DATA STRUCTURE (D.S)

Attempt any three of the following.

Q1) What is an Algorithm? Explain properties of an algorithm

→ An algorithm specisies a series of steps that perform a particular computation or task.

- Algorithm resemble recipes. Recapes tell you how to accomplish a task by performs a Dumbet of steps For example, to bake a cake the steps are: preheat the oven; mix flour, sugar, and eggs thraughly. pour into a banking pan; and so forth.
- However "algorithm" as a tenchnical term with more specific meaning than "recipe" and calling Something an algorithm means that the following properties are all true:
- An algorithm is an unambiguous description that makes clears what has to be implemented. In a recipe a step such as "Bake until done" is ambiguous because it doesn't explain what "done" means. A more explicit description such as "Bake until the cheese begins to bubble" is better. In a computational algorithm a step such as "choose a large number" is vague what is large? 1 million, 1 billion, or 100? Does the number have to be different each time, or can the same number be used on every run?
- An algorithm expects a defind set of inputs. for ex:- it, might require two numbers where both numbers are greater than zero. or it might require a a word, or a list zero or more number.

- An algorithm produces defined set of outputs. It mi output the larger of the two numbers an all-upperca version of a word, or a sorted version of the list of nu
- An algorithm is guaranteed to terminate and produce a re always stopping after a finite time if an algorithm Co potentially run forever, it wouldn't be very useful b you might never get an answer.
- Make algorisithm are guaranteed to produce the correct results, It's harely useful of an algorithm returns th largest number 99% of the time, but 1% of the time the algorithm fails and returns the smallest numbe i
- In an algorithm imposes a requirement on it's inputs (called a precondition). That requirement must be met. e.g A prepcondition might be that an algorithm will only a positive numbers as an input. If precondition aren't met the algorithm is called to fail by producing the corron answer of never terminating.

→ From data structure point of view, following are sar important propeties of algorithm.

- Search :- Algorithm to search on item in a data struct
- Sort :- Algorithm to sort item in certain order.
- Insert :- Alogrithm to insert item in a data structur
- Update :- Alogrithm to update an existing item in a da structure.
- Delete :- Algorithm to delete on existing item form data structure.

Properties of an algorithm :-

Not all procedures can be called an algorithm. An algorithm should have the below mentioned properties:-

- Unambiguous :- Algorithm should be clear and unambiguous. Each of it's steps (or phase), and their inputs/outputs should be clear and must lead to only one meaning.
- Input :- An algorithm should have 0 or more well defined input.
- Output :- An algorithm should have 1 or more well defined outputs, and should match the desired output.
- Finiteness :- Algorithms must terminate after a finite number of steps.
- Feasibility :- Should be feasible with the available resources.
- Independent :- An algorithm should have step-by-step directions which should be independent of any programming code.

b] Write an algorithm for searching the element in an array.

→ To search any element present inside the array in C++ programming using linear search technique, you have ask to the user to enter the array size and array elements to store the elements in the array. Now ask to the user to array elements to store the elements in the array. Now ask to the user to enter the element that he/she wants to check or search whether the entered number/element is present in the array or not.

- To check/search for the element, just compare with the number to each element present in the array if any element equals the entered number then print the exact position of the number in the array as known here in the following program.
- Following C++ program first ask to the user to enter the array size then it will ask to enter the array elements, then it

finally ask to enter a number to be sarch in the give array to check whether it is present in the array or no it is present then or which position.

EXAMPLE:-

```
#include <iostream.h>
#include <conio.h>
 void main ()
{
 clrscr();
 int arr[10], i, num, n, c=0, pos;
 cout << "Enter the array size: ";
  cin>>n
  cout<<"Enter Array Elements :";
  for (i=0; i<n ; i++)
  {
   cin >> arr[i];
   }
   cout<<"Enter the number to be search:";
   cin >> num;
    for (i=0; i<n; i++)
   {
    if (arr[i]==num)
    {
     c=1;
     pos= i+1;
     break;
    }
   }
   if (c=0)
   {
    cout<<"Number not found ---!!";
   }
   else
```

```
    {
      cout<<num<<" found at position" <<pos;
    }
    getch();
    }
```

ARRAY:-

when the above c++ program is compile and execut ed, it will produce the following result

```
Enter the array size:5
Enter the array Elements:.23
34
45
56
67
Enter the number to be search :.45
45 found at position 3;
```

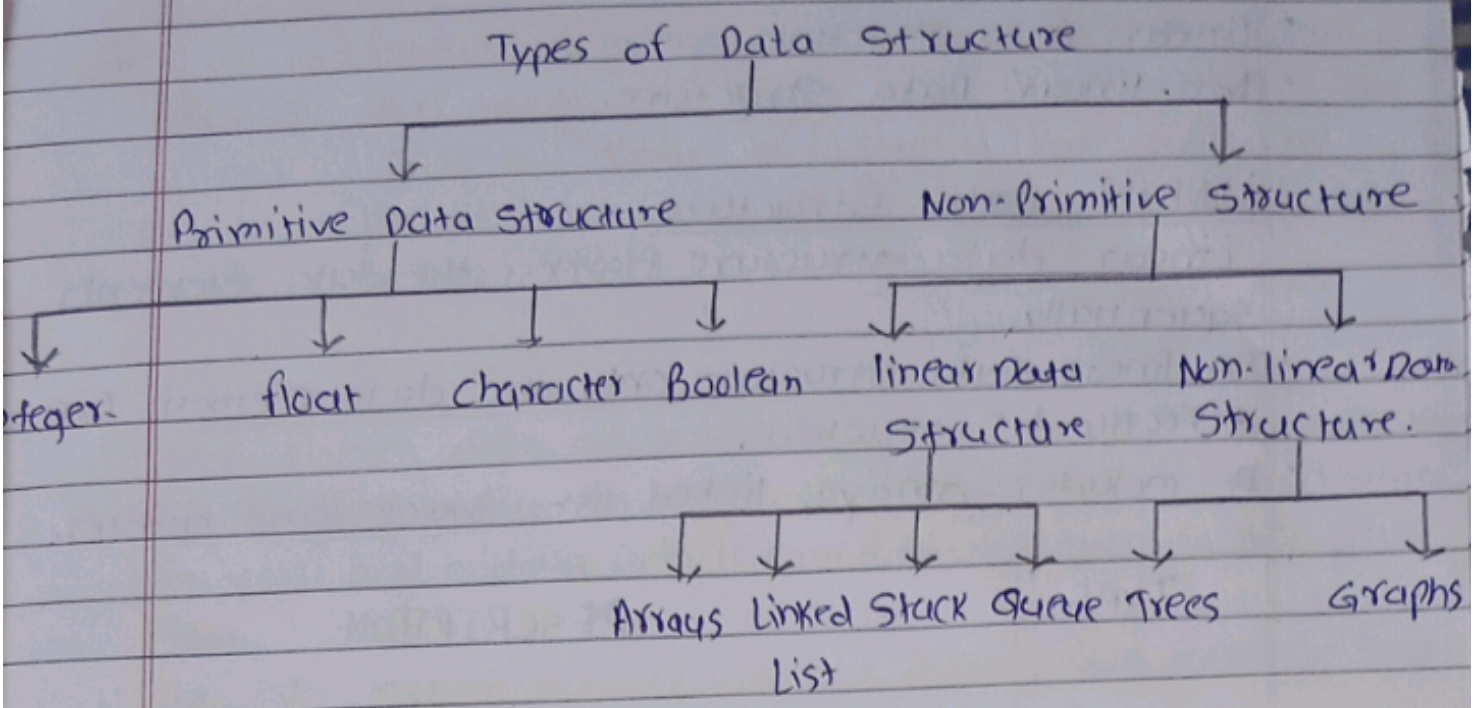
c] What is data structure? Explain primitive and non-primitive data structure.

→ A data structure is a specialized format for organizing storing data. General data structure types include array, the file, the record, the table, the tree and so on. A structure is designed to organize data to suit a specific so that it can be accessed and worked with in appropriate ways. In computer programming a data structure may selected or disigned to store data for the purpose working on it with various algorithms.

- Data structure is a way of collecting and organising d in such a way that we can perform operations on data elements in terms of same relationship, for bet organization and storage. Eg:- we have data playe name "virat" and age as 26 Here "virat" as a str data and 26 is of integer data type.
- We can organize the data as a record like player Now we can collect and store player's records in a fi database as a data structure. e.g:- "Dhon", 30, "G "sehwag" 33.
- In simple language, Data structure are structure programmed to store ordered data, so that variou operations can be performed on it easily.

Primitive and Non-primitive data structure:-

The data structure that are atomic (indivisible) are calle primitive. Examples are integer, real and character Data structures that are not atomic are called no primitive or composite. Examples. records, arra string.



- Primitive data types are the avaible in most of the programming languages.
- These data types are used to represent single value.
- It is a basic data type available in most of programming language.

TYPE	DESCRIPTION
Integer	Used to represent a number without decimal point
Float	used to represent a number with decimal point.
Character	used to represent a single character.
Boolean	used to represnt logical values either to true or false.

- Data type detived from primay data typescare knowas
- Non-primitive data types.

Non-primitive data types are used to some group of values.

- It can be divided into two types.

- Linear Data structure.
- Non-linear Data structure.

LINEAR Data Structure:-

Linear data structure traverses the data elements sequentially.

- In linear data structure, only one data element can directly be reached.
- It includes arrays, linked list, stack and queues.

TYPE	DESCRIPTION
Arrays	Array is a collection of elements. It is used in mathematical problems like matrix, algebra etc. Each element of an array is referred by a subscript variable or value, called subscript or index enclosed in parenthesis.
Linked list	linked list is a collection of data structures/ elements. It consists of two parts. Info and link. Info gives information and Link is an address or next node. Linked list can be implemented by using pointers.
Stack	Stack is a list of elements, In stack, an elem may be inserted or deleted at one end whi as known as top of the stack. It performs to operations, push and top. Push means adding element in stack and Pop means removi an element in stack. It is also called Last-in-first-out (LIFO).

QUEUE	queue is a linear list of element. In queue, elements are added at time one end called rear and the existing elements are deleted from other end called front. It is also called as first-in-First-out. (FIFO).

Non-Linear ~~Dat~~ Data structure:

Non-Linear data structure is opposite to linear data structure.

- In non-linear data structure, the data values are not arranged in order and a data item is connected to several other data ~~items~~ items.
- It uses memory efficiently. free contiguous memory is not required ~~for~~ for allocating data items.
- It includes tree and graphs.

TYPE	DESCRIPION.
Tree	Tree is a flexible, versatiable and powerful non-linear data structure. It is used to represent data items processing hierarchich relationship between the grand father and his children and grandchildren It is an ideal data structure for repressing hietarchical data.
Graph	Graph is a non-linal data structure which cons of a finite set of ordered pairs called edges. graph is a set of elements connected by edges. Each elements are called a vertex and node.

d) What is time and space complexity? Explain Big o and
Theta Notation.

→ Time complexity:-

- Time complexity of an algorithm represents the amou
time required by the algorithm to run to completion. Time
requirements can be defined as a numerical function
where T(n) can be measured as the number of steps, pr
each step consumes constant time.
- For example:- addition of two n-bit integers takes n s
Consequently, the total computational time is T(n) = c*n
Where c is the time taken for addition of two bits. Her
observe that T(n) grows linearly as input size in
- Asymptotic analysis of an algorithm, refers to defini
mathematical foundation framing of it's run-time perfor
Using asymptotic analysis, we can very well conclude the
case, average case and worst case scenario of an algo
- Asymptotic analysis are input bound i.e., if there is no inp
the algorithm it is concluded to work in a constant ti
other than the "input" all other factors are considere
- Asymptotic analysis refers to computing the running ti
any operation in mathematical units of computation the
running time of any operation in mathematical units o
computation. e.g:-, running time of one operation it is c
as g(n2). which means first operation running time
second operation will increase exponentially whe
increase, similarly, the running time of both op
will be nearly same if n is significantly small
Usually, time required by an algorithm falls un
three types:-

- Best case- minimum time required for program ex
- Average case- Average time required for program
execution.

- Worst Case. Maximum time required for program execution

SPACE COMPLEXITY:-

Space complexity of an algorithm represents the amount of memory space required by the algorithm in its life cycle. Space required by an algorithm is equal to the sum of the following two components:-

- A fixed part that is a space required to store certain data and variables that are independent of the size of the problem. e.g:- simple variables constant used, program size etc.
- A variable part is a space required by variables, whose size depends on the size of the problem. e.g:- dynamic memory allocation, recursion stack space etc.

Space Complexity S(P) of of any algorithm P is ~~P~~ S(P) = C+SP(I) where C is the fixed part and ~~P~~ S(I) is the variable part of the algorithm which depends on instance characteristic I following is a simple example that tries to explain the concept.

Algorithm:- SUM(A,B)
Step 1:- START
Step 2:- C ← A+B+10
Step 3:- Stop.

Here we have three variables A, B and C and one constant. Hence S(P) = 1+3. Now space depend on data types of given variables and constant types and it will be multiplied accordingly.

Big O and Big Theta Notation:-

· Big-O mega :- Big-omega, commonly written as Ω.

→ The order of growth of an algorithm is defined by using the Big a nation. The big o nation has been accepted as a fundamental technique for describing the efficiency of an algorithm.

The following table lists some possible orders of growth and their corresponding big O notations.

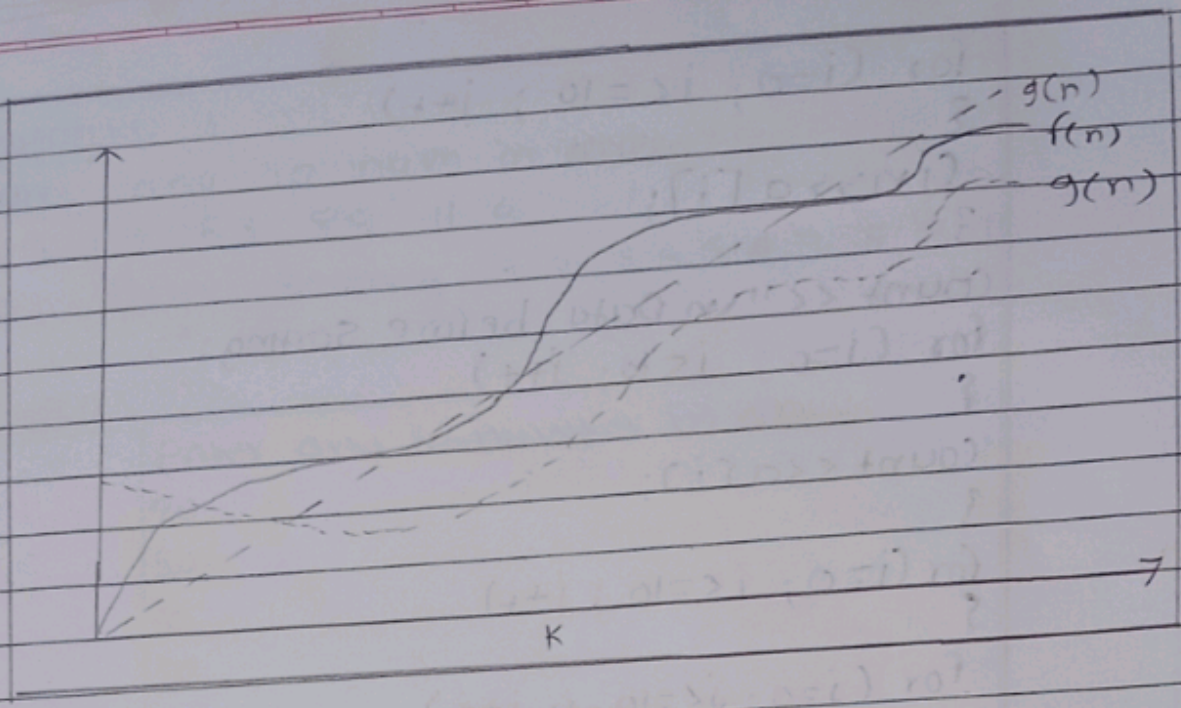
order of Growth	Big a notation
Constant	O(1)
Zelogarithmic	O(log n)
Linear	O(n)
Loglinear	O(n log n)
Quadratic	O(n²)
Cubic	O(n³)
Exponential	O(2ⁿ), O(10ⁿ).

→ If an algorithm has a linear order of growth, the algo is said to be of the order O(n). Similarly, If an algor has a quadratic order of growth, the algorithm as said to be of the order O(n²).

THETA NOTATION:-

The notation Θ(n) is the formal way to express b the lower bound and the upper bound of an algorith running time. It is represented as follow.

Θ(f(n) = {g(n) is and only if g(n) = O(f(n)) g(n) = Ω (f(n)) for all Dzno. }



Q.] Write an algorithm for sorting the elements of an array.

-> Sorting of ARRAY :-

Sorting is the process of putting data in order either numerically or alphabetically. It is often necessary to arrange the elements in an array in numerical order from highest to lowest values (descending order) or vice versa (ascencling order). If the array contains string values, alphetical order way be needed (which is actually ascending order using ASCII Values).

C++ program to sort elements of Array in Ascending order

```
#include <iostream.h>
#include <conio.h>

void main()
{
int i, a[10], temp, j;
clrscr();
cout<< "Enter any 10 num is array \n";
```

```
for (i=0; i<=10 ; i++)
{
cin>>a[i];
}
count<<"\n Data before sorting;"
for (j=0; j<10; j++)
{
count<<a[i];
}
for (i=0; i<=10 ; i++)
{
 for (j=0; j<=10-j; j++)
{
 if (a[i]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
Cout<<"\n Data aftersorting:";
for (j=0; j<10; j++)
{
Cout<<a[i];
}
getch();
}
```

Output:-
Enter any 10 num in array:-
2 5 1 7 53 89 11 4
Dat After Sorting. 1 2 3 4 5 7 7 8 9 11

Output:-

Enter any 10-number in array.
20
50
10
70
60
33
86
94
11
42
Data Before sorting:- 20 50 10 70 60 33 86 94 11 42
Data Before Sorting:- 10 11 20 33 42 50 60 70 86 94

f] ~~Go~~ Write an algorithm for merging two arrays.

-> To merge two arrays in C++ programming you have to ask to the user to enter the array the 1 size and elements then array2 size and elements to merge both the array and store the merged result in the third array say merge[].

- So to merge two array, start adding the element of first array to the third array (target array) after this start appending the elements of second array to the third

array (target array) as shown here in the following prog
- In this program we enter an elements in any two ar
and then these two array (elements of array) are sto
in third array.

Example:-

```
#include <iostream.h>
#include <conio.h>

Void main()
{
int a[10], b[10], c[20], i;
clrscr();
cout<<"Enter Elements in 1st Array:";
for(i=0; i<10; i++)
   {
cin>>a[i];
Array
   }
cout<<"Enter Elements in 2nd Array:";
for(i=0; i<10; i++)
   {
cin >>b[i];
   }
cout<<"\n Elements of Array After Merge:\n";
for(i=0; i<10; i++)
   {
c[i] = a[9];
c[i+10]= b[i];
   }
for(i=0; i<20; i++)
```

```
{
cout << c[i];
  3
getch();
  3
```

Output:

```
Enter Element in 1st array 11
2
3
4
5
6
7
8
9
10
Enter element in 1st array 2
3
4
5
6
7
8
9
2
```

1

Element of 1st Array 1 1 2 3 4 5 6 7 8 9 10

Element of 2nd Array 2 3 4 5 6 7 8 9 2 1

Element of Array Merge 1 1 2 3 4 5 6 7 8 9 10 2
3 4 5 6 7 8 9 2 1