LAB ASSIGNMENT

CH. HariPriya
AP19110010631
CSE-H

1. Write a C program to print pre order, in order and post order transversal on binary tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  int value;
  struct node *left;
  struct node *right;
};
void inorder(struct node *root)
{
  if(root==NULL)
   return;
 inorder(root->left);
    printf("%d->",root->data);
 inorder(root->right);
};
  void preorder(struct node *root)
  {
   if(root==NULL)
   return;
   printf("%d->",root->data);
   preorder(root->left);
     preorder(root->right);
  };
    void postorder(struct node *root)
```

Scanned with CamScanner

```
{
  if (root == NULL)
  return;
  post order (root -> left);
  post order (root -> Right);
      printf ("%d ->", root -> data);
  }
     struct node * create Node (value)
    { struct node * new Node = malloc( size of
                                (struct node));

      new Node -> data = value;
      new Node -> left = NULL;
      new Node -> Right = NULL;
       return New Node;
      }
      void main( )
      {
         struct node * root = create Node(1);
          root -> Left = create Node (12);
          root -> Right = create Node (9);
          root -> Left -> Left = create Node (10);
          root -> Left -> Right = create Node (15);
          root -> Right -> Left = create Node (11);
          root -> Right -> Right = create Node (16);
        printf ("In order traversal \n");
            in order (root);
        printf ("In pre order traversal \n");
            pre order (root);
```

Scanned with CamScanner

```
printf("\n postorder traversal\n");
    postorder(root);
}
```

Output:-

Inorder traversal

10 → 12 → 15 → 1 → 11 → 9 → 16 →

preorder traversal

1 → 12 → 10 → 15 → 9 → 11 → 16 →

postorder traversal

10 → 15 → 12 → 11 → 16 → 9 → 1 →

2. Write a c program to create (or insert) and inorder traversal on binary search tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int key;
  struct node *left, *right;
};
struct node * newnode(int item)
{
struct node *temp = (struct node*)malloc(sizeof
                                (struct node));
temp->key = item;
temp->left = temp->right = NULL
return temp;
};
  void inorder(struct node * root)
  {  if(root != NULL)
     {
```

Scanned with CamScanner

```
    inorder(root->left);
    printf("%d\n", root->key);
    inorder (root->right);
    }
}
struct node* insert(struct node* node, int key)
{
 if(node == NULL) return new Node(key);
 if (key< Node->key)
 node->left = insert(node->left, key);
 else if (key > node->key)
 node->right = insert(node->right, key);
 return node;
}
 int main()
 {
  struct node* root = NULL
  root = insert(root, 3);
  insert (root, 12);
  insert (root, 51);
  insert (root, 43);
  insert (root, 37);
  insert (root, 98);
  insert (root, 5);

  inorder (root);
  return 0;
 }
```

Output:-

```
37
43
51
98
```

Scanned with CamScanner

3. Write a c program depth first search (DFS) using a array.

```
#include<stdio.h>
void DFS(int);
int G[10][10], visited[10], n;
void main()
{
  int i, j;
  printf("enter number of vertices:");
scanf("%d",&n);
printf("\n enter adjacency matrix of the graph")
for( i=0; i<n; i++)
  for( j=0; j<n; j++)
scanf("%d",& G[i][i]);
  for( i=0; i<n; i++)
      visited[i]=0;
        DFS(0);
}
void DFS(int i)
{
  int j;
  printf("\n %d", i);
      visited[i]=1;
  for( j=0; j<n; j++)
      if( ! visited[j] && G[i][j]==1)
      DFS(j);
}
```

Output:-

Enter number of vertices: 6

Enter adjacency Matrix of the graph: 1 0 1 0 0 1

Scanned with CamScanner

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

0
2
1
4
3
5
```

4. write a c program breath first search (BFS) using array.

```
#include <stdio.h>
#include <stdlib.h>
#define Max 100
#define intial 1
#define waiting 2
#define visited 3

    int n;
    int adj [Max][Max];
    int state [Max];
    void create_graph ();
    void BF_Traversal();
    void BFS (int v);
    int queue[Max], front = -1, rear = -1;
    void insert_queue (int vertex);
    int delete_queue ();
    int is empty_queue ();
    int main ()
    {
```

Scanned with CamScanner

```
create_graph();
BF_Traversal();
return 0;
}
void BF_Traversal()
{
 int v;
 for(v=0; v<n; v++)
  state[v]=initial;
  printf("enter start vertex for BFS:\n");
  scanf("%d",&v);
  BFS(v);
 }
  void BFS(int v);
  {
   int i;
   insert_queue(v);
   state[v]=waiting;
   while(!is_empty_queue())
   {
    v=delete_queue();
    printf("%d",v);
    state[v]=visited;
    for(i=0;i<n;i++)
    {
     if(adj[v][i]==1 && state[i]==initial)
     {
      insert_queue(i);
      state[i]=waiting;
     }
    }
   }
   printf("\n");
  }
```

Scanned with CamScanner

```
void insert_queue (int vertex)
{
  if (rear == Max-1)
printf ("Queue over flow \n");
else
{
 if (front == -1)
   front = 0;
    rear = rear+1;
    queue[rear] = vertex;
    }
    }
     int is empty_queue()
     {
      if (front == -1 // front > rear)
        return 1;
        else
         return 0;
        }
        int delete_item;
        if (front == -1 || front > rear)
        {
        printf ("queue underflow\n")
        exit(1);
       }
       delete_item = queue[front];
       front = front+1;
       return delete_item;
       }
        void create_graph()
        {
        int count, max_edge, origin, destin;
```

Scanned with CamScanner

```
printf ("enter number of vertices");
scanf ("%d", &n);
max_edge = n * (n-1);
for ( count=1; count<=max_edge; count++)
{
  printf("enter edge %d (-1 to quit): "count);
  scanf ("%d %d", &origin, &destin);
  if((origin==-1)&&(destin==-1))
  break;
  if (origin>=n|| destin >= n || origin<0 || destin<0)
  {
  printf (" invalid edge ! \n");
  count--;
  }
  else
  {
  adj [origin][destin]=1;
  }
 }
}
```

Output:-

```
Enter number of vertices=9
enter edge 1 (-1 to quit): 0
1
enter edge 2 (-1 to quit): 0
3
enter edge 3 (-1 to quit): 0
4
enter edge 4 (-1 to quit): 1
2
enter edge 5 (-1 to quit): 3
6
enter edge 6 (-1 to quit): 4
7
enter edge 7 (-1 to quit): 6
4
```

Scanned with CamScanner

```
enter edges(-1 to quit) : 6
7
enter edge 9 (-1 to quit): 2
5
enter edge 10 (-1 to quit): 4
5
enter edge 11 (-1 to quit): 7
5
enter edge 12 (-1 to quit): 7
0
enter edge 13 (-1 to quit): 7
-1
enter start vertex to BFS:
0
0 1 3 4 2 6 5 7
```

5. Write 'c' program for Linear search algorithm.

```
#include<stdio.h>
int main()
{
    int array[100], search, c, n;
    printf("enter number of elements in array\n");
    scanf("%d", &n);
        printf("enter %d integer\n", n);
    for (c=0; c<n; c++)
      scanf("%d", &array[c]);
    printf("enter a number to search\n");
    scanf("%d", &search);
    for (c=0; c<n; c++)
    {
      if (array[c]==search)
      {
        printf("%d present at location %d\n",
                                search, c+1);
```

Scanned with CamScanner

```
    Break;
    }
  }
  if (c==n)
  printf ("%d isn't present in the array \n", search);
  return 0;
  }
```

output:-

```
Enter number of elements in array
5
Enter 5 integers
25
14
86
95
38
Enter a number to search
95
95 present at location 4.
```

6. Write a C program for binary search algorithm.

```
#include <stdio.h>
  int main()
  {
   int c, first, last, middle, n, search, array[100];
   printf ("enter number of elements \n");
   scanf ("%d", &n);
   printf ("enter %d integers \n", n);
   for ( c=0; c<n; c++)
    scanf ("%d", &array[c]);
    printf ("enter value to find \n");
    scanf ("%d", &search);
    first=0;
```

Scanned with CamScanner

```
last = n-1;
middle = (first + last)/2;
while (first <= last)
{
  if (array[middle] < search)
    first = middle+1;
  else if (array[middle] == search)
  {
    printf("%d found at location %d\n", search,
                                middle+1);
    break;
  }
  else
    last = middle-1;
    middle = (first + last)/2;
}
  if (first > last)
    printf("not found! %d is n't present in
        the list\n", search);

  return 0;
}
```

Output:

```
enter number of elements
5
enter 5 integers
25
36
41
51
95
Enter value to find in 51
51 found at location 4.
```

Scanned with CamScanner