Q) Doubly linked list. Tree traversal.

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node {
   int data;
   Struct Node *left, *right;
} node;
node * createNode (int data) {
    node * new1 = (node *) malloc (sizeof (node));
    new1 -> data = data;
    new1 -> left = new1 -> right = NULL;
    return new1;
}
node * insertNode (node * root, int data) {
   if (root == NULL) {
        return createNode (data);
        }
  if (data < root -> data) {
       root -> left = insertNode (root -> left, data);
       }
       else
      {
       root -> right = insertNode (root -> right, data);
       }
       return root;
}

Void inorderTravereal (node * root) {
    if (root != NULL) {
      inorderTraversal (root -> left);
      printf ("%d", root -> data);
      inorderTraversel (root -> right);
} }
```

```
Void preorderTraversal (node * root) {
  if (root != NULL) { printf ("%d ", root -> data);
  preorderTraversal (root -> left);
  preorderTraversal (root -> right);
}}

Void postorderTraversal (node * root) {
      if (root != NULL) {
order
postTraversal (root -> left);
Postorder Traversal (root -> right);
 printf ("%d", root -> data);
}
}

int main() {
 node * root = NULL;
 int choice, value;
 Printf ("binary search tree operations \n");
while (1) {
 printf("\n1. insert \n2 Inorder Traversal \n3 Preorder Traversal
      \n4 Post-order Traversal \n5 Exit \n");
printf ("Enter your choice :");
 Scanf ("%d", root -> data);

 Switch (choice) {
 case 1:
      printf ("Enter the value to insert :");
      scanf ("%d", &value);
      root = insertNode(root, value);
      break;
```

```
case 2:
    printf("In order Traversal: ");
    inorderTraversal(root);
    printf("\n");
    break;

case 3:
    printf("Pre-order Traversal: ");
    PreorderTraversal(root);
    printf("\n");
    break;

case 4:
    printf("Post order Traversal: ");
    postorderTraversal(root);
    printf("\n");
    break;

case 5:
    printf("Tree representation:\n");
    displayTree(root, 0);
    printf("\n"); break; case 6: exit(0);
default:
    Printf("Invalid choice. Please Try again\n");

}
}
return 0;
}
```

```
void displayByLevel(node * root){
 if (root == NULL){
  printf("The tree is empty \n");
   return;
 }
 Queue * front = NULL;
 Queue * rear = NULL;
 enque(&front, &rear, root);

while (!isQueueEmpty(front)){
  int node count=0
  Queue *temp = front;

 while (temp != NULL ){
  node count ++;
  temp = temp -> next;
 }
while (node count > 0){
 node * current = dequeue(&front);
 printf("%y.d", current -> data);
 if (current -> left != NULL){
      enqueue(&front, &rear, current -> left);

 if  (current -> right != NULL) {
      enqueue (&front, &rear, current -> right);
  }

  node count --;
 }
 printf("\n");
}
}
```

Output:

Menu:

1. Insert
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Display
6. Exit

Enter your choice: 1

Enter the value to insert: 15

1

8

1

23

1

6

1

11

1

22

1

66

Enter your choice: 2

In order traversal: 6 8 11 15 22 23 66

Enter your choice: 3

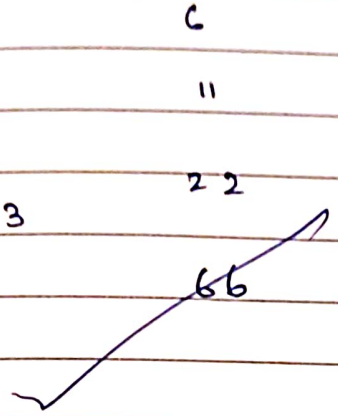
Pre order traversal: 15 8 6 11 23 22 66

Enter your choice: 4

Post order traversal: 6 11 8 22 66 23 15

store
67

Enter you choice: 5
visual tree representation



17/12/24

Q> Doubly Linked List insert to left & delete.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *left;
    struct Node *right;
};
typedef struct Node node;
node *start = NULL;
node *new1, *curr, *ptr;

void create();
void display();
void InsertLeft();
void DeleteSpecificElement();
void main(){
  int ch;
  while(1) {
  printf("\n1. create \n2. Display \n3. Insert Left
        \n4. Delete specific Element \n5. Exit");
  printf("\nEnter your choice : ");
  scanf("%d", &ch);
  switch(ch){
  case 1: create();
      break;
  case 2: display();
      break;
  case 3: InsertLeft();
      break;
  case 4: DeleteSpecificElement();
      break;
```

```
case 5:
exit(0);
}
}
}
Void create(){
char ch;
do{
  new1 = (node *) malloc (sizeof (node));
  printf("\n Enter Value:");
  new1 -> left = NULL;
  new1 -> right = NULL;
if (start == NULL){
   start = new1;
   curr = new1;
   }
  else
  {
  curr -> right = new1;
  new1 -> left = curr;
  curr = new1;
 }
printf("do you want to add an element (Y/N)? ");
scanf("%c", &ch);
}
 while (ch == 'Y' || ch == 'y');
}
```

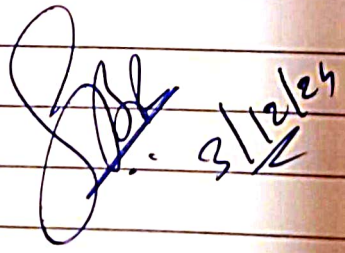
```
Void display(){
  if (start == NULL)
  {
   printf("\n LinkedList is Empty \n");
    return;
  }
  ptr = start;
  printf("\n Elements in Linked List : \n");
  while (ptr != NULL){
   printf("%d", ptr->data);
    ptr = ptr->right;
  }
  printf("\n");
}.
Void InsertLeft(){
   int val;
  printf("\n Enter Value to insert Left");
  scanf("%d", &val);
  new1 = (node*) malloc (sizeof(node));
   new1-> data = val;
   new1-> left = NULL;
   new1-> right = NULL;
   printf("\n Enter the value to Insert Left of :");
   scanf("%d", &val);
    ptr = start;
   while (ptr != NULL && ptr->data != val){
        ptr = ptr->right;
   }
    ptr -> left = new1;
   if (ptr == start){
     start = new1;
    } }
else { printf("\n value not found in the list \n");
}}
```

```
void DeleteSpecificElement(){
 int value;
 printf("\n Enter value to delete : ");
 scanf("%d", &value);
 ptr = start;
while(ptr != NULL && ptr->data != value){
  ptr = ptr -> right;
}
if(ptr == NULL){
  printf("\n value not found in the list \n");
  return;
}
if(ptr -> left != NULL){
   ptr -> left -> right = ptr -> right;
  }
if (ptr -> right != NULL){
   ptr -> right -> left = ptr -> left;
 }
if(ptr == start){
    start = ptr -> right;
}
 free(ptr);
 printf("\n Element with value %d deleted \n", value);
}
```

O/P:

1. Create
2. Display
3. Insert Left.
4. Delete specific element.
5. Exit

Enter your choice: 1
Enter value: 10
Enter value: 20
Enter value: 30.
Enter your choice: 2
10 20 30
Enter your choice: 3
Enter value to insert: 20
Enter to which value to insert = 20
Enter your choice 2
10 20 20 30
Enter your choice 4.
Enter your choice 30.
Enter your choice 2
10 20 20
Enter your choice 5



store
67 17/12/24

Q) write a code for dfs

```
#include <stdio.h>
#define max 10
int graph [max] [max];
int visited [max];
int n;
void dfs (int v) {
    visited [v] = 1;
    for (int i = 0; i < n; i++) {
        if (graph [v] [i] == 1 && ! visited [i])
            dfs (i);
    }
}

int is connected () {
    for (int i = 0; i < n; i++) {
        visited [i] = 0;
    }
    dfs (0);
    for (int i = 0; i < n; i++) {
        if (! visited [i]) {
            return 0;
        }
    }
    return 1;
}

int main () {
    int i, j;
    printf ("Enter the no. of vertices \n");
    scanf ("%d", &n);
```

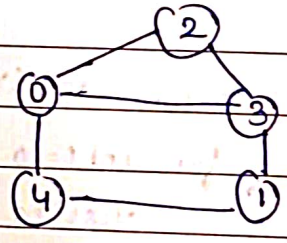
```
printf("Enter adjacency matrix \n");
for(i=0; i<n; i++){
    for(j=0; j<n; j++)
        scanf("%d", &graph[i][j]);
}
}
if(isconnected(0)){
    printf("The graph is connected \n");
}else{
    printf("The graph is not connected \n");
}
return 0;
}
```

Output

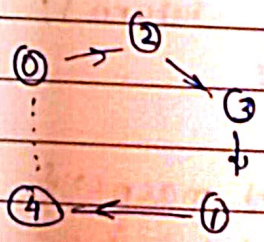
Enter number of vertices : 5

Enter adjacency matrix.

0
0
1
1
1
0
0
0
1
1 0
1 0
0 1
0 1
1 0
0 0
1 0
1
1



	0	1	2	3	4
0	0	0	1	1	1
1	0	0	0	1	1
2	1	0	0	1	0
3	1	1	1	0	0
4	1	1	0	0	0



The graph is connected.

Q) BFS Traversal using c

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int queue[MAX], front=-1, rear=-1;
void enqueue(int item){
    if(rear==MAX-1)
    printf("Queue Overflow\n");
    return;
  }
  if(front==-1) front=0;
  queue[++rear]=item;
}
int dequeue(){
  if(front==-1 || front>rear){
    printf("Que underflow\n");
    return -1;
  } return queue[front++];
}
void bfs(int graph[MAX][MAX], int visited[max], int start,
         int n){
  int i;
  enqueue(start);
  visited[start] = 1;
  printf("BFS Traversal: ");
  while(front<=rear){
    int current = dequeue();
    printf("%d ", current);
    for(i=0; i<n; i++){
      if(graph[current][i]==1 && !visited[i]){
        enqueue(i);
        visited[i]=1;
      }
    }
  }
  printf("\n");
}
```

```
int main()
int n, i, j, start;
int graph[MAX][MAX], visited[MAX] = {0};
printf("Enter no of vertices: ");
scanf("%d", &n);
printf("Enter adjacency matrix:\n");
 for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            scanf("%d", &graph[i][j]);
 printf("Enter starting vertex: ");
 scanf("%d", &start);
 bfs(graph, visited, start, n);
 return 0;
}
```

Enter the number of vertices : 5

Enter the adjacency matrix:

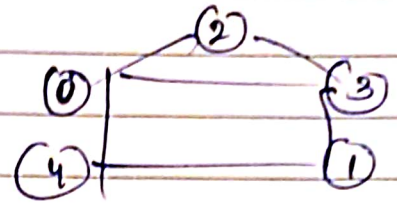
```
0 0 1 1 1
0 0 0 1 1
1 0 0 1 0
1 1 1 0 0
1 1 0 0 0
```

Matrix from

```
0
0
1
1
0
0
0
1
1
1
0
0 0
1 0
0 1
1 1
1 0
1 0
0
```

store
67

Enter starting vertex : 1
BFS Traversal : 1 3 4 0 2 .



17/12/24

	0	1	2	3	4
0	0	0	1	1	1
1	0	0	0	1	1
2	1	0	0	1	0
3	1	1	1	0	0
4	1	1	0	0	0