Q1. what is data structure.

Data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.
A data structure is a particular way of organizing data in a computer so that it can be used effectively.

Data Structure is used for organizing the data in memory. There are various ways of organizing the data in the memory for eg. array, list, stack, queue and many more.
Data structure isn't a programming language like C, C++, java, etc. It is a set of algorithms that can be used in any programming language to organize the data in the memory.
'n' number of algorithms were proposed to organize the data in memory. These algorithms are referred to as Abstract data types. Abstract data types are nothing but a set of rules.

Note :-
[

- They are mainly of two types:

1. Linear Data structure: Here the Data elements are organised in a sequence of some manner.

2, Non-linear Data structure: Here the data is ordered in any arbitrary order and not in a sequence.

- Some common data structures are
    - Linked list
    - Arrays
    - Stacks
    - Queues
    - Binary trees
    - Hash tables

- Areasof Application

    - Data structures are used in any program or software.
    - They are used in the areas of
    - Compiler Design
    - Operating System
    - DBMS
    - Graphics
    - Simulation
    - Numerical Analysis
    - Artificial Intelligence

]

Q2. Need of Data Structure –

As applications are becoming more complex and the amount of data is increasing day by day, which may cause problems with processing speed, searching data, handling multiple requests etc. Data structure provides a way of organizing, managing, and storing data efficiently. With the help of data structure, the data items can be traversed easily. Data structure provides efficiency, reusability and abstraction. It plays an important role in enhancing the performance of a program because the main function of the program is to store and retrieve the user’s data as fast as possible.

Q3. Types of Data Structure –

There are 2 types of Data Structure :

1. Primitive Data Structure

2. Non – Primitive Data Structure

1. Primitive Data Structure –

Primitive Data Structures directly operate according to the machine instructions. These are the primitive data types.
Data types like int, char, float, double, and pointer are the primitive data structures that can hold a single value.

2. Non – Primitive Data Structure –

Non-primitive data structures are complex data structures that are derived from primitive data structures.
Non – Primitive data types are further divided into two categories.

1. Linear Data Structure
2. Non – Linear Data Structure

1. Linear Data Structure –

Linear Data Structure consists of data elements arranged in a sequential manner where every element is connected to its previous and next element. This connection helps to traverse a linear arrangement in a single level and in a single run. Such data structures are easy to implement as memory is additionally sequential. Some examples of Linear Data Structure are List, Queue, Stack, Array etc.

- Types of Linear Data Structure –

1] Arrays –

An array is a collection of similar data elements stored at contiguous memory locations. It is the simplest data structure where each data element can be accessed directly by only using its index number.

- Applications of Array:

i. Contacts of a mobile phone
ii. Storing data in a tabular format
iii. Storage of matrices and binary tree elements of fixed count
iv. Building block element of other data structures such as heaps, vectors and more
v. Online ticket booking system — if a user wants to book a seat in C-4, the array becomes seat[C][4] or seat[3][4]

2] Linked List –

Linked list is a linear data structure which is used to maintain a list-like structure in the computer memory. It is a group
of nodes that are not stored at contiguous locations. Each node of the list is linked to its adjacent node with the help of pointers.

1. - Applications of Singly Linked List:

i. Prevent collision between data in a hash map
ii. UNDO, REDO or DELETE operations in a notepad
iii. Photo viewer to look at photos continuously in a slide show
iv. If one wants to add a bogie, they can either take a new bogie to add at the last or in between two bogies.
v. The next track feature of a music player

2. Doubly Linked List: a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.

- Applications of Doubly Linked List:

i. Represent deck of cards in games
ii. Used to represent various states of a game
iii. UNDO or REDO function
iv. Used by browsers to implement backward and forward navigation of the visited web pages
v. The next track and previous track feature of a music player

3. Circular Linked List: the last node of the list contains a pointer to the first node of the list. We can have circular singly linked list as well as circular doubly linked list.

- Applications of Circular Linked List:

i. All the running applications in an Operating System are kept in a circular linked list and the Operating System gives a fixed time slot to all for running.
The Operating System iterates the list over and over again until all the applications get completed
ii. In Role Based Multiplayer games, all the players are kept in a circular linked list and the pointer keeps moving forward as a player's chance ends
iii. Snake game in mobile phones, where head of the list is the snake's head and tail of the list is the snake's tail
iv. The repeat feature in a music player wherein a user will continuously listen to the playlist on repeat that is, when the songs of the playlist get over,
the first song is played

3] Stack –

Stack is a linear data structure that follows a specific order during which the operations are performed. The order could be
FILO (First In Last Out) or LIFO (Last In First Out).

- The basic operations performed in stack are as follows :

Push – Adds an item within the stack.
Pop – Deletes or removes an item from the stack.
Top – Returns the topmost element of the stack.
IsEmpty – Returns true if the stack is empty.

- Applications of Stack:

i. UNDO and REDO functions in a text editor
ii. Virtual Machines
iii. Expression Conversion (Infix to Postfix and vice versa)
iv. Reversal of a string
v. Back/Forward button in browsers and file browsers

4] Queue –

Queue is a linear data structure in which elements can be inserted from only one end which is known as rear and deleted from another
end known as front. It follows the FIFO (First In First Out) order.

Deque – Adds an element to the queue.
Enqueue – Deletes or removes an element from the queue.
IsFull – Returns true if the queue is full.
IsEmpty – Returns true if the queue is empty.

- Applications of Queues:

i. Processing requests on a single shared resource such as a printer, CPU task scheduling
ii. In a call center, queues are used to hold people calling them in an order until a service representative is free
iii. Handling of interrupts in a real-time system
iv. Priority queues used in file downloading operation of a browser

2. Non – Linear Data Structure –

Non-linear Data Structures do not have any set sequence of connecting all its elements and every element can have multiple paths to
attach to other elements. Such data structures support multi-level storage and sometimes can't be traversed in a singl e run. Such data structures
aren't easy to implement but are more efficient in utilizing memory. Some examples of non-linear data structures are Tree, BST, Graphs etc.

- Types of Non – Linear Data Structure –

1] Tree –

Tree is a multilevel data structure defined as a set of nodes. The topmost node is named root node while the bottom most nodes are
called leaf nodes. Each node has only one parent but can have multiple children.

- Types of Tree in Data structure

1. General Tree
2. Binary Tree
3. Binary Search Tree
4. AVL Tree
5. Red Black Tree
6. N-ary Tree

- Applications of Trees:

- i. In computer systems, directory and file systems
- ii. Implementation of navigation structure of a website
- iii. Decision making in video games
- iv. Path Finding Algorithms which are then implemented in Artificial Intelligence, Robotics and Video Games

2] Graph –

A graph is a pictorial representation of a set of objects connected by links known as edges. The interconnected nodes are
represented by points named vertices, and the links that connect the vertices are called edges.

- Types of Graph

Finite Graph
Infinite Graph
Trivial Graph
Simple Graph
Multi Graph
Null Graph
Complete Graph
Pseudo Graph
Regular Graph
Bipartite Graph
Labelled Graph
Diggraph Graph
Subgraph
Connected or Disconnected Graph
Cyclic Graph
Vertex Labelled Graph
Directed Acyclic Graph

A graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges.

- Applications of Graphs:

i. Resource utilization and availability in an organization
ii. Interconnections in Social Media and other Network Based platforms
iii. Ecommerce applications where user preferences are set
iv. Shortest path from Point A to Point B can be found with the help of certain algorithms

Q4. - Differences between Linear Vs Non-linear Data Structures

Now that we know about linear and non-linear data structures, let's see the major differences between them.

Linear Data Structures	Non Linear Data Structures
1. The data items are arranged in sequential order, one after the other.	1. The data items are arranged in non-sequential order (hierarchical manner).
2. All the items are present on the single layer.	2. The data items are present at different layers.
3. It can be traversed on a single run. That is, if we start from the first element, we can traverse all the elements sequentially in a single pass.	3. It requires multiple runs. That is, if we start from the first element it might not be possible to traverse all the elements in a single pass.
4. The memory utilization is not efficient.	4. Different structures utilize memory in different efficient ways depending on the need.
5. The time complexity increase with the data size.	5. Time complexity remains the same.

6. Example: Arrays, Stack, Queue	6. Example: Tree, Graph, Map

Q5. Classification of Data Structure –

Data Structure can be further classified as

1. Static Data Structure
2. Dynamic Data Structure

1. Static Data Structure –

Static Data Structures are data structures where the size is allocated at the compile time.Hence, the maximum size
is fixed and cannot be changed.

2. Dynamic Data Structure –

Dynamic Data Structures are data structures where the size is allocated at the run time. Hence, the maximum size
is flexible and can be changed as per requirement.

Q6. Data Structure Operations –

The common operations that can be performed on the data structures are as follows :

1. Searching – We can easily search for any data element in a data structure.
2. Sorting – We can sort the elements either in ascending or descending order.
3. Insertion – We can insert new data elements in the data structure.
4. Deletion – We can delete the data elements from the data structure.
5. Updation – We can update or replace the existing elements from the data structure.

Q7. Advantages of Data Structure –

- Data structures allow storing the information on hard disks.
- Appropriate choice of ADT (Abstract Data Type) makes the program more efficient.
- Data Structures are necessary for designing efficient algorithms.
- It provides reusability and abstraction .
- Using appropriate data structure, can help programmers save a good amount of time while performing operations such as storage, retrieval or processing of data.
- Manipulation of large amounts of data is easier.