UNIT- I
Linear Data Structures – LIST

Introduction to Data Structure- Abstract Data Types (ADTs) – List ADT – Array-based implementation – linked list implementation —singly linked lists- circularly linked lists- Doubly-linked lists – Stack ADT- Queue ADT-Applications of Stack-Infix to Postfix conversion, Evaluation of postfix expression.

PART A

1. **Define: data structure.**

A data structure is a way of storing and organizing data in the memory for efficient usage. The way information is organized in the memory of a computer.

2. **Give few examples for data structures?**

Arrays, stacks, queue, list, tree, graph, set, map, table and deque.

3. **What are the different types of data structures?**

i) Primitive
ii) Composite
iii) Abstract

4. **What are primitive data types?**

The basic building blocks for all data structures are called primitive data types.
(e.g) int, float, char, double, Boolean

5. **What are composite data types?**

Composite data types are composed of more than one primitive data type.
(e.g) array,structure,union

6. **What is meant by an abstract data type?**

An ADT is a mathematical model for a certain class of data structures that have similar behavior. (e.g) list, stack, queue.

7. **State the advantage of ADT.**

The advantages of ADT in Data Structures are: Provides abstraction, which simplifies the complexity of the data structure and allows users to focus on the functionality. Enhances program modularity by allowing the data structure implementation to be separate from the rest of the program.

8. **How can we categorize data structures based on data access?**

Linear – list, stack, queue
Non-linear- heap, tree, graph

9. **State the difference between linear and non-linear data structures.**

The main difference between linear and nonlinear data structures lie in the way they organize data elements.

In linear data structures, data elements are organized sequentially and therefore they are easy to implement in the computer's memory.

In nonlinear data structures, a data element can be attached to several other data elements to represent specific relationships that exist among them. Due to this it might be difficult to be implemented in computer's linear memory.

10. **List a few real-time applications of data structures?**

- Undo and redo feature - stack
- Decision making - graph
- Printer (printing jobs) – queue
- Compilers – hash table
- Directory structure- trees
- Communication networks- graphs

11. **Define List.**

The general form of the list is a_1, a_2, a_3... a_n. The size of the list is 'n'. Any element in the list at the position i is defined to be at a_i, a_{i+1} the successor of a_i , and a_{i-1} is the predecessor of a_i. a_1 doesn't have predecessor and a_n doesn't have successor.

12. **What are the various operations done on List ADT?**

The operations done under List ADT are Print list, Insert, Delete, FindPrevious, Find k^{th}, Find, MakeEmpty, IsLast and IsEmpty.

13. **What are the different ways to implement list?**

- Array implementation of list
- Linked list implementation of list
- Cursor implementation of list

14. **Arrays are not used to implement lists. Why?**

- Requires that the list size to be known in advance
- Running time for insertions and deletions is slow

15. **What are the advantages in the array implementation of list?**

- Print list operation can be carried out at linear time
- Finding K^{th} element takes a constant time

16. **What are the disadvantages in the array implementation of list?**

The running time for insertions and deletions is so slow and the list size must be known in advance.

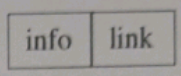
17. **What are the disadvantages of linked list over array?**

Memory usage: More memory is required in the linked list as compared to an array. ...

Traversal: In a Linked list traversal is more time-consuming as compared to an array.

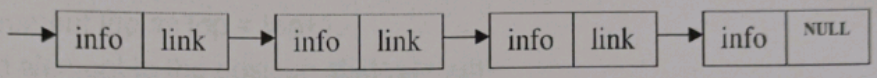
18. **Define node.**

A node consists of two fields namely an information field called INFO and a pointer field called LINK. The INFO field is used to store the data and the LINK field is used to store the address of the next field.



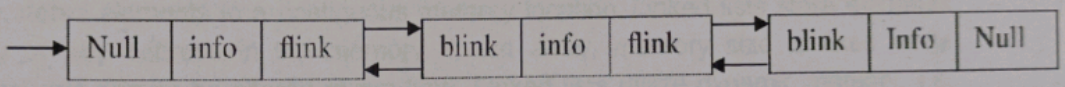
19. **What is a linked list?**

Linked list is series of nodes, which are not necessarily adjacent in memory. Each node contains a data element and a pointer to the next node.



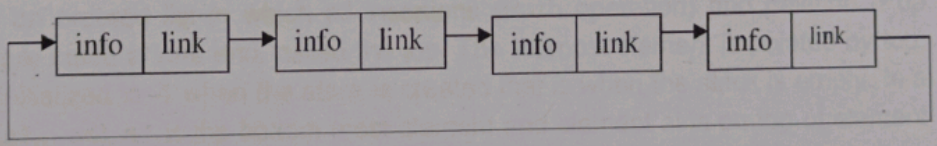
20. **What is a doubly linked list?**

In a doubly linked list, along with the data field there will be two pointers one pointing the next node(flink) and the other pointing the previous node(blink).



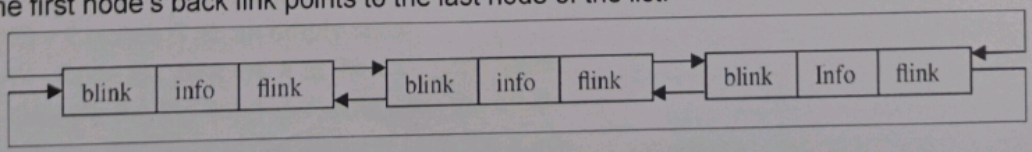
21. **Define circularly linked list?**

In a singly circular linked list the last node's link points to the first node of the list.



22. **Define double circularly linked list?**

In a circular doubly linked list the last node's forward link points to the first ode of the list, and the first node's back link points to the last node of the list.



23. **Mention the disadvantages of circular list?**

The disadvantage of using circular list is

- It is possible to get into an infinite loop.
- It is not possible to detect the end of the list.

24. **What is the need for the header?**

Header of the linked list is the first element in the list and it may store the number of elements in the list. It points to the first data element of the list.

Without header

- Insertion at the front of the list needs special coding & updating of the linked list address.
- Deletion at the front of the list also needs special coding & updating of the linked list address.

25. **Write the routine to push a element into a stack.**

Push an element

STEP 1 START.

STEP 2 Store the element to push into array.

STEP 3 Check if top== (MAXSIZE-1) then stack is full else goto step 4.

STEP 4 Increment top as top = top+1.

STEP 5 Add element to the position stk[top]=num.

STEP 6 STOP.

26. **State the difference between array and linked list.**

An array stores elements in a contiguous memory location. Linked lists store elements randomly at any address in the memory. In an array, memory size is fixed while declaration and cannot be altered at run time. Linked lists utilize dynamic memory, i.e. memory size can be altered at run time.

27. **List three applications that uses linked list?**

Three examples/applications that uses linked list are Polynomial representation, radix sort, & multi lists.

28. **Define Stack**

A Stack is an ordered list in which all insertions (Push operation) and deletion (Pop operation) are made at one end, called the top. The topmost element is pointed by top. The top is initialized to -1 when the stack is created that is when the stack is empty. In a stack S = (a1,…an), a1 is the bottom most element and element ai is on top of element ai-1. Stack is also referred as Last In First Out (LIFO) list.

29. **What are the various Operations performed on the Stack?**

The various operations that are performed on the stack are

CREATE(S) – Creates S as an empty stack.

PUSH(S,X) – Adds the element X to the top of the stack.

POP(S) – Deletes the top most elements from the stack.

TOP(S) – returns the value of top element from the stack.

ISEMTPTY(S) – returns true if Stack is empty else false.

ISFULL(S) - returns true if Stack is full else false.

30. **How do you test for an empty stack?**

The condition for testing an empty stack is top =-1, where top is the pointer pointing to the topmost element of the stack, in the array implementation of stack.In linked list implementation of stack the condition for an empty stack is the header node link field is NULL.

31. **Name two applications of stack?**

Nested and Recursive functions can be implemented using stack.Conversion of Infix to Postfix expression can be implemented using stack.Evaluation of Postfix expression can be implemented using stack.

32. **Define a suffix expression.**

The notation used to write the operator at the end of the operands is called suffix notation.

Suffix notation format : operand operand operator

Example: ab+, where a & b are operands and '+' is addition operator.

33. **What do you meant by fully parenthesized expression? Give eg.**

A pair of parentheses has the same parenthetical level as that of the operator to which it corresponds. Such an expression is called fully parenthesized expression.

Ex: (a+((b*c) + (d * e))

34. **Write the postfix form for the expression -A+B-C+D?**

A-B+C-D+

35. **What are the postfix and prefix forms of the expression?**

A+B*(C-D)/(P-R)

Postfix form: ABCD-*PR-/+

Prefix form: +A/*B-CD-PR

36. **Explain the usage of stack in recursive algorithm implementation?**

In recursive algorithms, stack data structures is used to store the return address when a recursive call is encountered and also to store the values of all the parameters essential to the current state of the function.

37. **Define Queues.**

A Queue is an ordered list in which all insertions take place at one end called the rear, while all deletions take place at the other end called the front. Rear is initialized to -1 and front is initialized to 0. Queue is also referred as First In First Out (FIFO) list.

38. **What are the various operations performed on the Queue?**

The various operations performed on the queue are

CREATE(Q) – Creates Q as an empty Queue.

Enqueue(Q,X) – Adds the element X to the Queue.

Dequeue(Q) – Deletes a element from the Queue.

ISEMTPTY(Q) – returns true if Queue is empty else false.
ISFULL(Q) - returns true if Queue is full else false.

39. **How do you test for an empty Queue?**

The condition for testing an empty queue is rear=front-1. In linked list implementation of queue the condition for an empty queue is the header node link field is NULL.

40. **Write down the function to insert an element into a queue, in which the queue is implemented as an array. (May 10)**

Q – Queue
X – element to added to the queue Q
IsFull(Q) – Checks and true if Queue Q is full
Q->Size - Number of elements in the queue Q
Q->Rear – Points to last element of the queue Q
Q->Array – array used to store queue elements

```
void enqueue (int X, Queue Q) {
if(IsFull(Q))
  Error ("Full queue");
else     {
Q->Size++;
            Q->Rear = Q->Rear+1;
            Q->Array[ Q->Rear ]=X;
        }
}
```

41. **Define Deque.**

Deque stands for Double ended queue. It is a linear list in which insertions and deletion are made from either end of the queue structure.

42. **Define Circular Queue.**

Another representation of a queue, which prevents an excessive use of memory by arranging elements/ nodes $Q_1, Q_2, \ldots Q_n$ in a circular fashion. That is, it is the queue, which wraps around upon reaching the end of the queue

43. **Illustrate the purpose of front to rear?**

The purpose of "front to rear" in a queue data structure is to maintain the order in which elements were added, ensuring that the first element added (front) is the first one to be removed, and the most recently added element (rear) is the last one to be removed. This behavior is useful in various scenarios where you need to process elements in a specific order, such as in scheduling tasks, managing requests, or implementing algorithms like breadth-first search (BFS).

PART-B

1. Derive an ADT to perform insertion and deletion in a singly linked list.(8) (Nov 10)
2. Design an algorithm to reverse the linked list. Trace it with an example?(8)
3. Write an algorithm for inserting and deleting an element from Circular linked list?(8)
4. Write the algorithm for the deletion and reverse operations on doubly linked list. (8) (Nov 09)
5. Write algorithms to perform the following in doubly linked list: (may 10)

(i) To insert an element in the beginning, middle, end of the list. (8)

(ii) To delete an element from anywhere in the list. (8)

An element is a structure variable that contains an integer data field and a string data field.

6. Write an algorithm to perform the following polynomial manipulation using linked list representation

Insertion, Deletion, Merge, Traversal

7. Write an algorithm for Push and Pop operations on Stack using Linked list. (8)
8. Explain the linked list implementation of stack ADT in detail?
9. Define an efficient representation of two stacks in a given area of memory with n words and explain.
10. Explain linear linked implementation of Stack and Queue?
11. write an ADT to implement stack of size N using an array. The elements in the stack are to be integers. The operations to be supported are PUSH, POP and DISPLAY. Take into account the exceptions of stack overflow and stack underflow. (8)

a. A circular queue has a size of 5 and has 3 elements 10,20 and 40 where F=2 and R=4. After inserting 50 and 60, what is the value of F and R. Trying to insert 30 at this stage what happens? Delete 2 elements from the queue and insert 70, 80 & 90. Show the sequence of steps with necessary diagrams with the value of F & R. (8 Marks)

12. Write the algorithm for converting infix expression to postfix (polish) expression?
13. Explain in detail about priority queue ADT in detail?
14. Write a function called 'push' that takes two parameters: an integer variable and a stack into which it would push this element and returns a 1 or a 0 to show success of addition or failure. (6)
15. What is a DeQueue? Explain its operation with example?
16. Explain the array implementation of queue ADT in detail?
17. Explain the addition and deletion operations performed on a circular queue with necessary algorithms.(8) (Nov 09)