Week 3

Lecture - 7

Time & space Complexity :-

Time Complexity :- It is a type of computation program complexity that describe the time taken to execute a

① Amount of time taken by an algorithm to run as a function of length of input.

Ex:-

```
cin >> n;
for (int i=0; i<n; i++){
    cout << "hello";
}
```

↳ It runs n time by compiler.

Time depends $\propto n$ → directly proportional

Compiler take

⊙ It is a CPU operation time not a actual time

⊙ $f \propto n$

Time complexity : O(n)

Q-1- Why to study Time & space Complexity?

Ans:- ① Good computer engineer always think about the complexity of the code written by him.

② Resource are limited.

③ Measure algorithm to make efficient program.

④ Asked by interview after every soln you give

Q. What is space complexity?

Ans: Amount of space taken by an algorithm to run as a function of length of input.

Ex: int a = 2; → variable → 2B byte

int b[5]; → array

[S.C → O(1)]

Ex: int n;

cin >> n;

int *b = new int[n];

// print array b

for(int i=0; i<n; i++){

cout << b[i];

}

If we increase value of n then the space is increasing

[S.C → O(n)]

** Unit to Represent Complexity:-

1- Big(O): upper bound. It describe the

2- upper bound of an algorithm's

3- runtime.

Ex- search → | 1 | 2 | 3 | 4 | 5 | 6 |

find 2 → O(1)

6 → O(n)

a) Theta (Θ) : Average case
a) Omega (Ω) : Lowest bound

Big (O) : Complexities

(1) Constant time : $O(1)$

Ex:- int a = 5;

(2) Linear time : $O(n)$

Ex:- for(i=0 ; i<n ; i++){
cout << "*";
}

(3) Logarithmic time :- $O(\log N)$ (Learn Further)

(4) Ex:- for(i=0 ; i<n ; i++){
for(j=0 ; j<n ; j++){
}
};

$O(N^2)$

(4) Quadratic time : $O(N^2)$

(5) Cubic time : $O(N^3)$

```
for(i=0 ; i<n ; i++){
for(j=0 ; j<n ; j++){
for(k=0 ; k<n ; k++){
}
}
}
```

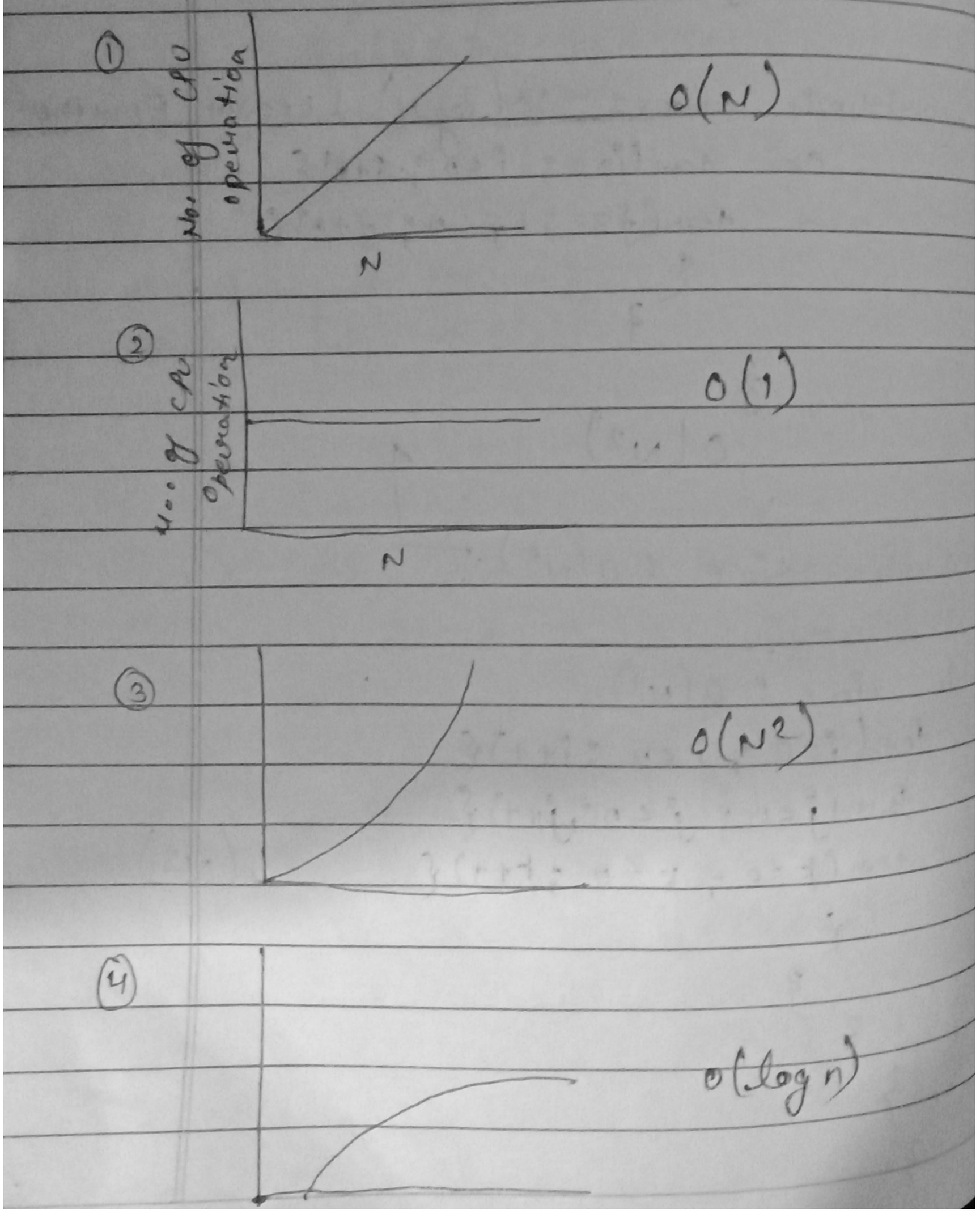
$O(N3)$

Ex:- for (i → N)
{
} N

for (i → N)
{
} N

N + N
O(2N)
O(N) //

Graphs



Example :- $f(n) = 2n^2 + 3n \Rightarrow \theta(2n^2) \Rightarrow \theta(n^2)$

$f(n) = 4n^4 + 3n^3 \rightarrow \theta(n^4)$

$f(n) = N^2 + \log n \rightarrow \theta(n^2)$

$f(n) = 200 \rightarrow O(1)$

$f(n) = (n/4) \rightarrow \theta(n/4) \rightarrow O(n)$

Least Complexity

$O(1)$

$O(\log N)$

$O(\sqrt{N})$

$O(N)$

$O(n \log n)$

$O(N^2)$

$O(N^3)$

$O(2^n)$

$O(N!)$

$O(N^n)$

Maximum Complexity

Ex:-

1	2	3	4	5	6	7	8

(1) Linear search $\rightarrow O(n)$

(2) Binary Seach $\rightarrow O(\log n)$

Example 1:-

```
int main(){
    int a=0, b=0, n, m;
    cin >> n >> m;
    for(int i=0; i<n; i++){
        cout << "Hi" << \n";
    }
    for(int i=0; i<m; i++){
        cout << "Hi 2\n";
    }
}
```

return 0;

}

$O(n) + O(m)$

$TC \rightarrow O(n+m)$ //

(2) int main(){

int a=0, b=0, n;

cin >> n;

for(int i=0; i<n; i++){ $\rightarrow O(n^2)$

for(int i=0; i<n; i++){

cout << "Hi\n";

}

}

for(int i=0; i<n; i++){ $- O(n)$

cout << "Hi 2\n";

}

return 0;

}

$O(n^2) + O(n)$

$TC \rightarrow O(n^2)$ //

(3) int main(){

int a=0, b=0, n;

cin >> n;

for(int i=0; i<n; i++){ $\longrightarrow O(n)$

for(int j=n; j>i; j--){ $- O(n)$

cout << "Hi 1\n";

}

}

$TC \rightarrow O(n^2)$ //

return 0;

}

& space!-

(1) int a = 5; → O(1)

(2) array [5]; → O(1)

(3) int *a = new int [N]; → O(n)

(4) int *b = new int [N2]; → O(n²)

∴ → ✕ ← 1'D - ARRAY → ✕ ←

Arrays in C++!-

Q.1- What is Array?

Ans:- Array are define in following list

① Collection of element

② Similar element

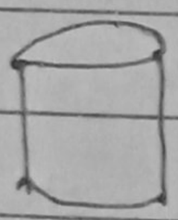
③ continous Memory Block

④ List of similar item

⑤ Data Structure

⑥ Linear Data structure.

Data Structure a like as entity where we store similar data. It may have diff^h technique to storing data.



→ set of similar item
(same data type elements only)

If a bucket of integers then we only store integers value not a char, bool, string & many more values.

⊙ Array always provide continous memory
↓
(line by line)

A → B → C → D → E

Array :- Array is a linear data structure that collect elements of the same type & store them in a contiguous & adjacent memory locations.

Q- Why Array is needed?
Ans:- When we get multiple data & then we have to implement algorithm on it & store data or use it.
So, we use Array.

Ex:- int arr [30000];

Create an Array :-

Syntax :-
datatype variablename [ ]
Square brackets
size of array

Ex:-
//declaring int arr[10];
10 Integer value ka continous space arr ko allocate ho gya hai (40 byte ka)

Starting address called base add.

Ex:- int arr[20];
cout << " Array created" << endl;

② When we creating array we must declare data type of array & size of array.

③ How to find base address?

```
int arr[7];
cout << arr << endl;
cout << &arr << endl;
```

→ Declaration of Array ←

① int arr1 [53];
② char a1 [106];
③ bool arr [23];

→ Intialization ←

① int arr [ ] = {2, 4, 6, 9, 12}

int value

2	4	6	9	12

Ex:- int arr [ ] = {2, 3, 4, 6, 7};
int brr [5] = {2, 4, 6, 8, 10};
int crr [10] = {2, 4, 6, 8, 10};
Array size is (10) but value is (5) then remaining (5) values initialized by (0).

int drr [4] = {2, 4, 6, 8, 10};
It show error array size exceeds the stored value 4 > 5 (No).

All these array are static. (size fixed)
Dynamic array is created when we take arr size by input & provide

input values.

Index & Access in Array :-

Index start from 0 to n-1.

It take 0 based indexing

Ex:- int arr[5] = {2, 4, 6, 8, 10}

arr ↵ 1st 2nd 3rd 4th

0	1	2	3	4
2	4	6	8	10

arr[0] = 2

arr[1] = 4

arr[2] = 6

arr[3] = 8

arr[4] = 10

value of address/r address + [index] * [value of data type]
Base size

Q- Why we start index by 0?

Ans/- Because the formula of value of addresses.

Ex:- int arr[5] = {2, 4, 6, 8, 9}

for (int i=0 ; i<n ; i++){

cout << arr[i] << endl;

}

Static array
I/P

// Taking input array in fixed size?

```
int arr[10];
cout << " Enter the input value " << endl;
for ( int i=0 ; i < 10 ; i++){      // Input
    cin >> arr[i];
}
for (int i=0; i < 10 ; i++){        // Print
    cout << arr[i] << endl;
}
```

Q.1. Take 5 elements I/P in array & print their double of value's?

Ans).

```
int arr[500];
int n}
cout << "How many values you want to input" << endd;
cin >> n;
cout << "Enter the values" << endl;
for (int i=0 ; i < n ; i+1){
    cin >> arr[i];
}
for (int i=0 ; i < n ; i++){
    cout << 2* arr[i] << " ";
}
return 0;
}.
```

Q-2- arr[] = {1, 3, 5, 7, 9}
replace all value to 1

```
int arr[5] = {1,3,5,7,9};
for (int i=0; i<5; i++) {
    arr[i] = 1;
}
for (int i=0; i<5; i++) {
    cout << arr[i] << " ";
}
return 0;
}
```

Q-1- What is Memset function ? Why we use it & where we need it & How apply it in function ?

Ans:- Memset function in c++ copies single character for a specified no. of times to an object.

Ex:- memset ()

```
void * memset (void * dest, int ch,
                size_t count);
```

dest:- Pointer to object to copy characters
ch :- characters to copy
Count:- Number of times to copy

→ The memset function use & need to copies a single characters for a specified no. of time to an object.

→ When we using memset function have to add #include <cstring> header file

```
char ch = 'a';
memset(dest, ch, 5);
cout << "after calling memset" << endl;
cout << "dest contains" << dest;
```

Q- Can be use memset in ques 2 program?

Ans:- Nope, we can not use memset function because memset() casts down to a byte & dupes it across the region.

Q- What is Call by Reference & Pass by reference?

Ans:- Call by Reference is a method which it passes the reference or address of the actual parameters to the function's formal parameters "which means if there any change in the values inside the function, it reflect in the actual value."

Pass by Reference means to pass the reference of an argument in ~~call~~ calling function to the corresponding formal parameters of the called function. "The called function can modify the value of the argument using its reference passed in."

```
int arr[10] = {1, 2};
for (int i=0; i<10; i++){
    cout << arr[i] << " ";
}
return 0;
}
```

o/p - 1, 2, 0, 0, 0, 0, 0, 0, 0, 0

`int arr[10] = {0};` If we want initialize array with(0) only

```
for (int i=0; i<10; i++){
    cout << arr[i] << " ";
}
```

# int arrr[10] = {1};
for(int i=0; i<10; i++){
cout << arrr[i] << " ";
}

cheat it → When we want initialize whole array with 1
code must be wrong
int arrr[10] = {0};
memset(arrr, 1, arrr*10)
for(int i=0; i<10; i++){
cout << arrr[i] << " ";
}

→ Yes, it not converted in 1 as whole array.

Arrays & Function :-

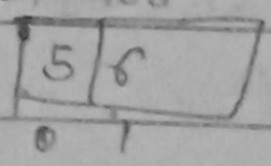
⊙ In function :- (Pass by value) Pass by value

```
main(){
int a=5;
inc(a)
cout a;
}
```

```
int (inc a)
{ a++;
cout << a;
}
```

⊙ In Array :- (Pass by reference)

```
main(){
int arrr[] = {5,6}
inc(arrr)
print(arrr)
}
```




```
inc(int arrr[])
{ arrr[0] = arrr[0]*10;
cout << arrr;
}
```

```
int arr[] = {5,6,3};
int size = 2;
inc(arr, size);
printArray(arr, size);

void printArray(int arr[], int size){
    for(int i=0; i<size; i++){
        cout << arr[i] << " ";
    } cout << endl;
}
void inc(int arr[], int size){
    arr[0] = arr[0] + 10;
    printArray(arr, size);
}
```

We always during we must declare array & size of array.

fun(arr, size)

This program working on pass by reference. In pass by reference we work on actual array place updation or deletion

Q#2) Linear search of Array:-

2	8	6	7	4	12	15	16	5
0	1	2	3	4	5	6	7	8

Array

6 is present in Array → Yes

```
int arr[5] = {1,3,5,7,8};
int size = 5;
cout << "Enter the key to find:" << endl;
int key;
cin >> key;
if (find(arr, size, key)){
    cout << "found" << endl;
}
else{
cout << "Not found" << endl;
}
```

→

```
bool find (int arr[], int size, int key){
    // Linear search
for (int i=0; i<size; i++){
    if (arr[i] == key){
        return true;
    }
    return false;
}
```

OR

Also make normal code without function

Q-2- Count 0's & 1's in Array?

0	0	1	0	1	1	1	0	1

o/p → 0's → 5

1's → 4

```
int arr[] = {0,1,1,1,0,0,0,0,1,1,1,0,1,0};
int size = 14;
int numZero = 0;
int numOne = 0;
for (int i=0; i<size; i++){
    if (arr[i] == 0){
        numZero++;
    }
    if (arr[i] == 1){
        numOne++;
    }
}
cout << "Number of zero" << numZero << endl;
cout << "Number of one" << numOne << endl;
```

Q-3- Maximum number in an Array?

Ans:-

2	4	1	6	8	9	0

max - 9

our minimum No.

When we find maximum no. we have always intialize as:-

int maxNum = INT_MIN

int minNum = INT_MAX

```
#include <limits.h>
int size = 11;
int maxi = INT_MIN;
int arr[] = {2,4,6,1,3,7,9,12,56,43,21};
for (int i=0; i<size; i++){
    if (arr[i] > maxi){
    maxi = arr[i];
}
}
cout << "Maximum no. " << maxi << endl;
```

Their are diff case of maximum find up

① all +ve → working properly
② all -ve → working properly
③ 0, +ve → working properly
④ 0, -ve → working properly
⑤ Half +ve, Half -ve → working properly
⑥ all 0 → working properly
⑦ If maximum no at first position → less time working properly
⑧ If maximum no at last position → same as 7
⑨ If maxi at first & last → working It shows last maximum
⑩ To duplicate maximum value → ↓
It provide last maximum value.

Q.4- Minimum number in an array?

```
#include <limits.h>

int main(){
int arr[] = {2,4,6,10,3,7,9,12,56,43,21};
int size = 10;
int minNum = INT_MAX;

}
```

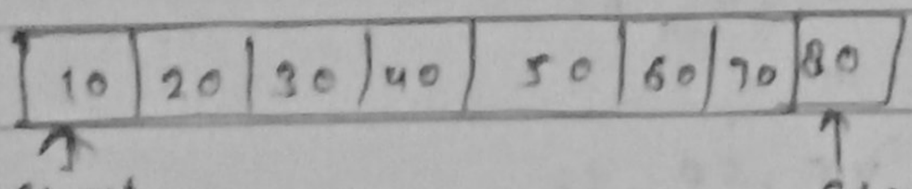
```
for (int i=0; i< size ; i++){
    if (arr[i] < minNum) {
        minNum = arr[i];
    }
}
cout << " Minimum in Array is : " << minNum
     << endl;

}
```

Q.5- Extreme point in Array?

I/P → {1, 2, 3, 4, 5, 6, 7, 8}

O/P → {1, 8, 2, 7, 3, 6, 4, 5}




```
print arr[start]
print arr[start2]
start++;
start2--;
```
} up till start > end

```
int arr[8]= {10,20,30,40,50,60,70,80}
int size = 8;

int start = 0;
int end = size-1;

while (true) {
    if (start > end)
        break;
```
} // while (start <= end)

```
if (start == end){
 cout << arr[start] << " ";
}
 else {
  cout << arr[start] << " ";
  cout << arr[end] << " ";
 } start ++;
  end --;
}
return 0;
 }
```

Q-6- Reverse an ^sorted Array?

Ans:-

I/P- | 10 | 20 | 30 | 40 | 50 | 60 |

O/P- | 60 | 50 | 40 | 30 | 20 | 10 |

10, 20, 30, 40, 50, 60

① 60, 20, 30, 40, 50, 10

② 60, 50, 30, 40, 20, 10

③ 60, 50, 40, 30, 20, 10;

swap (a,b)-

s = 0

e = n-1

① swap (arr[s], arr[e]) } (Loop)

② s++;

③ e--;

```
int arr[8] = {10, 20, 30, 40, 50, 60, 70, 80};
int size = 8;
int start = 0;
int end = size - 1;
while (start <= end)
swap(arr[start], arr[end]);
start++;
end--;
}
for(int i=0; i<size; i++){
cout << arr[i] << " ";
}
```

Case — ① Even element → Done
② Odd element → Done

Homework :-

① Try all question → Done
② Try all question with 3 diffh example
③ Try swap (a,b) function in 3 way's :-
ⓐ + - → Done
ⓑ XOR → Done
ⓒ temp variable → Done

↳ These different technique on Next page

① + , −

```
arr[start] = arr[start] + arr[end];
arr[end] = arr[start] - arr[end];
arr[start] = arr[start] - arr[end];
```

② XOR (^)

```
arr[start] = arr[start] ^ arr[end];
arr[end] = arr[start] ^ arr[end];
arr[start] = arr[start] ^ arr[end];
```

③ Temp

```
temp = arr[start];
arr[start] = arr[end];
arr[end] = temp;
```