LAB PROGRAMS

Write a C program to print preorder, inorder, and postorder traversal on Binary Tree

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

```
}
struct node* insertLeft(struct node* root, int value)
{
   root->left = createNode( value);
   return root->left;
}
struct node* insertRight(struct node* root, int value)
{
root->right = createNode( value);
return root->right;
}
int main()
{
   struct node* root = createNode(1);
   insertLeft(root,8);
   insertRight(root,3);
   insertLeft(root->left, 9);
   insertRight(root->right, 7);

   printf("\npreorder traversal \n");
   preorder(root);
   printf("\nInoreder traversal \n");
   Inorder(root);
   printf("\npostorder traversal \n");
   postorder(root);
}
```

Write a C program to create (or insert) and inorder traversal on Binary Search Tree

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node* left;
struct node* right;
};
struct node* createNode(value){
struct node* newNode = malloc(sizeof(struct node));
newNode->data = value;
```

```
newNode->left = NULL;
newNode->right = NULL;
return newNode;
}
struct node* insert(struct node* root, int data)
{
if (root == NULL) return createNode(data);

if (data < root->data)
root->left = insert(root->left, data);
else if (data > root->data)
root->right = insert(root->right, data);
return root;
}
void inorder(struct node* root){
if(root == NULL) return;
inorder(root->left);
printf("%d ->", root->data);
inorder(root->right);
}
int main(){
struct node *root = NULL;
root = insert(root, 20);
insert(root, 10);
insert(root, 30);
insert(root, 40);
insert(root, 50);

insert(root, 60);
insert(root, 70);
insert(root, 80);
inorder(root);
}
```

Write a C program breath first search (BFS) using array.

```
#include<stdio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v) {
        for(i = 1; i <= n; i++)
                if(a[v][i] && !visited[i])
```

```
                q[++r] = i;
        if(f <= r) {
                visited[q[f]] = 1;
                bfs(q[f++]);
        }
}

void main() {
        int v;
        printf("\n Enter the number of vertices:");
        scanf("%d", &n);

        for(i=1; i <= n; i++) {
                q[i] = 0;
                visited[i] = 0;
        }

        printf("\n Enter graph data in matrix form:\n");
        for(i=1; i<=n; i++) {
                for(j=1;j<=n;j++) {
                        scanf("%d", &a[i][j]);
                }
        }

        printf("\n Enter the starting vertex:");
        scanf("%d", &v);
        bfs(v);
        printf("\n The node which are reachable are:\n");

        for(i=1; i <= n; i++) {
                if(visited[i])
                        printf("%d\t", i);
                else {
                        printf("\n Bfs is not possible. Not all nodes are reachable");
                        break;
                }
        }
}
```

Write a C program Depth first search (BFS) using array

```
#include<stdio.h>
```

```
int A[10][10],v[10],n;

void DFS(int i)
{
    int j;
printf("\n%d",i);
    v[i]=1;
for(j=0;j<n;j++)
       if(!v[j]&&A[i][j]==1)
            DFS(j);
}
void main()
{
    int i,j;
    printf("Enter number of vertices:");
    scanf("%d",&n);
printf("\nEnter adjacency matrix of the graph:");

for(i=0;i<n;i++)
       for(j=0;j<n;j++)
scanf("%d",&A[i][j]);

   for(i=0;i<n;i++)
        v[i]=0;
    DFS(0);
}
```

Write a C program for linear search algorithm

```
#include <stdio.h>
void main()
{
  int array[100], search, i, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d integer(s)\n", n);
  for (i = 0; i < n; i++)
  scanf("%d", &array[i]);
 printf("Enter a number to search\n");
```

```
 scanf("%d", &search);
  for (i = 0; i < n; i++)
  {
    if (array[i] == search)       {
      printf("%d is present at location %d.\n", search, i+1);
      break;
    }
  }
  if (i == n)
    printf("%d isn't present in the array.\n", search);
}
```

Write a C program for binary search algorithm

```
#include <stdio.h>
void 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;
  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 isn't present in the list.\n", search);
```

```
}
```