Page No.
Date

Time Complexity (TLE) Table.

$\leq [10 \ldots 11]$	$O(n!)$ $O(n^6)$
$\leq [15 \ldots 18]$	$O(2^n * n^2)$
< 100	$O(n^4)$
< 400	$O(n^3)$
< 2000	$O(n^2 * \log n)$
$< 10^4$	$O(n^2)$
$< 10^6$	$O(n \log n)$
$< 10^8$	$O(n)$, $O(\log n)$

Note: To convert the C++ style string to the C style string use the

- `C_str();` → convert the string style type.

(#) **Binary Search** : Only applicable on monotonic fun^n (sorted).

Time complexity = O(logn).

Q. WAP to search an element using Binary search ?

Hint : mid = start + (end - start)/2 ;
To reduce the out of range error.

Q. Calculate the first and the last occurence of an element in an sorted array.

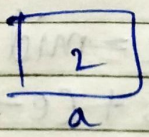
Hint: Same as Binary search.

first = - - -	last . . .
ans = mid ;	ans = mid ;
e = mid - 1 ;	s = mid + 1 ;

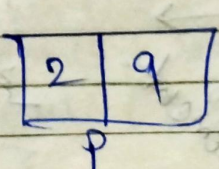
(*) `Pair< int , int> P;`

↳ use to send two values at the same time

Eg:- `int a = 2;`




```
Pair<int, int> P;
P.first = 2;
P.second = 9;
```

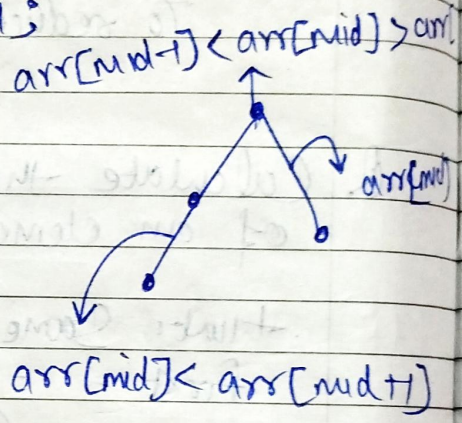


(*) Total no. of occurence = (last − first) + 1;

(*) Largest element in Peak mountain sorted array.

Hint: if(arr[mid] < arr[mid+1])
s = mid+1;
else
e = mid;
return s;

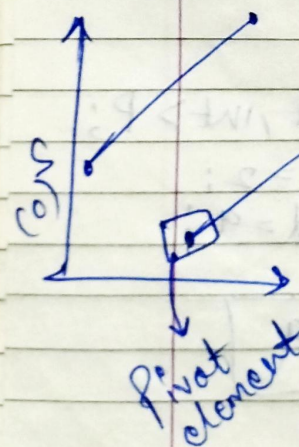
Leetcode



Eg:- 7 9 1 2 3 → Pivot.

(*) Pivot element in an rotated sorted array.

Hint: while(s<e){
if(arr[mid] >= arr[0])
s = mid+1;
else
e = mid;
mid = s + (e-s)/2;
} return s;



Page No.
Date

(*) Parameters written in function definition are formal parameters

Parameters written in function call is actual parameters.

```
Eg:- int fun_call( int arr[], int n)
     {
       =
       =
     }
int main() {
     fun_call(even, 4);
     }
```

formal parameter

actual parameter

mid+1]

→ arr[mid+1]

(*) Pass by value

↳ only copy is send to the ~~actual~~ formal Parameter.

```
Eg:- int calc(int a, int b)
       {
         =
       }
main(){
    calc (5, 4);
}
```

Pass by reference.

↳ reference is send to the formal Parameter.

```
Eg:- int calc(int &a, int &b)
       {
         =
       }
main () {
int x = 15; y = 19;
   calc (x, y);
}
```

Page No.
Date

Recursive function :

function is called recursive when function calls itself., either directly or in-directly.

- Recursive problem Solving
  * Base case
  * Divide the rest of problem into Sub-problem and do recursive call.

Eg:- factorial (n!)

```
usigned long long factorial (unsigned long long n)
                {
        ┌──→      if (n==0)
base case //            return 1;
Recursive case //→ return n * factorial(n-1);
                }

int main() {
        cout << factorial(8) << endl;
        return 0;
        }
```

ans. 8! = 40320

Page No.
Date

(*) Eg:- Fibonacci Series → f(5)=5, f(7)=13.

0 0, 1, 1, 2, 3, 5, 8, 13, 21 ...

fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2) // Recursive Relation.

Base case:
{ fib(0)=0
{ fib(1)=1

Recursive case:
{ fib(n) = fib(n-1) + fib(n-2)

code: ↓

```
unsigned long long fibonacci (unsigned long long n) {
base case //     if ( n <= 1)
                    return n;
Recursive case //  return fibonacci (n-1) + fibonacci (n-2);
                }

                int main() {
                cout << fibonacci (30) << endl;
                    return 0;
                }
```

→ f(30).

output: 832040.

Time complexity: O(2^n)
Space complexity: O(n)

Page No.
Date

(*) Search Space → Binary Search.

↳ Eg:- Book allocation
Eg:- painter partition problem
Eg:- Aggressive cows problem.

* imp.

Soln: Book Allocation

| 10 | 20 | 30 | 40 |

S.S (min, max)
(s) min = 0 ;
(e) max = Sum;

0 —— 50 (mid) —— 100

10, 20, 30, 40

mid = 50

I : 10 + 20 + 30

II : 30 + 40

No soln.

s = mid + 1;

51 —— 75 (mid) —— 100

10, 20, 30, 40

I : 10 + 20 + 30 + 40 ← move to left

II : 40 ✓

75 ✓

e = mid - 1;

Store ans;

10	90
10	20, 30, 40

30	70
10, 20	30, 40

60	40
10, 20, 30	40

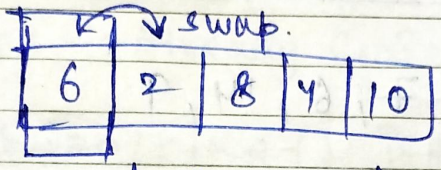
90
70
60 ↓ min (60) ✓

Page No.
Date

(*) **Selection Sort** : → select element and swap with min element.

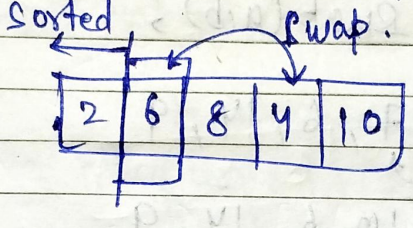
Eg :- | 6 | 2 | 8 | 4 | 10 |

Pass 1 :



Check rest of the arr for min term and swap

Pass 2 :



∴ and so on.

Code :

```
for(int i=0; i<n-1; i++) {
    int minIndex = i;
    for(int j=i+1; j<n; j++) {
        if( arr[j] < arr[minIndex])
            minIndex = j;
    }
    swap( arr[i], arr[minIndex]);
}
```

Space complexity : O(1)
Time complexity : $O(n^2)$

Best/worst case : $O(n^2)$.

use case : arr size small ↓ use this !

Page No.
Date

(*) **Bubble Sort :-** → Swap the corresponding element if arr[j] > arr[j+1];

Eg:-

0	1	2	3	4	5
10	1	7	6	14	9

Pass 1: 10, 1, 7, 6, 14, 9

compare a > b
swap(a, b);

1, 10, 7, 6, 14, 9

Pass 1:
0 → (n-1) 1, 7, 10, 6, 14, 9

Pass 2:
0 → (n-2) 1, 7, 6, 10, 14, 9.

Pass 3:
0 → (n-3) 1, 7, 6, 10, 14, 9

1, 7, 6, 10, 9 | 14 → sorted.

unsorted ← | → at right place

... and so on.

Pass i:
0 → (n-i)

Hint:

Pass / loop = (n-1).

Code:

```
for(int i=0; i<n-1; i++) {
    for(int j=0; j<n-i-1; j++) {
        if( arr[j] > arr[j+1])
            swap( arr[j], arr[j+1]);
    }
}
```

Page No.
Date

- Time complexity = $O(n^2)$.
- Space complexity = $O(1)$.
- Best case: → Already sorted. $O(n)$

```
for(int i=0; i<n-1; i++) {
        bool swapped = false;
    for( int j=0; j< n-i-1 ; j++) {
        bool swapped = false;
        if (arr[j] > arr[j+1]);
        { swap ( arr[j] , arr[j+1]);
          swapped = true;
        }
    }
    if (swapped == false)
        break;
}
```

Optimized.

- worst case: Reverse Sorted $O(n^2)$.

→ understand using placing cards in order.

✱ Insertion Sort:

Eg:- arr[] =

0	1	2	3	4	5	6
10	1	7	4	8	2	11

Pass1: 10

i=1 1 < 10 → left side.
10 → shift by 1.
then put 1 to left side of 10.

0	1	2	3	4	5	6
1	10	7	4	8	2	11

Pass2 7

i=2 7 < 10 → left side.
10 → shift by 1
then put 7 to left side of 10
and check 1 > 7 ✗

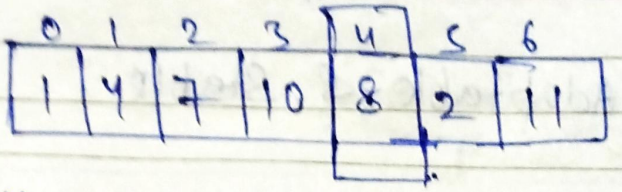
7 > 1 → 9 9

0	1	2	3	4	5	6
1	7	10	4	8	2	11

Pass3 4

i=3 4 < 10 shift to left
4 < 7 shift to left
4 > 1

Put 4 right of 1 and left of 7 & 10



Pass 4
i=4.

. . and so on.

Code:

```
for(int i=1; i<n; i++) {

        int temp = arr[i];
        int j = i-1;
    for( ; j>=0; j--) {
            if( arr[j] > temp) {
// Shift //        arr[j+1] = arr[j];
            }
            else {
                break;
            }
    }
// element at right pos //  arr[j+1] = temp;
}
```

Page No.
Date

use case : Adaptable & Stable.

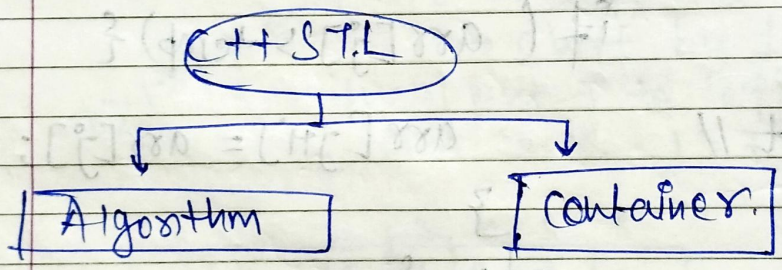
Time complexity :- $O(n^2)$

Space complexity : $O(1)$.

Best case : already sorted. $O(n)$.

Worst case : $O(n^2)$.

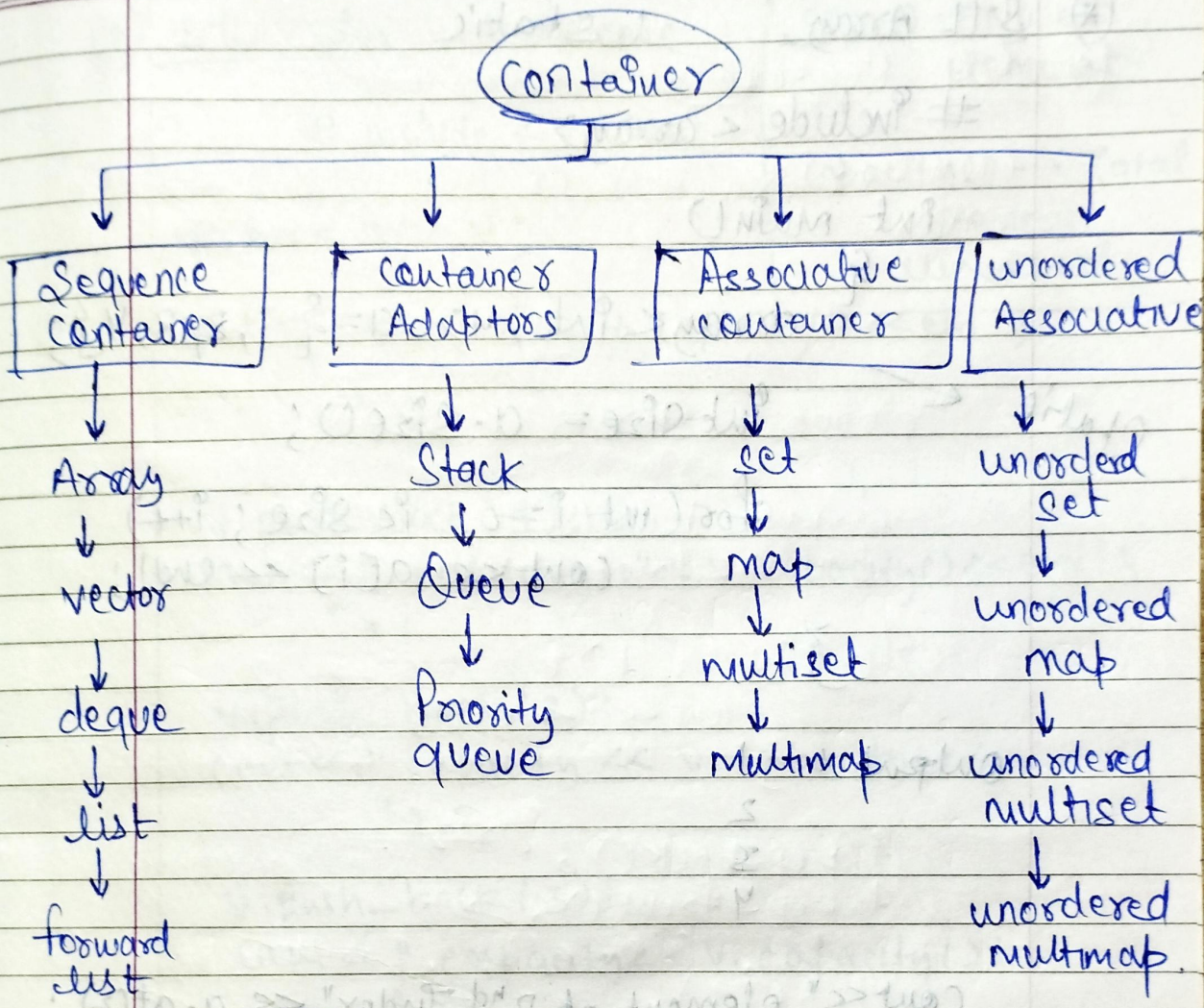
C++ STL (Standard Template Library).

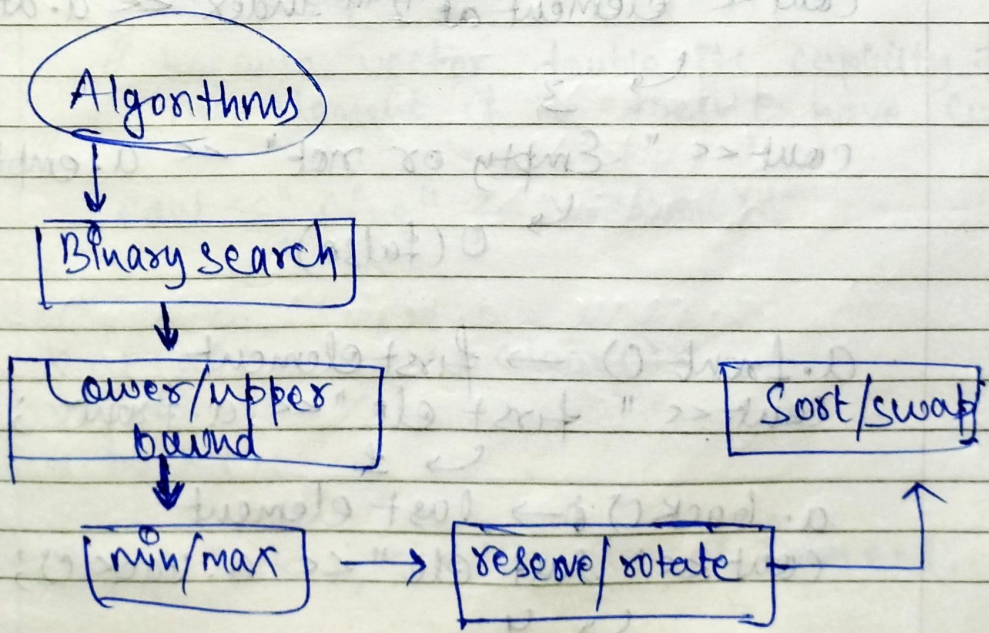



```
#include <bits/stdc++.h>
```

↳ this header file of C++ contains all STL functions.

Page No.
Date


container
Sequence container
container Adaptors
Associative container
unordered Associative
Array
vector
deque
list
forward list
Stack
Queue
Priority queue
set
map
multiset
multimap
unordered set
unordered map
unordered multiset
unordered multimap.


Algorithms
Binary search
Lower/upper bound
min/max
reserve/rotate
Sort/swap

Page No.
Date

(*) STL Array : Static

```
#include <array>

int main()
{
    array<int, 4> a={1,2,3,4};
    int size= a.size();
    for(int i=0; i<size; i++)
        cout<< a[i] << endl;
}
```

static ←

output :
1
2
3
4

```
cout<<" element at 2nd Index" << a.at(2);
```
↳ 3

```
cout<< " Empty or not" << a.empty();
```
↳ 0 (false)

```
a.front() → first element
cout<< " first ele" << a.front;
```
↳ 1

```
a.back() → last element
cout<<" last ele" << a.back();
```
↳ 4

Page No.
Date

(*) STL vector : dynamic

```
#include <vector>
vector<int> v;
```

- .size() → No. of element
- .capacity() → Total memory allocated.

```
cout<<"capacity is" << v.capacity()<<endl;
```
↳ 0

```
v.push_back(1);
cout<<" capacity is"<< v.capacity()<<endl;
```
↳ 1

```
v.push_back(2);
cout<<" capacity << v.capacity();
```
↳ 2

```
v.push_back(3);
cout<<" capacity<< v.capacity();
```
↳ 4.

// because vector double its capacity for new element if it doesn't have size.

```
cout<<" size" << v.size();
```
↳ 3.

```
v.at(2);
v.front();
v.back();
```

```
v.clear();
```

```
v.pop_back();
```
→ delete element from end.

Page No.
Date

(*) STL deque : : dynamic

Can perform push or pop from both the ends.

```
#include<deque>

deque<int> d;

d.push_back(1);
d.push_front(2);

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

→ 2
1

```
d.pop_back();
d.pop_front();
```

→ 2
→ 1

after cout<<;

```
d.at(1);

d.front();
d.back();
d.empty();

d.erase(d.begin(), d.begin()+1);
```

Page No.
Date

(*) **STL List:** implementation using daddy L.L

```
#include <list>

list<int> l;

l.push_back(1);
l.push_front(2);

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

→ 2 1

.begin(), .end(), .empty(), .erase();
.pop_front();, .pop_back();, .size();,

if I want to copy list;

```
list<int> n(l);
list<int> m(5,100);
for(int i: m){
    cout<<i<<endl;
}
```

→ 100
100
100
100
100

Page No.
Date

(*) STL stack : (LIFO)

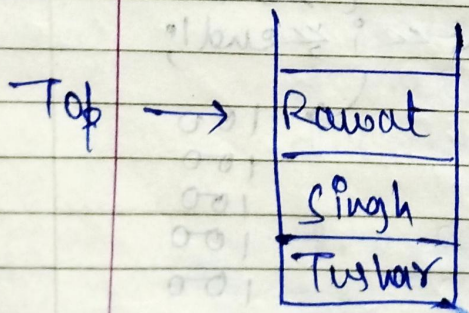
```
#include <stack>

stack <Strings> s;

s.push("Tushar");
s.push("Singh");
s.push("Rawat");

cout<<"Top element" << s.top() <<endl;
```

↳ Rawat

~~cout << "bottom element << s.bottom() <<endl;~~

~~↳ Tushar~~

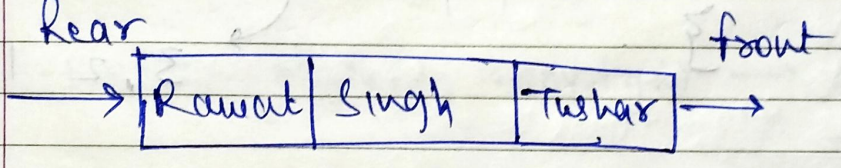
s.pop(); , s.size(); , s.empty(); , etc.



Page No.
Date

(*) STL Queue: (FIFO)

```
#include <queue>

queue <string> q;

q.Push("Tushar");
q.Push("Singh");
q.Push("Rawat");

(cout << "firstele" << q.front();
        ↳ Tushar.
```



.size(), .POP(), ..etc.

(*) STL priority queue:

```
#Include < queue>

// max heap // default;
  priority-queue <int> maxi;

// min heap;
  priority-queue <int, vector<int>, greater
                                    <int>>
  mini;
```

Page No.
Date

```
maxi.push(1);
maxi.push(3);
maxi.push(2);
maxi.push(0);

cout<<" size is " << maxi.size() <<endl;

int n= maxi.size();

for(int i=0; i<n; i++){
    cout<< maxi.top() <<" ";
    maxi.pop();
}
```

→ 3, 2, 1, 0

(*) **STL set** : only unique element
implementation using BST

```
#include <set>

set<int> s;

s.insert(5);
s.insert(1);
s.insert(5);
s.insert(0);

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

→ 0, 1, 5

Page No.
Date

```
s.erase(s.begin());        // delete only first element.

for(auto i:s){
    cout<<i<<endl;
}                          // → 1, 5

cout<<"5 is present or not"<<s.count(5);
                           // → 1 / True

Set<int>::iterator it = s.find(5);
```

.insert(), .find(), .erase(), .count() → O(n)

.size(), .begin(), .empty() → O(1)

(*) STL Maps :

```
#include<map>

map<int, string> m;

m[1] = "Tushar";
m[2] = "Singh";
m[13] = "Rawat";

for(auto i:m){                 // i.first → 1, 2, 13
    cout<<i.second<<endl;      // i.second → Tushar, Singh, Rawat
}
```

Page No.
Date

```
cout<< "finding -13" << m.count(13) <<endl;
```
→ 1 / true

```
m.erase(13);
```
→ delete m.

.insert(), .erase(), .find(), .count()
→ O(log n)

(*) STL Algorithms:

```
#include<algorithms>

vector<int> v;

v.push_back(1);      1
v.push_back(3);      3
v.push_back(5);      5
v.push_back(7);      7

cout<<"finding 7 -> " << binary_search(v.begin(),
                          v.end(), 7) <<endl;
```
→ 1 / true / present.

Page No.
Date

```
cout<<" lower bound" << lower_bound ( v.begin(), v.end(),
                        6) - v.begin() <<endl;
```
↳ 2

```
cout<<" upper bound << upper_bound ( v.begin(), v.end(),
                        6) - v.begin() <<endl;
```
↳ 3

```
sort(v.begin(), v.end());
```
↳ to sort vector.

```
int a=5;
int b=7;

max(a,b); , min(a,b); , swap(a,b); ,

string S = "abcd";
reverse. (S.begin(), S.end());
```
↳ "dcba"

```
rotate (v.begin(), v.begin()+1);
cout<<" after rotate <<endl;
for( int i:v) {
    cout<<i<<" ";
}
```

1 3 5 7 → (rotate) → 3 5 7 1

Page No.
Date

Soln Reverse an array.

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

using namespace std;

vector<int> reverse(vector<int> v);
void Print(vector<int> v);
int main(){

vector<int> v;

v.push_back(11);
v.push_back(7);
v.push_back(3);
v.push_back(12);
v.push_back(4);

vector<int> ans = reverse(v);

cout<<"printing reverse array"<<endl;
  print(ans);

  return 0;
}

vector<int> reverse(vector<int>v){

  int s = 0, e = v.size()-1;
```

Page No.
Date

```
while(s<=e){
        swap(v[s], v[e]);
            s++;
            e--;
    }
    return v;
}
void Print(vector<int>v){
for(auto i:v){
        cout<<v[i]<<" ";
    }cout<<endl;
}
```

Leetcode

Q4. Merge two sorted array.

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

int main()
{
```

Page No.
Date

```
void merge(int arr1[], int n, int arr2[], int m, int
{
    int i=0, j=0;
    int k=0;
  while( i<n && j<m){
    if( arr1[i] < arr2[j])
        {
          arr3[k++] = arr1[i++];
        }
    else {
          arr3[k++] = arr2[j++];
        }
    }
  while( i<n){
     arr3[k++] = arr1[i++];
    }
  while(j<m){
     arr3[k++] = arr2[j++];
     }
}
```

Page No.
Date

```
arr3[])  void print(int ans[], int n){
           for(int i=0; i<n; i++)
             {
               cout<< ans[i] << " ";
             }
             cout<<endl;
           }

         int main(){
           int arr1[5] = {1,3,5,7,9};
           int arr2[3] = {2,4,6};

           int arr3[8] = {0};

           Merge( arr1,5, arr2,3, arr3);
           print(arr3,8);
           return 0;
         }
```

Eg: arr1[5] = {1,3,5,7,9}
arr2[3] = {2,4,6}

arr3[8] = {1,2,3,4,5,6,7,9}