LAB CODES

J MANOJ MAHANANDI
AP19110010390
CSE-F

Write a C program to print preorder, in order and postorder traversal on Binary Tree.

CODE :

```
#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;
}
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, 40);
```

```
insert(root, 90);
insert(root, 20);
insert(root, 55);
insert(root, 66);
insert(root, 76);
insert(root, 150);
insert(root, 170);
inorder(root);
}
```

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

CODE :

```
#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 depth-first search (DFS) using an array.

CODE :

```
#include <stdio.h>
#include <stdlib.h>
int source,V,E,time,visited[100],G[100][100];
void DFS(int i)
{
int j;
visited[i]=1;
printf(" %d->",i+1);
for(j=0;j<V;j++)
{
if(G[i][j]==1&&visited[j]==0)
DFS(j);
}
}

int main()
{
int i,j,v1,v2;
printf("\t\t\tGraphs\n");
printf("Enter the no of edges:");
scanf("%d",&E);
printf("Enter the no of vertices:");
scanf("%d",&V);
for(i=0;i<V;i++)
{
for(j=0;j<V;j++)
G[i][j]=0;
}
/* creating edges :P */
for(i=0;i<E;i++)
{
printf("Enter the edges (format: V1 V2) : ");
scanf("%d%d",&v1,&v2);
G[v1-1][v2-1]=1;

}
for(i=0;i<V;i++)
{
for(j=0;j<V;j++)
printf(" %d ",G[i][j]);
printf("\n");
}
printf("Enter the source: ");
scanf("%d",&source);
DFS(source-1);
return 0;
}
```

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

CODE :

```
#include<stdio.h>
int G[100][100],q[100],visited[100],n,front = 1, rear = 0 ;
void bfs(int v)
{
int i;
visited[v] = 1;

for(i=1;i<=n;i++)
if(G[v][i] && !visited[i])
q[++rear]=i;
if(front <= rear)
bfs(q[front++]);
}
int main(){
int v,i,j;
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",&G[i][j]);
printf("\n Enter the starting vertex:");
scanf("%d",&v);
bfs(v);
printf("\n The nodes which are reachable are:\n");
for(i=1;i<=n;i++)
if(visited[i])
printf("%d\t",i);
else
printf("\n %d is not reachable",i);
return 0;
}
```

Write a C program for the linear search algorithm

CODE :

```
#include<stdio.h>

int main(){

int a[100],i,x,n;
printf("How many elements?");
scanf("%d",&n);
printf("Enter array elements:\n");
for(i=0;i<n;++i)
scanf("%d",&a[i]);
printf("\nEnter element to search:");
scanf("%d",&x);
for(i=0;i<n;++i)
if(a[i]==x)
break;
if(i<n)
printf("Element found at index %d",i);
else
printf("Element not found");
return 0;
}
```

Write a C program for binary search algorithm.

CODE :

```
#include<stdio.h>
int main()
{
int arr[100],i,n,x,flag=0,first,last,mid;
printf("Enter size of array:");
scanf("%d",&n);
printf("\nEnter array element(ascending order)\n");
for(i=0;i<n;++i)
scanf("%d",&arr[i]);
printf("\nEnter the element to search:");
scanf("%d",&x);
first=0;
last=n-1;
while(first<=last)
{
mid=(first+last)/2;

if(x==arr[mid]){
flag=1;
break;
}
else
if(x>arr[mid])
first=mid+1;
else

last=mid-1;
}
if(flag==1)
printf("\nElement found at position %d",mid+1);
else
printf("\nElement not found");
return 0;
}
```