**21-6**

**LINKEDLIST CONTAINS LOOP OR CYCLE:**

Floyd's algorithm is used to find out whether linkedlist contains cycle or not.

To find out the meeting point if cycle assign 2 pointers to head nodeone pointer jumps one step other jumps at two steps.

if they are meeting at one node we declare linked list contain cycle that nodeis meeting point

**FINDING START NODE OF CYCLE:**

1.assign 2 pointers.One with head node other one with meeting point node.

2.Now both jump one one step at a time where they are meeting is called starting node of the cycle.

**REMOVING CYCLE FROM LINKEDLIST:**

Find out the node which is connected with staring node of the cycle and make its next is none.

**CIRCULAR LINKEDLIST:**

last node is connected with first node then it becomes circluar linkedlist.Output will be infinite loop.

**DELETING NODE FROM LINKEDLINK**:

1.Deleting the first node.

2.Deleting the middle node.

3.Deleting at last.

**1.DELETING THE FIRST NODE:**

Make the second node as head node.

Make the first node head as NONE.

**2.DELETING LAST NODE:**

Traverse till last but before node.

Make its next as NONE.

**DOUBLE LL:**

1.Let newnode next point to head node

2.Let head node previous point to new node

3.Make your newnode as head

newnode.next=head

head.prev=new

self.head=new

**INSERT AT LAST:**

1.Traverse till lastnode.

2.Create node to be inserted name it as new.

temp.next=new

new.prev=temp

node:Temp is last node.

22-6

**Stack**:(push ,pop)

LIFO

1.USing arrays.

2.Using linkedlists

3.Using inbuilt functions.

NOTE:Insertion and deletion happens at one end instance which is on top.

**2.USING LINKEDLIST:**

Inserting node at last and deleting the last node becomes stack.

METHOD-2:

While implementing stack using linked lists we can opt this method also.Here we do insert node a beginning deleting the head node or first node.

NOTE:

In this above 2 methods second one is efficient.Because in first method we have to traverse trough last node to do pop operation.

**PROGRAM:**

Get combination of parenthesis as input and chech whether balenced or not.

**QUEUE**:(enqueue,dequeue)

This follows FIFO.

1.USing arrays.

2.Using linkedlists

3.Using inbuilt functions.

Insertions and deletions happens at 2 different ends rear and front.

**PROGRAM:**

Create a stack using user input extract only even numbers and print.

**PROGRAM:**

Creat 2 linkedlist both same or not.

**APPLICATION OF STACK:**

One of most imp application of stack is expression conversions.

1.Infix

2.Prefix

3.Postfix

a+b+c

Infix to postfix:

ab+c+

**RULES:**

1.Check priorities.()

\*/

+-

2. No two operaters of same priority stays together in stack.

3.Lowest priority operator cannot be placed before highest priority in the stack.

4.If one operator is inbetween open and closed brace in the stack pop it.

**23-6**

**TYPES OF QUEUES:**

Circular queue

Priority queue

**PRIORITY QUEUE:**

Jobs will be in a queue.However priorities will be assigned.

As per the prioritites jobs will be alloted.

eg:

T1 3

T2 2

T3 4

T4 1

priority t4

sort=ascending order

sort(reverse=True)=Descending order

**CIRCULAR QUEUE:**

In normal queue concept even if there is place after deleting items we would not be able to utilize those spaces in order to use those spaces we are coming with a concept called circular queue.

1.Initizaling circular queue.

Front=Rear=-1

2.Insertion happens at rear and deletion happens at front.

3.Inserting first element.

Front=Rear=0

4.Insertion from second element.

Rear=Rear+1

Then insert.

5.After deleting front element

Front=Front+1

Condition for queue is full:

Self.rear+1

**NON-LINEAR DATASTRUCTURE:**

**TREE:**

To implement hierarchical data or inf we use tree data structure.

In a tree every node can have n number of children.

**Height of node :** Look down and count the levels

**Depth**:Look up and count levels.

If parent having 2 children we call them as left child and right child.

Tree can be divided into 2 types:

1.Left subtree and right subtree.

**NOTE:**

Whenever we perform any operation in tree we have to complete left subtree first.Then we move ahead with right subree.

**Leafnode** : A node which has no children.

**Internal** nodes **:**

**Degree=size-1**

**Breadth:total no of leaf nodes**

**Tree size:all nodes**

**Neighbours : nodes in same** level

**TREE DATASTRUCTURE:**

1.Binary tree

2.Binary search tree

3.AVL tree

4Trie Tree

5.Heap Tree

1**.Binary Tree** :

Any node can have maximum 2 children.

1. **Complete Binary Tree:** all levels should be full except last level .Last level all nodes should be at atmost left arranged.
2. **Strict binary**

All nodes should have 2 or no children.

1. **Perfect Binary Tree:**

All leafnode should be at same level . In which all interior nodes have 2 children and all leaves have the same depth or same level.

**Tree traversal:**

1.Level order traversal

2.Depth order traversal

1. **Level order traversal:**

Traverse level by level left to right.

6 10 15 17

1

**6 10**

15 17

**a. Inorder Left Root Right**

**b. Preorder Root Left Right**

**c. Postorder Left Right Root**

**a.Inorder:**

1

4 2 5 1 3

**2 3**

4 5

**(LDR)**

**b.Preorder:**

DLR

1

1 2 4 5 3

**2 3**

4 5

**C. Postorder:**

LRD

1

**2 3 4 5 2 3 1**

4 5

**BINARY SEARCH TREE:**

1.Every node is lesser than its parents.

2.If it is lying at leftside.

3.Every node is greater than its parent if it is lying on right side.

15

9 18

4 13 17 25

PROGRAM:

Create BST and search a number.

100,70,50,60,9,-3