**Types of data:**

1. **Static data:** Static data remains the same after you have recorded it. It is not subject to change or is rarely changed. Eg. Name of continents, scientific terms, mathematical constants, etc.
2. **Dynamic data**: Dynamic data is subject to change after initial recording. Eg. Bank transaction, inventories, production quotas, etc.

**Linked list:**

* Most popular type of data structure for handling dynamic data elements.
* It is a linear data structure that stores collection of data elements dynamically.
* Each data element is called a node.
* Each node consists of two fields, the information stored in a linked list and a pointer that stores the address of its next node.
* The last node contains null in its second field because it will point to no node.
* A linked list can grow and shrink its size, as per the requirement.
* It does not waste memory space.

**Types of linked lists**

1. **Singly linked list –** Each node has data and an address field that contains reference to the next node.
2. **Doubly linked list –** Each node has two pointers. First one stores the address of previous node and second contains the address of second node. Thus, you can go in both the directions.
3. **Circular linked list –** Same as singly linked list but last node contains the address of the first node.

**Operations of linked list**

1. **Traversing:** Transvers all the nodes one by one.
2. **Insertion:** Insert new node at specific position
3. **Deletion:** To delete node at specific position.
4. **Searching:** To search for an element from the linked list.

**Examples:**

1. Blockchain is a type of linked list.
2. Browser history

**Stack**

* Stacks in Data Structures is a linear type of data structure that follows the LIFO (Last-In-First-Out) principle and allows insertion and deletion operations from one end of the stack data structure, that is top.
* Implementation of the stack can be done by contiguous memory which is an array, and non-contiguous memory which is a linked list.

**Operations of Stack**

1. **Push -** Push operation involves inserting new elements in the stack. Since you have only one end to insert a unique element on top of the stack, it inserts the new element at the top of the stack
2. **Pop -** Pop operation refers to removing the element from the stack again since you have only one end to do all top of the stack. So removing an element from the top of the stack is termed pop operation.
3. **Peek -** Peek operation refers to retrieving the topmost element in the stack without removing it from the collections of data elements

**Example:**

1. Deck of cards
2. Pile of books
3. Browser stack

**Queue**

* Queue in data structures is a linear collection of different data types which follow a specific order while performing various operations.
* It can only be modified by the addition of data entities at one end or the removal of data entities at another.
* By convention, the end where insertion is performed is called **Rear**, and the end at which deletion takes place is known as the **Front.**
* This is equivalent to the requirement that once an additional data element is added, all previously added elements must be removed before the new element can be removed.

**Operations of queue**

1. **Enqueue**- Insertion of elements to the queue.
2. **Dequeue** - Removal of elements from the queue.
3. **Peek** - Acquires the data element available at the front node of the queue without deleting it.
4. **isFull** - Validates if the queue is full.
5. **isNull** - Checks if the queue is empty.

**Example:**

1. **Printer:** Queue data structure is used in printers to maintain the order of pages while printing.
2. **Process scheduling in Operation system:** Queues are used to implement round-robin scheduling algorithms in computer systems.
3. **Switches and routers:** Both switch and router interfaces maintain ingress (inbound) and egress (outbound) queues to store packets.