "<<d.at(4)<<endl;
cout<<"The fifth Index element is: "<<d.at(5)<<endl;

//in order to print the first and last element of the deque,we use front() and back() operation
respectively

  cout<<endl;

cout<<"the first element of the deque is: "<<d.front()<<endl;
cout<<"the last element of the deque is:"<<d.back()<<endl;

//to check whether the deque is empty or not,in this we use empty() function

  cout<<"empty or not-> "<<d.empty()<<endl;

 //if we want to remove one element from the deque( from starting)

 cout<<"before erase->"<<d.size()<<endl;

  d.erase(d.begin(),d.begin()+1);
 cout<<"after erase->"<<d.size()<<endl;

  cout<<endl;
  cout<<endl;
```

```
10 20 30 40 50 60
The zeroth Index element is: 10
The first Index element is: 20
The second Index element is: 30
The third Index element is: 40
The fourth Index element is: 50
The fifth Index element is: 60

the first element of the deque is: 10
the last element of the deque is:60
empty or not-> 0
before erase->6
after erase->5
```

List:

```
//implementation through doubly linked list
//direct accesssing of element is not possible.
//man lete hai ki hame fourth element nikalna hai...toh hame wana tak travel karke jana padega....aisa ni ki hum at() operation ka use karke pahuch jayenge....

//creating a list of int type..

  list<int>l;

 //inputing elements in the list
l.push_back(2);
l.push_back(3);
l.push_back(4);
l.push_front(1);

//for printing the list..
 cout<<"the list is:"<<endl;
for(int i:l)
  {
    cout<<i<<" ";
  }cout<<endl;

 //for erasing the element of the list...
//for this we use erase() function.

//In this function we give iterator...and to which element it points it deletes that element for eg- l.begin() points to first element and l.begin()+1 points to second element and so on.

  cout<<"before erase the list is"<<endl;
  for(int i:l)
  {
    cout<<i<<" ";
  }cout<<endl;

 l.erase(l.begin());//deleting the first element
cout<<"after erase the list is"<<endl;
for(int i:l)
{
  cout<<i<<" ";
}cout<<endl;
//if we want to make new list of size 5 and initialize  all with 100

list<int> n(5,100);

cout<<"the new list n is"<<endl;
  for(int i:n)
    {
    cout<<i<<" ";
    }cout<<endl;
```

```
the list is:
1 2 3 4
before erase the list is
1 2 3 4
after erase the list is
2 3 4
the new list n is
100 100 100 100 100
```

STACK:

```
//follows last in first out mechanism.

//creating a stack of string  data type
stack<string> s;

//inserting an elements into the stack,this can be done with push() operation.
 s.push("Animish");
 s.push("Tripathy");

//if we want to see the top element then this can be implemented using top() operation
 cout<<"the top element of the stack is: "<<s.top()<<endl;

//if we want to remove the top element i.e. Tripathy,in this case we use pop() operation.
cout<<"the top element of the stack before poping  is: "<<s.top()<<endl;
 s.pop();
cout<<"the top element of the stack after poping is: "<<s.top()<<endl;

//if we want to see the size of the stack then we use size() function
cout<<"the size of the stack is: "<<s.size()<<endl;

//if we want to see whether the stack is empty or not then in this case we use empty() operation
//returns boollean value
 cout<<"empty or not-> "<<s.empty()<<endl;
```

```
the top element of the stack is: Tripathy
the top element of the stack before poping  is: Tripathy
the top element of the stack after poping is: Animish
the size of the stack is: 1
empty or not-> 0
```

Queue:

```
 //follows first in first out mechanism

 //creating queue of int type
 queue<string> q;
  //inserting element using push() operation
 q.push("Animish");
 q.push("tripathy");
```

```
  //if we want to see the first element and last element of the queue then in this case we use front() and back() functions respectively.
 cout<<"the first element of the queue is : "<<q.front()<<endl;//result must be animish

 cout<<"the last element of the queue is : "<<q.back()<<endl;

//if we want to remove the element from the queue then we use pop() operation.
q.pop();
 cout<<"after poping the first element of the queue is: "<<q.front()<<endl;
  //if we want to see the size of the queue,in this case we use size() operation.
 cout<<"size of the queue after pop is: "<<q.size()<<endl;
```

//time complexity of all of these operations is O(1).

```
the first element of the queue is : Animish
the last element of the queue is : tripathy
after poping the first element of the queue is: tripathy
size of the queue after pop is: 1
```

Priority Queue:

```
 //it is a type of data structure in which the first element is always greatest.
//default is max heap.
//in this of max heap if we want to take out the element of the priority_queue then in that case the maximum element will be popped out.

//creating the priority_queue MAX HEAP:
 priority_queue<int> maxi;

  //if we want to create MIN HEAP then in that case we have to write
priority_queue<int,vector<int> ,greater<int> > mini;

//inserting element in max heap
 maxi.push(1);
 maxi.push(3);
 maxi.push(2);
 maxi.push(0);
 maxi.push(4);

  //size of the priority_queue
  cout<<"size of the priority_queue(MAX HEAP):"<<maxi.size()<<endl;

  //printing the element
  int n = maxi.size();
  for(int i=0;i<n;i++)
    {
      cout<<maxi.top()<<" ";
      maxi.pop();
```

```
  }cout<<endl;
 cout<<endl;
 cout<<endl;

 //inserting the elements in the min heap
 mini.push(1);
 mini.push(3);
 mini.push(5);
 mini.push(4);
 mini.push(2);

//printing the elements of the min heap
 int m = mini.size();
 for(int i=0;i<m;i++)
 {
   cout<<mini.top()<<" ";
   mini.pop();
 }cout<<endl;
```

//to check whether the priority_queue is empty or not then we have to use empty() function

```
cout<<"khalli hai kya bhai ??? "<<mini.empty()<<endl;//result must be one as we have popped out all the elements of the min heap.
```

```
size of the priority_queue(MAX HEAP):5
4 3 2 1 0


1 2 3 4 5
khalli hai kya bhai ??? 1
```

SET:

//saare ke saare unique element store hote hai....agar humne five times five insert kar diya tab bhi ek hi baar five store hoga.
//implementation behind the scene using BST.
//returns elements in sorted order.

```
 //creating a set of int type
set<int> s;

//inserting the elements into the set,this can be done using insert() function.
s.insert(0);
s.insert(5);
s.insert(3);
s.insert(2);
s.insert(4);
s.insert(1);
```

//time complexity of insert operation is O(logn);

```
  //printing the elements of the set
```

```
 for(int i:s)
   {
     cout<< i << " ";
   }cout<< endl;

//initializing the iterator(it)
 set<int>::iterator it = s.begin();
  it++;

//erase the element of the set
  s.erase(it);

  for(auto i:s)
    {
      cout<< i << " ";
    }cout<<endl;//output must be 0 2 3 4 5

 //count(<element>)-batata hai ki element hai ya ni
 cout<<"5 is present or not-> "<<s.count(5)<<endl;

  //find()-it returns the iterator of that element
  set<int>::iterator itr = s.find(4);

  //for eg--
  for(auto it=itr;it!=s.end();it++)
    {
      cout<< it <<" ";

    }cout<<endl;
```

//time complexity of insert(), find(),erase(),count() is O(log n).

//time complexity of size(),begin(),empty() is O(1);

MAP:

```
//value stored in the form of key value pair.
//keys must be unique.
//value can be same

 //creating a map
  map<int,string> m;

 m[1]="animish";
 m[2]="tripathy";
```

NOTE :-another method of inserting the elements into map.that is using insert() function.

```
m.insert({3,"abhinay"});
m.insert({4,"tripathy"});
```

```
//printing the elements of the map
 for(auto i:m)
   {
     cout<<i.first<<" "<<i.second<<endl;
   }

//to check whether the key 3 is present or not
cout<<"finding 3-> "<<m.count(3)<<endl;

//erasing any key...this can be done using erase(<key>) function.
  cout<<endl;
 cout<<"before erase the map is->"<<endl;
   for(auto i:m)
   {
     cout<<i.first<<" "<<i.second<<endl;
   }
 m.erase(4);
  cout<<endl;
 cout<<"after erase the map is->"<<endl;
  for(auto i:m)
   {
     cout<<i.first<<" "<<i.second<<endl;
   }
```

the time complexity of the erase(),insert(),find(),count() is:- O(log n)

```
  cout<<endl;

 auto it=m.find(1);

for(auto i=it;i!=m.end();i++)
  {
    cout<<(*i).first<<" ";
  }cout<<endl;
```

```
1 animish
2 tripathy
3 abhinay
4 tripathy
finding 3-> 1

before erase the map is->
1 animish
2 tripathy
3 abhinay
4 tripathy

after erase the map is->
1 animish
2 tripathy
3 abhinay

1 2 3
```

STL Algorithms

```
// using binary_search() function->
    bool res=binary_search(v.begin(),v.end(),3);

   cout<< "finding 3-> "<< res << endl;

    bool result=binary_search(v.begin(),v.end(),12);

    cout<<"finding 2-> " << result << endl;

  cout<<endl;

  //finding maximum of a and b using max() function

  int a=3;
  int b=4;

  cout<<"the maximum of a and b is->"<<max(a,b)<<endl;

  cout<<"the minimum of a and b is->"<<min(a,b)<<endl;

  cout<<endl;

  //swapping using swap() function

  cout<<"before swapping the value of a is->"<<a<<endl;
  cout<<"before swapping the value of b is->"<<b<<endl;

  swap(a,b);

  cout<<"after swapping the value of a is->"<<a<<endl;
  cout<<"after swapping the value of b is->"<<b<<endl;

  cout<<endl;

  //reversing any string using reverse() function.

  string s="animish";
  reverse(s.begin(),s.end());
  cout<<"the reversed string is: "<< s <<endl;

  cout<<endl;


//in order to rotate any vector,we use rotate()function

Note:-in this we have to give the forward iterator,the middle element(upto which element we have to
rotate),end(specifying the end of the vector)

  rotate(v.begin(),v.begin()+1,v.end());
```

```
  cout<<"after rotate"<<endl;
  for(int i:v){
    cout<< i << " ";

  }

//in order to sort, we use sort() operation

  cout<<"the vector before sorting is: "<<endl;


}
```

```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    vector<int>v;
    int n;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        int x;
        cin>>x;
        v.push_back(x);
    }
    auto it = min_element(v.begin(),v.end());
    cout << "The minimum element in the array is:" << (*it) << endl;

    auto iterator = max_element(v.begin(),v.end());
    cout << "The maximum element in the array is:" << (*iterator) << endl;

    int sum = accumulate(v.begin(),v.end(),0/*initial sum*/);
    cout << "The sum of all the elements in the array is:" << sum << endl;

    int cnt = count(v.begin(),v.end(),2);
    cout<<"Count of 2 is:- " << cnt  << endl;

    auto value = find(v.begin(),v.end(),12);
    if(value!=v.end())
    {
        cout << *value << endl;
    }
    else
    {
        cout << "Element not found" << endl;
    }

```

```
13
1 1 2 2 3 3 3 4 5 5 5 5 5
```

outputf.in

```
The minimum element in the array is:1
The maximum element in the array is:5
The sum of all the elements in the array is:44
Count of 2 is:- 2
Element not found
```

```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    //lambda function
    cout << [](int x){return x>0;}(-2) << endl;

    auto sum = [](int x,int y){return x+y;};
    cout << sum(3,4) << endl;

    int n;
    cin >> n;
    vector<int>v;
    for(int i=0;i<n;i++)
    {
        int x;
        cin>>x;
        v.push_back(x);

    }
    cout << all_of(v.begin(),v.end(),[](int x){return x>0;})<< endl;
    cout << any_of(v.begin(),v.end(),[](int x){return x>0;})<< endl;
    cout << none_of(v.begin(),v.end(),[](int x){return x<0;})<< endl;
```

```
13
1 1 2 2 -3 3 3 4 5 5 5 5 5
```

outputf.in

```
0
7
0
1
0
```