1.What is a data structure?

A data structure is a mechanical way to organise, align, and manipulate data as per requirements. It is not restricted to putting data in a table but deals with different datasets and how well they are aligned.

The aim is to ensure that data can be organised and accessed efficiently. Data organisation determines how a program performs. Moreover, data dependency and relationships between two or more datasets play a crucial role in data structures.

While designing code, we need to pay utmost attention to how data is structured because incorrectly structured or inefficiently stored data can hamper the overall performance of the code.

2. What are the applications of data structures?

Data structures are applied across multiple industries and domains as algorithms are the primary requirements in every business. Here are the major applications of data structures:

- Numerical and statistical analysis
- Compiler design
- Artificial intelligence
- Database management
- Graphical processing
- Lexical analysis
- Blockchain
- Decision making
- Operating system design
- Simulation

3. Describe the types of data structures.

Data structures are divided into two categories:

- Linear data structure
- Non-linear data structure

Linear Data Structure

It includes data elements that are arranged sequentially or linearly. Also, each element is connected to its previous or next adjacent element.

Types of linear data structure

- **Static Data Structure:** It has a fixed memory size, and it's easier to access data elements. An example includes Array.
- **Dynamic Data Structure:** Size is not fixed and can be updated randomly. Examples include Queue and Stack.

Common linear data structures

- **Array Data Structure:** All the elements or values are of the same type.
- **Stack Data Structure:** It follows LIFO (Last in, first out) order to perform operations, so the last element to be stored will be the first to be retrieved.
- **Queue Data Structure:** It follows FIFO (first in, first out) order for operations, so the item stored first will be the first to be accessed.
- **Linked List Data Structure:** Elements in this data structure are connected through a series of nodes, each containing data items and the address of the next node.

Non-linear Data Structure

Here, data elements are not arranged in a sequence, and all elements can't be traversed in a single run. Common non-linear data structures are:

- **Tree Data Structure:** A hierarchical data structure where elements are placed in a tree-like structure. It contains a central node, structural nodes, and sub-nodes connected via edges. Tree-based data structures are Binary Tree, Binary Search Tree, B- Tree, B+ Tree, AVL Tree, and Red-Black Tree.
- **Graph Data Structure:** Consists of vertices and edges, and a pair of nodes is connected through edges. It is used to solve complex problems.

4. Explain the difference between a File Structure and a Storage Structure.

The difference is that the storage structure has data stored in the memory of the computer system, whereas the file structure has the data stored in the auxiliary memory.

File Structure	Storage Structure

Represent data in auxiliary memory, such as disk and pen drive.	Data is stored in the computer me
Data is intact until deleted manually.	Once the function using the data g
Standard storage policies.	Customized storage policies.
Compatibility with external apps is low.	Highly compatible with external a

It is one of the most asked data structure interview question for freshers.

5. What is a stack data structure? What are the applications of a stack?

A stack data structure represents the state of an application at a specific point in time. It consists of a series of elements arranged in a linear structure like a real physical stack or pile where you add or removes items from the top.

This linear data structure follows a particular order, LIFO (Last In First Out) and FILO (First In Last Out), to perform operations.

Elements are arranged in a sequence, so the item added last will be restricted first. A real-life example is a stack of clothes where the cloth placed on the top, or the last to be added, will be the first to be picked.

Applications of a stack data structure are:

- Temporary storage during recursive operations
- Reversing a string
- Redo and Undo operations in doc editors
- Parenthesis matching
- Syntax parsing
- Postfix, prefix, and infix expressions
- Backtracking

6. What operations can be performed on a stack?

Common operations that can be performed in a stack data structure are:

a) push():

It inserts a new element at the top of the stack. However, before we insert a value, it is important to verify if TOP=MAX–1. If the stack is filled, no more insertion is possible, and you can see an OVERFLOW message if y try to put an element in the already-filled stack.

b) pop():

Its basic function is to remove an element from the top of the stack. Before you retrieve an item, verify if TOP=NULL. If the stack is empty, no further deletions are

allowed, and you will get an UNDERFLOW message if you try to remove an element from an empty stack.

c) peek():

It returns a value of the top-most element in the stack but doesn't remove it from the data elements collection.

d) isFull():

To check if the stack has reached its maximum capacity of data insertion.

e) isEmpty():

Checks if the stack is empty or not.

f) size():

To find the number of elements in a stack.

7. Explain what multidimensional arrays is.

A multidimensional array spans across more than one dimension, which means it has more than one index variable for each storage point. It has multiple layers and is primarily used where data can't be stored or represented in one dimension.

Basic multidimensional arrays are 2D and 3D arrays. It's technically an array of different arrays. A 2D array, also known as a matrix or a table, consists of rows and columns.

8. What is a binary search tree?

It is a type of data structure that stores data elements efficiently. Here, each node stores a key and a value. The former is used to access an item, while the latter is used to find if the item is present or not.

A key is constant and defines the dataset (e.g., colour, gender, price), while a value is variable and belong to any of the dataset (e.g., red, female/male, 100).

A binary search tree has a specific order to place elements and is known for three qualities:

- The values of elements of nodes in the left sub-tree are less or equal to that of the root nodes.
- Moreover, the values of the right root nodes are higher or equal to the root node.
- Both the left and right sub-tree are two individual binary search trees at all points in time.

Major applications of binary search trees are:

- For indexing and multi-level indexing
- For organising and sorting data
- For implementing different search algorithms

9. What is a linked list data structure?

A linked list data structure is a sequence of elements that are not stored in adjacent memory locations. The elements in this data structure use pointers to form a chain. Think of it as a series of linked nodes (each element or node representing a separate object) connected by paths or links. Every node has two items- a data field and a reference to the next node.

Each link is an entry into the linked list, and every entry point, also known as the head, points to the next node. In an empty list, the head is a null reference, with the last node consisting of a reference to the null.

The order to add nodes to the list is based on the order in which they are created. A linked list data structure doesn't have a fixed number of nodes, and the list can grow or shrink as per demand.

Some of the key applications of a linked list data structure are as follows:

- Dealing with the unknown number of elements
- Implementing queue, stack, graphs, and binary trees data structures
- Accessing elements randomly is not required
- Round-robin scheduling for an operating system
- Real-time computing when time predictability is important
- Forward and backward operation in the browser
- Interesting items in the middle of the list
- Dynamic management for Operating System memory

10. Are linked lists linear or non-linear data structures?

Linked lists can be linear and non-linear data structures based on their usage. If it's used for storage, it's non-linear, but if it's used for access or retrieval strategies, it's considered a linear data structure.

11. What are the pros of a linked list over an array? In which scenarios do we use a linked list and when array?

The benefits of a linked list over an array are:

a) Dynamic Data Structure

The linked list is a dynamic data structure, so there is no need for an initial size. It grows and shrinks at runtime through memory allocation and deallocation.

In an array, size is limited because the number of elements is stored statistically in the memory.