C++

Vector class of STL

=> including & initialising vectors -

```
#include<vector>
vector<int> arr(n,0);
```

(vector → type, arr → name, n → size of vector, 0 → initialising value for all elements)

```
arr.resize(9); → resizing
arr.resize(9,3);
```

=> getting spaced inputs in C++.

```
int l, r.
cin >> l >> r;
```

=> size of vector → arr.size()

=> arr.push_back(i) → append i at the last of vector

=> arr.pop_back() → to remove an element from last

=> Pbds → Policy based data structures

find_by_order(k) → return the pointer to Kth largest element, K ∈ [0, n)

order_of_key()

Notes

Appointment

to avoid any error "comparison b/w signed & unsigned int" : use "size_t" data type for storing indices

02 July
Wednesday

JULY					2008
Mon		07	14	21	28
Tue	01	08	15	22	29
Wed	02	09	16	23	30
Thu	03	10	17	24	31
Fri	04	11	18	25	
Sat	05	12	19	26	
Sun	06	13	20	27	

1, st largest

1, 2, 3, 4

0th largest element → min no.

=> vector<vector<int> > vv;

=> if(vvv)
{
continue;
}
This statement won't run

=> #include <bits/stdc++.h>
→ min(a,b)
→ max(a,b)
→ swap(a,b)

→ double pow(double base, double exponent)
↳ all values are double typed

→ cout << fixed << setprecision(15) << num << "\n";

=> Global variables - defined on top
- initialised with zero (By default)
- visible everywhere
Local variables - defined in functions
- initialised randomly (garbage value)
- visible just in the function

Notes

=> vector<vector<int> > vv(5, vector<int>(10,23));

Appointment

→ we can rewrite global variables by declaring local variable in a function with same name as the local varible.

⇒ Block of code - code between braces

⇒ codeblocks debugger -  step into go to the line then & press next line button.

watcher ≡ workspace in MATLAB

Q) Appearance array -
mode of - 1,7,7,7,3,2,5,10
make an array of size 11 & increment each index when you read it.

⇒ Stack is a list with a restriction that insertion and delition can be performed only from one end, called top.
→ also called last in, first out Data Structure
→ push() - pushes a value on top of the stack 
→ pop() - remove value from top of the stack 
→ isEmpty() - returns true if stack is empty
→ top() → returns top value of a stack

Notes
Appointment

04 July
Friday

Tue	01	08	15	22	29
Wed	02	09	16	23	30
Thu	03	10	17	24	31
Fri	04	11	18	25	
Sat	05	12	19	26	
Sun	06	13	20	27	

=> stack.cpp

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

int stack[100];
int ind;
void push(int x){
    ++ind;
    stack[ind] = x;
}
bool isEmpty(){
    if(ind>=1) return false;
        else return true;
  }

void pop(){
    stack[ind]=0;
 ind--;
  }

int top(){
    return stack[ind];
   }
int main()
 {  ind=0
    push(1);
    push(2);
    pop();
  cout<<top(); }
```

Notes

Appointment

=> To find a given string of brackets is valid or not, use stack open & close brackets do push & pop. if stack is empty in end, strings are valid.

=> Queue - A queue is a list with the restriction that insertion can be performed only from one end (called back) & deletion can be performed only from the other end (called front)

Sunday 06

→ operations - push() - inserts at the back of queue
- pop() - deletes element from back
- isEmpty() - Returns true if queue is empty
- front() - returns the value at front end

Appointment

→ It is called first in, first out d.s.

=> Queue.cpp

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

int Queue[1000], backInd, frontInd;

void push(int x){
    backInd++;
    Queue[backInd] = x;
}
void pop(){
    Queue[frontInd] = 0
    frontInd++;
}
bool isEmpty(){
    if(backInd < frontInd) return true;
        else return false;
}
int front(){
    return Queue[frontInd];
}
int main()
{
    backInd = -1;
    frontInd = 0;
    push(5);
    push(7);
}
```

Notes

Appointment

=> Binarysearch.cpp.

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

int A[1000], number of elements;

int binarySearch(int x){
    int left=1, right = numberofElements, mid;
    while(left <= right){
        mid = (Left + Right)/2;

        if (x == A[mid]) return mid; // found x
        else if(x < A[mid]) Right = mid - 1;
                            else left = mid + 1;

    }
    return -1;
}
int main()
{
  f>> numberElements;
  for(int i=1; i<= no. of elemts ; i++)
    f >> A[i]
  cout<<binarysearch(2);
  return 0;
}
```

Notes

Appointment

09 July Wednesday

A[] = [1, 2, 5, 3]

JULY 2008

Mon		07	14	21	28
Tue	01	08	15	22	29
Wed	02	09	16	23	30
Thu	03	10	17	24	31
Fri	04	11	18	25	
Sat	05	12	19	26	
Sun	06	13	20	27	

⇒
```
#include <bits/stdc++.h>
sort(A+0, A+4)
```
0 → 1st index, 4 → last index + 1

→ to sort in decending order
```
bool cmp(int x, int y)
{
    return x > y;
}
```
main
```
sort(A+0, A+4, com);
```

⇒
```
#include <string>
int sum = 5;
string sums = to_string(sum);
```

```
#include < array>
array<int, 5> a = {0, 1, 2, 3, 4}
for(int &i : a)
{ a[i] = a[i] + 1; i = i + 1
}
for(int i : a)
{ cout << a[i] << endl;
```

Notes

Appointment

NPTEL

string class of STL

=> #include <string>
string v = "abcdab";
string w(v) // equal to w = v
string x = v + w
v[2] = v[3]; // v = "abddab" now
cout << v.substr(2) // substring "ddab"
v.substr(1, 3)
↳ starts at 1 of length 3
int i = v.find("ab"); // position of 1st occurrence
.find("ab", 1);
↳ starts for searching from given index

→ if string find f" doesn't find anything, it returns a string "string::npos"

=> for reading a sentence

#include <bits/stdc++.h>
char c[100];
cin.getline(c, 100);

=> vector<int> v(10)
sort(v.begin(), v.end());

Notes

Appointment

11 July Friday

JULY 2008

Mon		07	14	21	28
Tue	01	08	15	22	29
Wed	02	09	16	23	30
Thu	03	10	17	24	31
Fri	04	11	18	25	
Sat	05	12	19	26	
Sun	06	13	20	27	

=> STL - maps -

- General form :

```
map <indextype, valuetype > mapname ;
```

- Example :

```
map < string, double > population;
```

```
population ["India"] = 1.37;
population.count(country);
cout << population["India"];
```

```
# include < utility >
    pair < int, char > P;
    P = make_pair(65, 'A');
    cout << P.first;
    cout << P.second;
```

=> Hybrid containers.

```
array < pair < string, int >, 3 > stu ;
```

Notes

Appointment

Harshit
Github
stl

July
Saturday

12

Pointers - datatypes

=> &a → address of variable a

=> *p → value stored at address p.

=> int *p;

*p = a;

If p = 1245

then, p+1 = 1249 } Difference of 4 because integer is of 4 bytes