01 September 2023 20:53

STL

Standard Template Library

data structures & their algo.

✓✓ `#include <bits/stdc++.h>` → All data structures

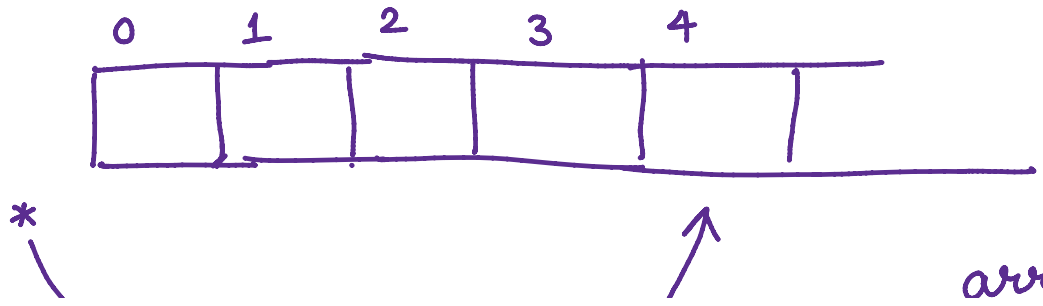
```
#include <vector>
#include <map>
```

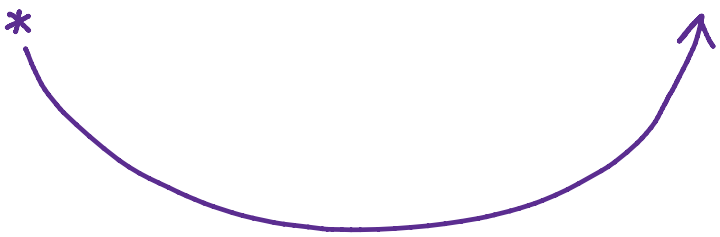
- Vector
- list
- set
- map
- stack
- queue

Vector (dynamic array)

→ contiguous, same data type.

↓ size can change



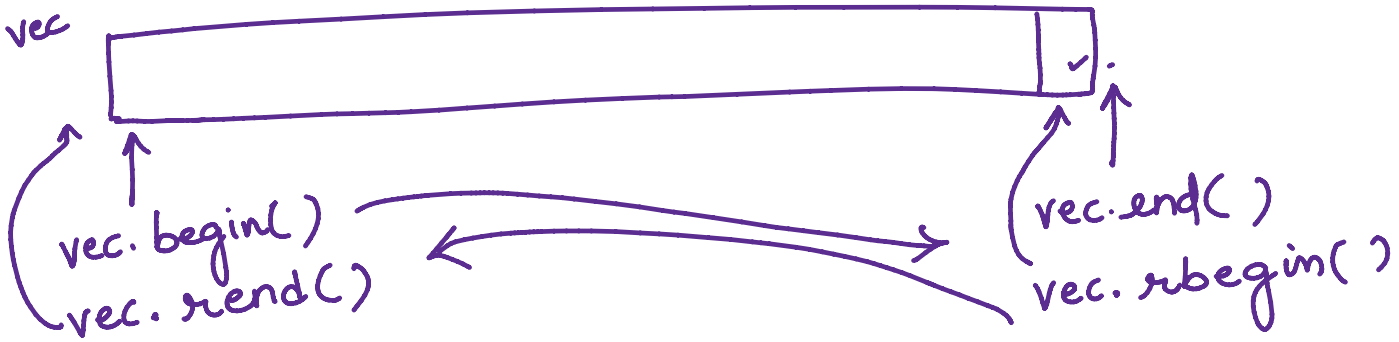


arr [3]

vector <datatype> name;

vector <int> vec1;

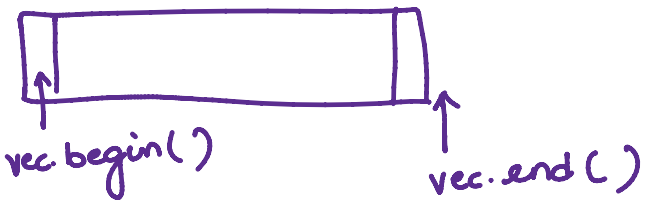
vector <char> vec2;

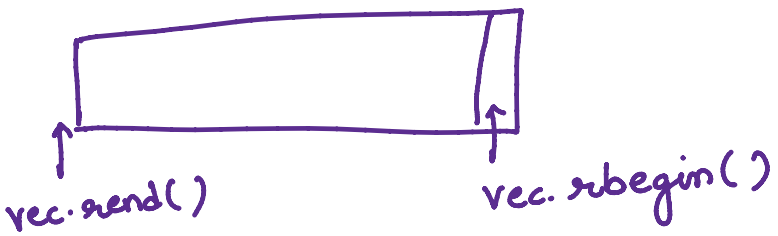


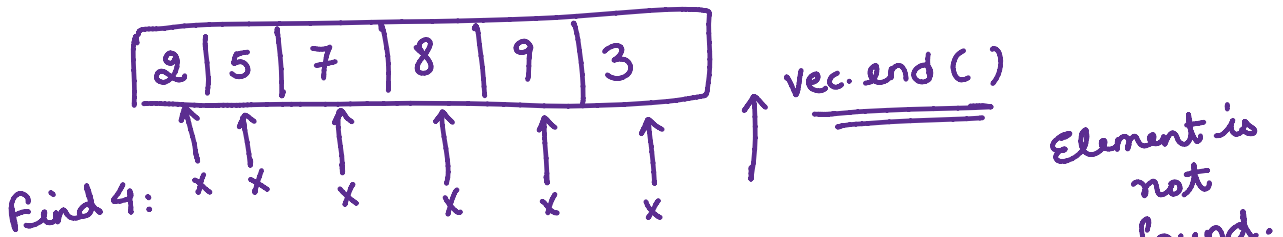
vector <int> :: iterator it = vec.begin();

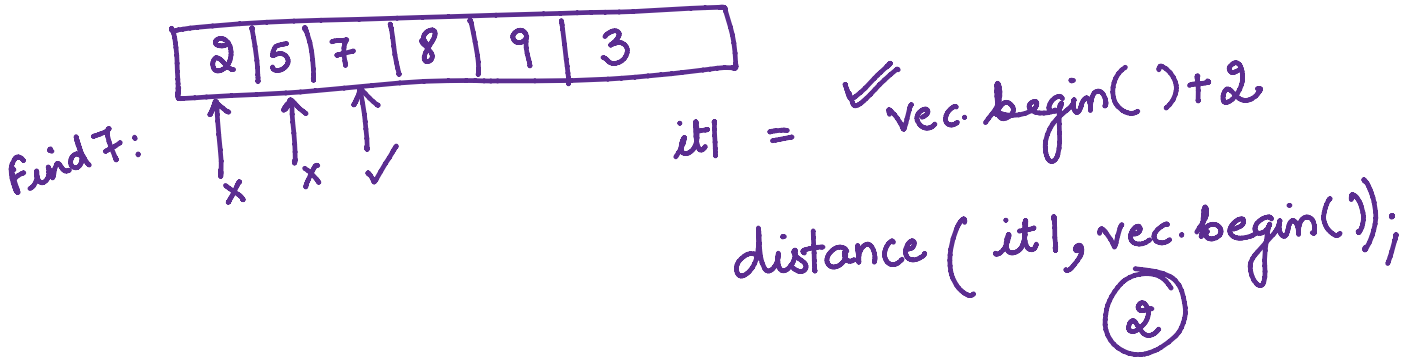
auto it = vec.begin();

```
for (int i = 0; i < vec.size(); i++)
        cout << vec[i];
```









.

```
#include<iostream> //Header File
#include<bits/stdc++.h>
using namespace std;
int main(){ //Start of the main function
   vector<int> vec;
   //Insertion (at end)
   vec.push_back(10);
   vec.push_back(15);
   vec.push_back(7);
   for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   for(auto it=vec.rbegin();it!=vec.rend();it++){ //Print from right to left
      cout<<*it<<" ";
   }
   cout<<endl;
   //Insertion (at middle)
   vec.insert(vec.begin()+1,20);
   for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Overwriting
   vec[2]=5; //OR vec.at(2)=5;
   for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Deletion (at end)
   vec.pop_back();
   for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Deletion (in middle)
   vec.erase(vec.begin()+1);
   for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
```

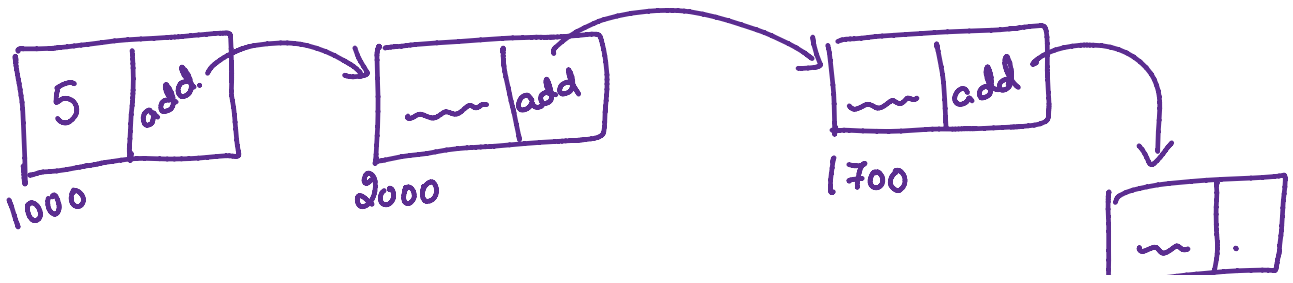
```
        cout<<*it<<" ";
    }
    cout<<endl;
    vec.push_back(20);
    vec.push_back(2);
    vec.push_back(35);
    vec.push_back(18);
    vec.push_back(2);
    vec.push_back(2);
    for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
        cout<<*it<<" ";
    }
    cout<<endl;
    //Sort (in ascending order)
    sort(vec.begin(),vec.end());
    for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
        cout<<*it<<" ";
    }
    cout<<endl;
    //Reverse
    reverse(vec.begin(),vec.end());
    for(auto it=vec.begin();it!=vec.end();it++){ //Print from left to right
        cout<<*it<<" ";
    }
    cout<<endl;
    //Searching
    int element = 100;
    auto it2 = find(vec.begin(),vec.end(),element);
    if(it2!=vec.end()){
        cout<<element<<" is found at "<<distance(vec.begin(),it2)<<endl;
    }
    else{
        cout<<element<<" is not found"<<endl;
    }
    element = 10;
    auto it3 = find(vec.begin(),vec.end(),element);
    if(it3!=vec.end()){
        cout<<element<<" is found at "<<distance(vec.begin(),it3)<<endl;
    }
    else{
        cout<<element<<" is not found"<<endl;
    }
    //Counting
    element=2;
    int n= count(vec.begin(),vec.end(),element);
    cout<<element<<" occurs "<<n<<" times."<<endl;
    //Minimum Element
    int min_value = *min_element(vec.begin(),vec.end());
    cout<<"Minimum Value is "<<min_value<<endl;
    //Maximum Element
    int max_value = *max_element(vec.begin(),vec.end());
    cout<<"Maximum Value is "<<max_value<<endl;
}
```

2 2 5 5

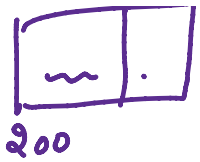
(2) List

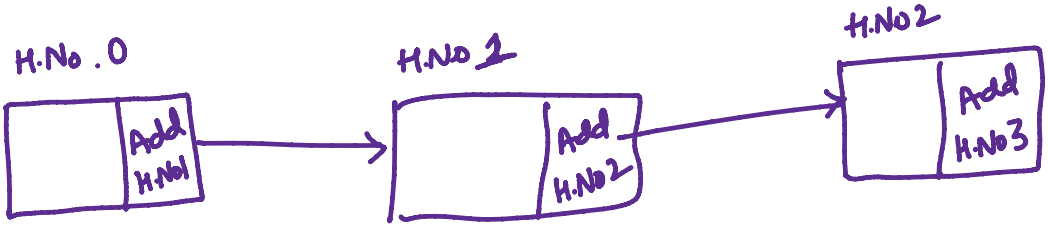
Linked list: → Elements are not stored in contiguous location

→ same data type.

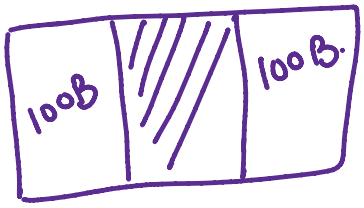


1000 2000 1700





list <int> li;



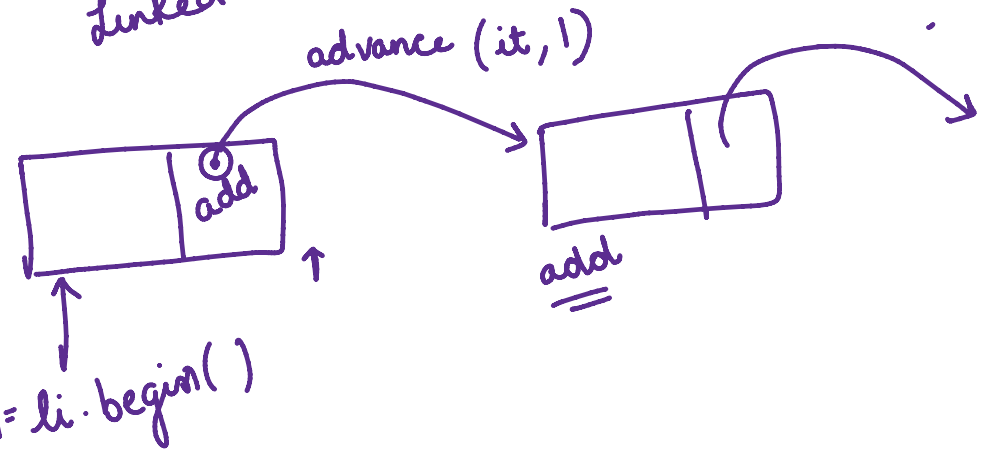
50 elements

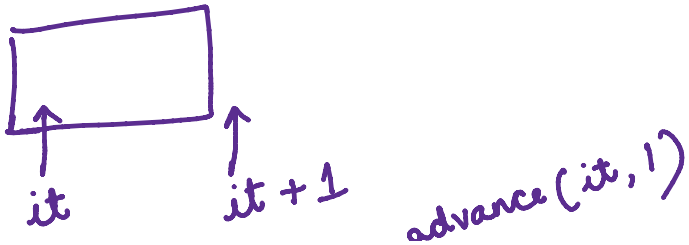
INT 4B

200 Bytes

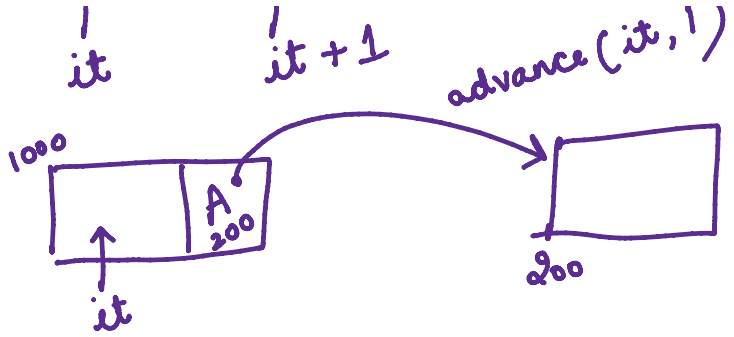
ARRAY: X

Linked list ✓





advance (it, 1)



.

```
#include<iostream> //Header File
#include<bits/stdc++.h>
using namespace std;
int main(){ //Start of the main function
   list<int> li;
   //Insertion (at end)
   li.push_back(10);
   li.push_back(15);
   li.push_back(7);
   for(auto it=li.begin();it!=li.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Insertion (at middle)
   auto it1=li.begin();
   advance(it1,1);
   li.insert(it1,20);
   for(auto it=li.begin();it!=li.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Deletion (in middle)
   auto j=li.begin();
   advance(j,2);
   li.erase(j);
   for(auto it=li.begin();it!=li.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   li.push_back(20);
   li.push_back(2);
   li.push_back(35);
   li.push_back(18);
   li.push_back(2);
   li.push_back(2);
   for(auto it=li.begin();it!=li.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Sort (in ascending order)
   li.sort();
   for(auto it=li.begin();it!=li.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Reverse
   li.reverse();
   for(auto it=li.begin();it!=li.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Searching
   int element = 100;
   auto it2 = find(li.begin(),li.end(),element);
   if(it2!=li.end()){
      cout<<element<<" is found"<<endl;
   }
   else{
      cout<<element<<" is not found"<<endl;
   }
   element = 10;
   auto it3 = find(li.begin(),li.end(),element);
   if(it3!=li.end()){
      cout<<element<<" is found"<<endl;
   }
   else{
      cout<<element<<" is not found"<<endl;
   }
   //Counting
   element=2;
   int n= count(li.begin(),li.end(),element);
```

```
    cout<<element<<" occurs "<<n<<" times."<<endl;
    //Minimum Element
    int min_value = *min_element(li.begin(),li.end());
    cout<<"Minimum Value is "<<min_value<<endl;
    //Maximum Element
    int max_value = *max_element(li.begin(),li.end());
    cout<<"Maximum Value is "<<max_value<<endl;
}
```

(3) Set

→ All elements are unique

2 ~~2~~ 5 ~~5~~ 1 ~~1~~

→ Automatically Sorted 1 2 5

→ Cannot access elements by index

set [0] ✗

10, 20, 20, 40, 5, 35, 5, 30, 20

↓

5, 10, 20, 30, 35, 40 (Sorted & unique)

.

```
#include<iostream> //Header File
#include<bits/stdc++.h>
using namespace std;
int main(){ //Start of the main function
   set<int> s;
   //Insertion
   s.insert(10);
   s.insert(7);
   s.insert(5);
   s.insert(7);
   s.insert(20);
   for(auto it=s.begin();it!=s.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
   cout<<endl;
   //Deletion
   s.erase(7);
   for(auto it=s.begin();it!=s.end();it++){ //Print from left to right
      cout<<*it<<" ";
   }
```

```
    cout<<endl;
    //Searching
    int element = 100;
    auto it2 = s.find(element);
    if(it2!=s.end()){
       cout<<element<<" is found"<<endl;
    }
    else{
       cout<<element<<" is not found"<<endl;
    }
    element = 10;
    auto it3 = s.find(element);
    if(it3!=s.end()){
       cout<<element<<" is found"<<endl;
    }
    else{
       cout<<element<<" is not found"<<endl;
    }
    //Counting (Either 0 or 1)
    element=20;
    int n= s.count(element);
    cout<<element<<" occurs "<<n<<" times."<<endl;
    //Minimum Element
    int min_value = *s.begin();
    cout<<"Minimum Value is "<<min_value<<endl;
    //Maximum Element
    int max_value = *s.rbegin();
    cout<<"Maximum Value is "<<max_value<<endl;
}
```

(4) map

Pair

Key → Value

* Key Value Pairs

Roll No — Name

(1, "A")
(2, "B")
3 "C"

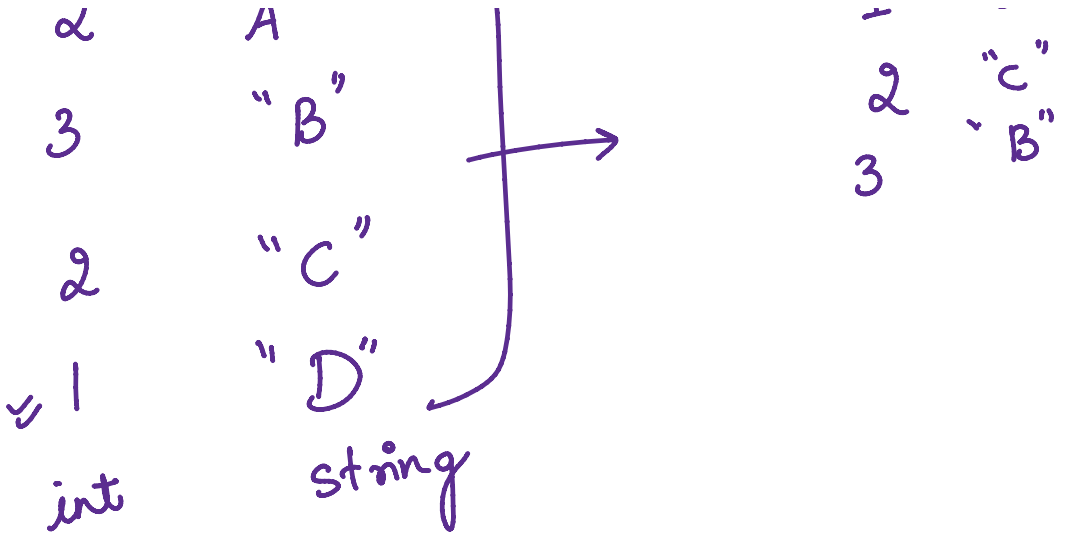
Keys are unique

1 → "A" ✗
1 → "B"

Keys are stored in sorted form

2 "A"
"B"

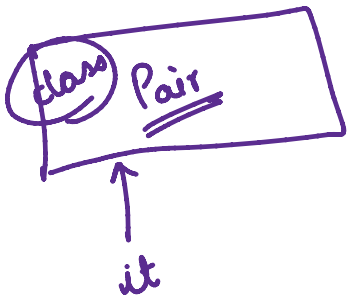
1 "D"
2 "C"




```
map < int , string > m1;
map < int , int > m2;
```

Key — Value

```
map < Key , Value > name;
```




```
class Pair {
    int first;
    string second;
}
```

First (key)
Second (Value)

it -> first
it -> second

Sec

it -> second

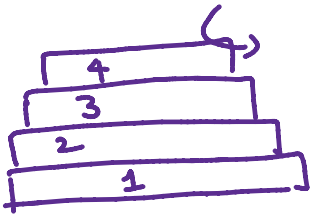
*it;

.

```
#include<iostream> //Header File
#include<bits/stdc++.h>
using namespace std;
int main(){ //Start of the main function
   map<int,string> m;
   //Insertion  name[key] = value
   m[101]="ABC";
   m[51]="PQR";
   m[200]="XYZ";
   m[101]="MNO";
   for(auto it=m.begin();it!=m.end();it++){ //Print from left to right
      cout<<it->first<<" "<<it->second<<" ";
   }
   cout<<endl;
   cout<<"Value at Key 101 is "<<m[101]<<endl;
   //Minimum Element (Key)
   auto min_ptr = *m.begin();
   cout<<"Minimum Key is "<<min_ptr.first<<" and its value is
"<<min_ptr.second<<endl;
   //Maximum Element (Key)
   auto max_ptr = *m.rbegin();
   cout<<"Maximum Key is "<<max_ptr.first<<" and its value is
"<<max_ptr.second<<endl;
}
```

Stack -> LIFO

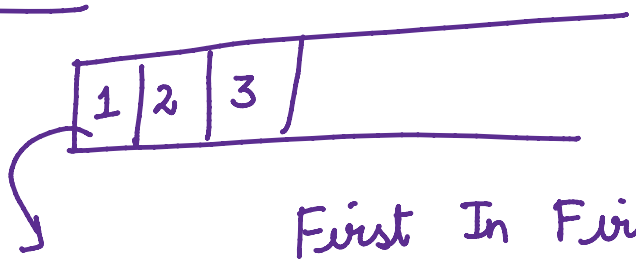
my_stack.top()



Last In First Out

Queue -> FIFO

my_queue.front()



First In First Out

stack <int> my_stack;

stack <int> my_stack;

queue <int> my_queue;

Add (Stack, Queue)

Push (5)

Deletion (Stack, Queue)

Pop ( )

.

```cpp
#include<iostream> //Header File
#include<bits/stdc++.h>
using namespace std;
int main(){ //Start of the main function
   stack<int> my_stack;
   my_stack.push(10);
   my_stack.push(15);
   my_stack.push(5);
   my_stack.push(20);
   while(!my_stack.empty()){
      cout<<my_stack.top()<<endl;
      my_stack.pop();
   }
}
```

.

```cpp
#include<iostream> //Header File
#include<bits/stdc++.h>
using namespace std;
int main(){ //Start of the main function
   queue<int> my_queue;
   my_queue.push(10);
   my_queue.push(15);
   my_queue.push(5);
   my_queue.push(20);
   while(!my_queue.empty()){
      cout<<my_queue.front()<<endl;
      my_queue.pop();
   }
}
```