Assignment 1

Q1→ Write a Algorithm FIND (DATA, N, LOC1, LOC2) which finds the location LOC2 of the second largest element in an array DATA with n>1 elements?

Ans → FIND (DATA, N, LOC1, LOC2)

(i) declare I, FIRST, SECOND, TEMP
(ii) set FIRST = DATA [1], SECOND = DATA [2], LOC1=1, LOC2=2
(iii) if (FIRST < SECOND)
then TEMP = FIRST; FIRST = SECOND, SECOND = TEMP, LOC1=1, LOC2=2.
(iv) Repeat steps 5 for I = 3 to N
(v) if (FIRST < DATA [I]) then
set SECOND = FIRST, FIRST = DATA [I]
set LOC2 = LOC1, LOC1 = I
else if (SECOND < DATA [I]) then
set SECOND = DATA [I] and LOC2 = I
(vi) write FIRST: LOC1, SECOND: LOC2
(vii) quit

Q2→ Write a program to insert a new element in the given unsorted at Kth position?

Ans →
```
#include <stdio.h>
#include <conio.h>
```

```
int main ()
{
    int array [10], position, c, n, value;
    printf ("Enter number of elements in
            array \n");
    scanf ("%d", &n);

    printf ("Enter %d elements \n", n);

    for ( c = 0; c < n; c++)
    {
        scanf ("%d", &array [c]);
    }

    printf ("Enter the location where you
            wish to insert an element \n");
    scanf ("%d", position);

    printf ("Enter the value to insert \n");
     scanf ("%d", &value);

    for (c = n-1; c >= position-1; c--)
    {
        array [c+1] = array [c];

        array [position-1] = value;
```

```
printf ("Resultant array is \n");
    for (c=0; c<=n; c+1)
        {
            printf ("%d \n", array [c]);
        }
    return 0;
}.
```

Q3→ Write a algorithm to multiply two matrices of size N*N each. Also prove that complexity of this algorithm is O(N²)?

2D ARRAY - MULT

Ans→ Step 1. Read A and B
Step 2. declare C[N][N], I, J, K
Step 3. Repeat step 4 to Step 7 for I=0 to N-1
Step 4. Repeat Step 5 to Step 7 for J=0 to N-1
Step 5. set C[I][J]=0
Step 6. Repeat Step 7 for K=0 to N-1
Step 7. set C[I][J]= C[I][J] + A[I][K] * B[K][J]
Step 8. write C[ ][ ]
Step 9. exit

No. of Steps	Time Constant
1	1

No. of steps	Time Constant	Total time
1	C_1	C_1
1	C_2	C_2
$n+1$	C_3	$C_3(n+1)$
$n(n+1)$	C_4	$C_4 n(n+1)$
n^2	C_5	$C_5 n^2$
$n^2(n+1)$	C_6	$C_6 n^2(n+1)$
n^3	C_7	$C_7 n^3$
1	C_8	C_8
1	C_9	C_9

Total time $= C_1 + C_2 + C_3(n+1) + C_4 n(n+1) + C_5 n^2 + C_6 n^2(n+1) + C_7 n^3 + C_8 + C_9$

$= 1 + 1 + 1(n+1) + 1n(n+1) + 1n^2 + 1n^2 (n+1) + 1n^3 + 1 + 1$

$= 2n^3 + 3n^2 + 2n + 5$

Time Complexity $= O(n^3)$

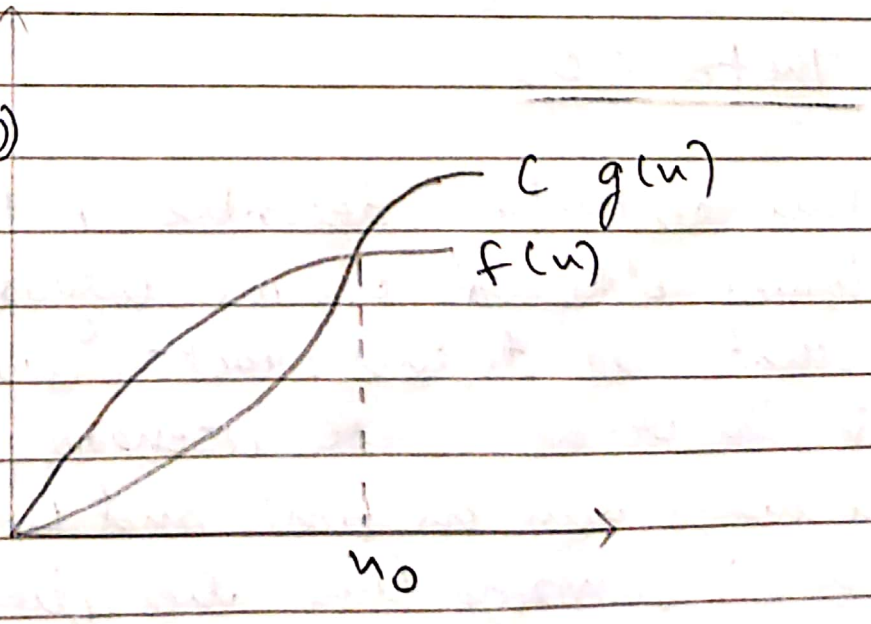
Q4→ What do you understand by complexity of a Algo-rithm ? Explain Big O, omega Ω and Theta Θ Notations with examples ?

Ans→ The complexity of an Algorithm is a function $f(n)$ which measures the time and space used by an algorithm in terms of input size n. In computer science, the complexity of an algorithm is a way to classify how efficient an algorithm is, compared to alternative ones.

- **Big Oh. (O)**

Big Oh notation specially describes worst case scenario. It represents the upper bound running time complexity of an algorithm. Lets take few examples to understand how we represent the time and space complexity using Big O Notation.

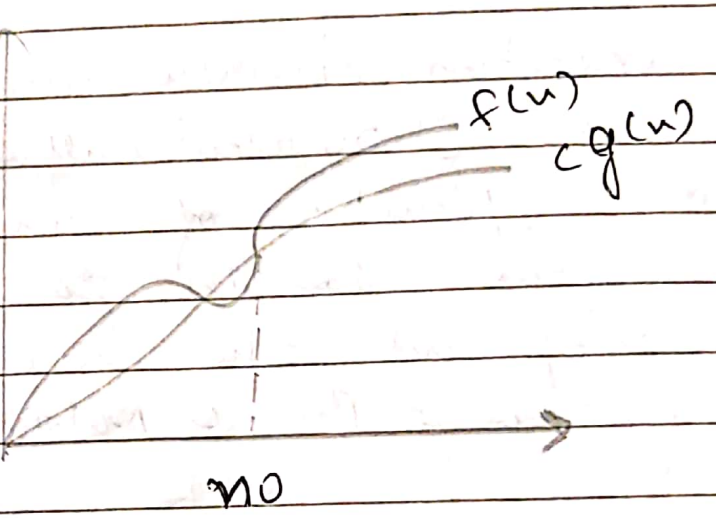
$f(n) \leq c\,g(n)$
then $f(n) = O(g(n))$
for all $n \geq n_0$,
$c > 0$



- **Omega (Ω)**

Omega notation specifically describes best case scenario. It represents the lower bound running time complexity of an algorithm. So if we represent a complexity of an algorithm in omega notation, it means that the algorithm cannot be completed in less time than this, it would at-least take the time represented by omega notation or it can take more (when not in best case scenario).

$f(n) \geq c\,g(n)$
then $f(n) = \Omega(g(n))$
for all $n \geq n_0$
and $c > 0$,



- **Theta (Θ)**

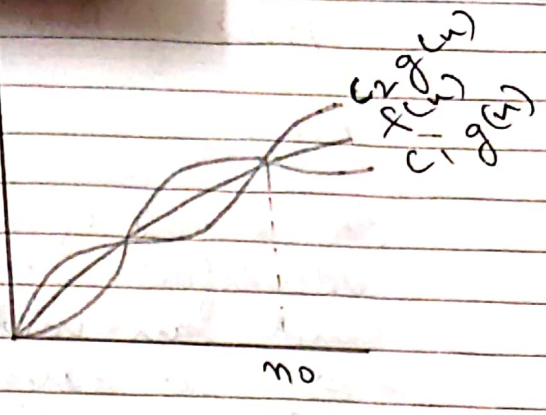
This notation describes both upper bound and lower bound of an algorithm so we can say that it defines exact asymptotic behaviour. In the real case scenario the algorithm not always runs on best and worst cases, the average running time lies between best and worst case & between the theta notation

$c_1 g(n) \leq f(n) \leq c_2 g(n)$

$f(n) = \theta(g(n))$

~~$c_1 g(n) \leq$~~

for all $n \geq n_0$ and $c > 0$



Q5⇒ What is Data structure. Differentiate Data and Information. Also explain types of data structures with examples?

Ans⇒ Data Structure is a data organization and storage format that enables efficient access and modification.

Data – It is raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless until it is organized.

Information → When data is processed, organized, structured or presented in a given content so as to make it is called information.

Types of Data structure

- **Array**

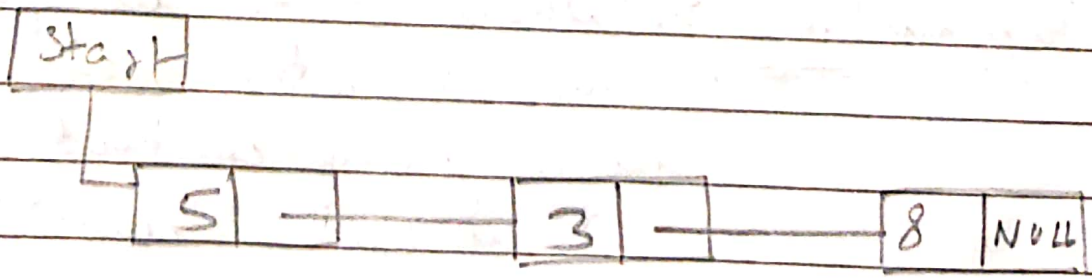
It consists of a collection of elements, each identified by a common variable name and an index. A block of continous memory is assigned for an array. Arrays may be one-dimensional or multi dimensional.

0	5
1	3
2	8
3	4
4	1

- **Linked list**

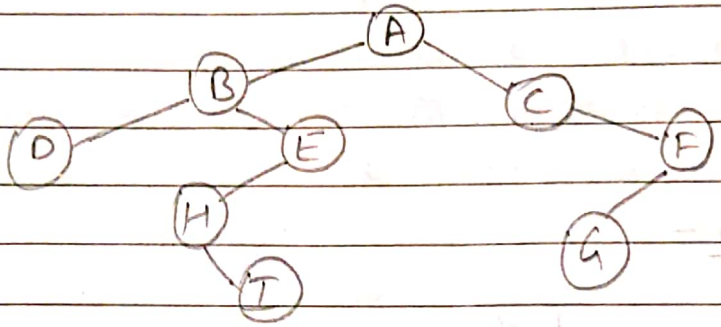
It consists of a group of nodes together which represent a sequence. It consists of two main parts: the data and a pointer pointing to the next node in the list. The start is marked by head pointer and the end is denoted by null pointer.

-



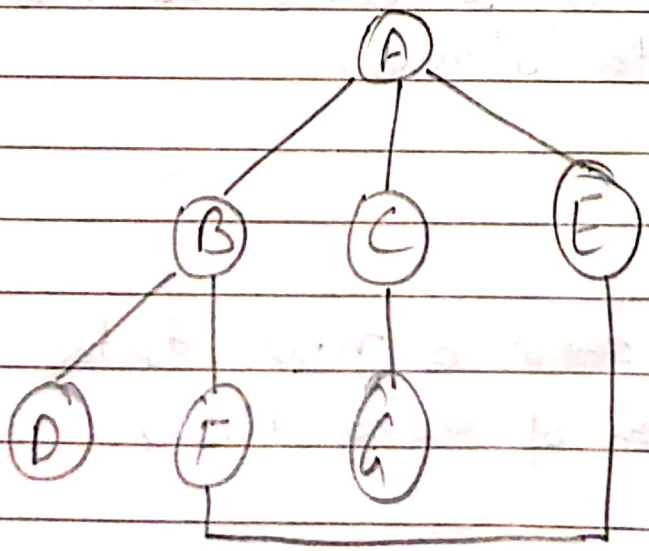
- Tree

It has a set of nodes which have data and pointers or references. The pointers point to the children nodes of the node. The constraint is that no reference is duplicated and no node points to the root.



- Graph

It is a collection of a set of vertices V and a set of edges E. Each edge E is a pair (v, w) where v and w are elements of V (i.e. they are vertices).



Q.6→ Explain different operations can be performed on Data Structures?

Ans→ Following operations can perform on data structure -s:-

1. Traversing
2. Searching
3. Inserting
4. Deleting
5. Sorting
6. Merging.

- Traversing -

It is used to access each data item exactly once so that it can be processed.

- Searching -

It is used to find out the location of the data item if it exists in the given collection of data items.

- Inserting -

It is used to add a new data in the given collection of data items.

- **Deleting**

It is used to delete an existing data item from the given collection of data items.

- **Sorting**

It is used to arrange the data in some order i.e. in ascending or descending order in case of numerical data and in dictionary order in case of alphanumeric data.

- **Merging**

It is used to combine the data items of two sorted files into single file in sorted form.