Array Representation and Implementation of Stack:

Representation: Stack can be implemented using arrays where elements are stored sequentially, and a pointer indicates the top of the stack.

Operations:

Push: Add an element to the top of the stack.

Pop: Remove the top element from the stack.

Linked Representation: Stack can also be implemented using linked lists where each node points to the next node, simulating the behavior of a stack.

Application of Stack:

Conversion of Infix to Prefix and Postfix Expressions: Stack-based algorithms can convert infix expressions to prefix or postfix notation, simplifying expression evaluation.

Evaluation of Postfix Expression using Stack: Postfix expressions can be evaluated efficiently using stacks by pushing operands onto the stack and performing operations when encountering operators.

Recursion:

Recursive Definition and Processes: A function is said to be recursive if it calls itself either directly or indirectly. Recursion involves breaking a problem into smaller, similar subproblems.

Recursion in C: C programming language supports recursion where a function calls itself.

Example of Recursion: Common examples include factorial calculation, Fibonacci sequence generation, and binary tree traversal.

Tower of Hanoi Problem: A classic problem that can be solved recursively, involving moving disks from one rod to another, obeying certain constraints.

Queues:

Array and Linked Representation: Similar to stacks, queues can be represented and implemented using arrays or linked lists.

Operations on Queue:

Create: Initialize an empty queue.

Add (Enqueue): Insert an element at the rear of the queue.

Delete (Dequeue): Remove an element from the front of the queue.

Full and Empty: Check if the queue is full or empty.

Circular Queues, D-queues, and Priority Queues: Variations of queues that offer specific functionalities such as circular buffer implementation, double-ended queues, and queues where elements have priorities.