Chapter Three
Data Structures and Operations

Data Structure:-Particular way of organizing similar or dissimilar logically related data items which can be processed as a single unit.

1.Simple Data Structure:-Arrays and Structures

2.Compound Data Structures: Simple Data Structures are combined together to form complex data structures.

a. Linear:-The elements are arranged in a linear format.1. Stack 2. Queue 3. Linked list

b. Non Linear:-Tree and Graph

Operations of Data Structures:

1. Traversing:-visiting each element in a data structure.
2. Searching:- finding the location of a particular element .
3. Inserting:- adding a new element .
4. Deleting:- removing an existing element.
5. Sorting:- arranging the elements in a specified order.
6. Merging:-combining the elements of two data structures and to form a third one.

Stack: It is a linear data structure in which the insertion and deletion takes place at one end of the stack called top of the Stack. The organizing principle is **LIFO(Last In First Out).**The process of adding new elements into a stack is called **Pushing**. The process of adding new elements into a full stack is called **Stack Overflow**. The process of removing existing elements from a stack is called **Popping.** The process of removing elements from an empty stack is called **Stack Underflow.**

Algorithm to Add an Item(PUSH)

start

1. if (TOS<N-1) Then
2. TOS=TOS+1
3. STACK[TOS]=VAL
4. Else
5. Print “Stack Overflow”
6. End of if

stop

Algorithm to Delete an Item(POP)

start

1. if (TOS>-1) Then
2. VAL=STACK[TOS]
3. TOS=TOS-1
4. Else
5. Print “Stack Underflow”
6. End of if

stop

Queue:- It is a linear data structure in which insertion will take place at one end called **rear** end and deletion will take place at other end called **front** end of the queue. Here the organizing principle is **FIFO(First In First Out)**.The process of adding new elements into a queue is called **insertion** and the process of removing elements from a queue is called **deletion**. The process of adding new elements into a full queue is called **Queue Overflow** and the process of removing elements from an empty queue is called **Queue Underflow.**

Algorithm to add an Item

start

1: if (REAR==-1) Then
2: FRONT=REAR=0
3: Q[REAR]=VAL
4: Else If(REAR<N-1) Then
5: REAR=REAR+1
6: Q[REAR]=VAL
7: Else
8: Print “Queue Overflow”
9: End of if

Stop

Algorithm to Delete an Item

start

1: if (FRONT>-1) Then
2: VAL=Q[FRONT]

3: FRONT=FRONT+1
4: Else
5: Print “Queue Underflow”
6: End of if
7: If(FRONT>REAR) Then
8: FRONT=REAR=-1
9: End of if

Stop

Circular Queue: A queue in which two end points meet.

Linked List:-a collection of nodes, where each node consists of a **data and a link** – a pointer to the next node.

Operations on Linked List

a. Creation of Linked list

steps

1. Create a node and obtain its address.
2. Store data and NULL in the node.
3. If it is the first node, store its address in start.
4. If it is not the first node,store its address in the link part of the previous node.
5. Repeat the steps 1 to 4 as long as the user wants.

b. Traversing a linked list

steps

1. Get the address of the first node from start and store it in temp.
2. Using the address in Temp, get the data of the First/next node and store in Val
3. Also get the content of the link part of this node (i.e the address of the next node) and store it in Temp.
4. If the content of Temp is not NULL, goto step 2 otherwise stop.

c. Insertion in a linked list

steps

1. Create a node and store its address in Temp
2. Store the data and link part of this node using Temp.

3. Obtain the addresses of the nodes at positions (POS-1) and POS in the pointers PreNode and PostNode respectively, with the help of a traversal operation.
4. Copy the content of Temp (address of the new node) into the link part of node at position (POS-1),which can be accessed using PreNode.
5. Copy the content of PostNode (address of the node at position POS) into the link part of the new node that is pointed to by Temp.

d. Deletion from a linked list

steps

1. Obtain the addresses of the nodes at positions (POS-1) and (POS+1) in the pointers PreNode and PostNode respectively, with the help of a traversal operation.
2. Copy the content of PostNode (address of the node at position POS+1) into the link part of the node at position (POS-1), which can be accessed using PreNode.
3. Free the node at position POS.