**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*print preorde,post order,in order\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

struct node\* newNode(int data)

{

struct node\* node = (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

printPostorder(node->left);

printPostorder(node->right);

printf("%d ", node->data);

}

void printInorder(struct node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

printf("%d ", node->data);

printInorder(node->right);

}

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

printf("%d ", node->data);

printPreorder(node->left);

printPreorder(node->right);

}

int main()

{

struct node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

printf("\nPreorder of binary tree is \n");

printPreorder(root);

printf("\nInorder of binary tree is \n");

printInorder(root);

printf("\nPostorder of binary tree is \n");

printPostorder(root);

getchar();

return 0;

}

Output1

Preorder of binary tree is

1 2 4 5 3

Inorder of binary tree is

4 2 5 1 3

Postorder of binary tree is

4 5 2 3 1

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*inorder transversal\*\*\*\*\*\*\*\*\*\*\*\***

#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) {

if(root == NULL) return;

postorder(root->left);

postorder(root->right);

printf("%d ->", root->data);

}

struct node \*createNode(value)

{

struct node\* newNode = malloc(sizeof(struct node));

newNode->data = value;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

void main()

{

struct node\* root = createNode(1);

root->left=createNode(480);

root->right=createNode(140);

root->left->left=createNode(650);

root->left->right=createNode(288);

root->right->left=createNode(365);

root->right->right=createNode(420);

printf("Inorder traversal \n");

inorder(root);

printf("\nPreorder traversal \n");

preorder(root);

printf("\nPostorder traversal \n");

postorder(root);

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*linear search\*\*\*\*\*\*\*\*\*\*\*\***

**#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(s)\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 is present at location %d.\n", search, c+1);**

**break;**

**}**

**}**

**if (c == n)**

**printf("%d isn't present in the array.\n", search);**

**return 0;**

**}**

**Output3**

**Ennter number of elements in array**

**5**

**Enter 5 integer(s)**

**25**

**14**

**36**

**95**

**38**

**Enter a number to search**

**95**

**95 is present at location 4.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*binary search\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**#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)**

**{**

**inorder(root->left);**

**printf("%d \n", root->key);**

**inorder(root->right);**

**}**

**}**

**struct node\* insert(struct node\* node, int key)**

**{**

**if (node == NULL) return newNode(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, 120);**

**insert(root, 590);**

**insert(root, 436);**

**insert(root, 360);**

**insert(root, 999);**

**insert(root, 500);**

**inorder(root);**

**return 0;**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*DFS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

#include<stdio.h>

int G[10][10],visited[10],n;

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);

}

}

}

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",&G[i][j]);

for(i=0;i<n;i++)

{

visited[i]=0;

DFS(0);

}

}

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BFS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

#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");

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*