**DAA LAB 1**

**NAME: TUSHAR MAROO**

**REGISTRATION NUMBER: 190905238**

**SECTION: A**

**ROLL NUMBER: 40**

**Q1.** Write a program to construct a binary tree to support the following operations.

Assume no duplicate elements while constructing the tree.

i. Given a key, perform a search in the binary search tree. If the key is found

then display “key found” else insert the key in the binary search tree.

ii. Display the tree using inorder, preorder and post order traversal methods.

**CODE**

#include <stdio.h>

#include <stdlib.h>

typedef struct node Node;

struct node

{

Node \*left;

Node \*right;

int data;

};

void inorder(Node \*root)

{

if (root == NULL)

return;

inorder(root->left);

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

inorder(root->right);

}

void preorder(Node \*root)

{

if (root == NULL)

return;

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

preorder(root->left);

preorder(root->right);

}

void postorder(Node \*root)

{

if (root == NULL)

return;

postorder(root->left);

postorder(root->right);

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

}

Node \*search(Node \*root, int x)

{

if (root == NULL)

{

Node \*temp = (Node \*)malloc(sizeof(Node));

temp->data = x;

temp->left = temp->right = NULL;

printf("KEY INSERTED.\n");

return temp;

}

if (x == root->data)

{

printf("KEY FOUND.\n");

}

else if (x > root->data)

{

root->right=search(root->right, x);

}

else

{

root->left=search(root->left, x);

}

return root;

}

int main()

{

Node \*binTree = NULL;

int x, key;

printf("COMMANDS : \n");

printf("1 : ENTER KEY.\n");

printf("2 : INORDER TRAVERSAL.\n");

printf("3 : PREORDER TRAVERSAL.\n");

printf("4 : POSTORDER TRAVERSAL.\n");

printf("5 : EXIT.\n");

printf("ENTER COMMAND : ");

scanf("%d", &x);

while (x != 5)

{

switch (x)

{

case 1:

printf("ENTER KEY : ");

scanf("%d", &key);

binTree = search(binTree, key);

break;

case 2:

printf("INORDER : ");

inorder(binTree);

printf("\n");

break;

case 3:

printf("PREORDER : ");

preorder(binTree);

printf("\n");

break;

case 4:

printf("POSTORDER : ");

postorder(binTree);

printf("\n");

break;

default:

printf("INVALID COMMAND.\n");

break;

}

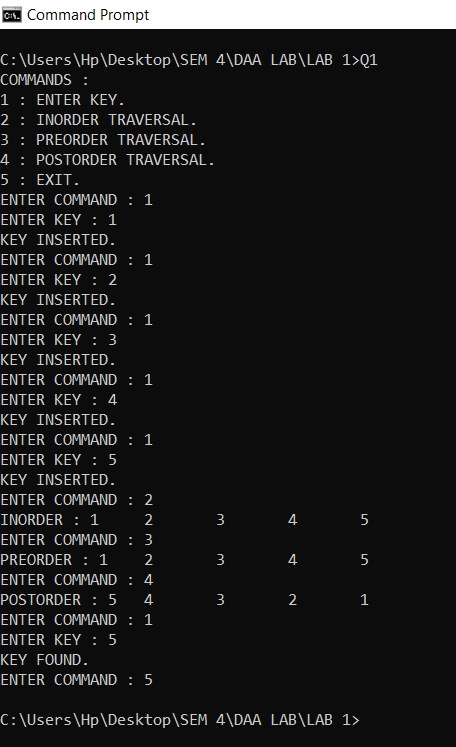
printf("ENTER COMMAND : ");

scanf("%d", &x);

}

}

**OUTPUT**



**Q2.** Write a program to implement the following graph representations and display

them.

i. Adjacency list

ii. Adjacency matrix

**CODE**

#include <stdio.h>

#include <stdlib.h>

typedef struct node Node;

struct node

{

Node \*next;

int data;

};

Node \*createNode(int x)

{

Node \*temp = (Node \*)malloc(sizeof(Node));

temp->data = x;

temp->next = NULL;

return temp;

}

Node \*insert(Node \*head, int x)

{

if (head == NULL)

{

head = createNode(x);

return head;

}

Node \*temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = createNode(x);

return head;

}

void print(Node \*head)

{

Node \*temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

printf("NULL\n");

}

int main()

{

int v, e, i, j;

printf("Enter the number of vertices : \n");

scanf("%d", &v);

int \*\*adjMat = malloc(v \* sizeof(int \*));

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

adjMat[i] = malloc(v \* sizeof(int));

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

for (j = 0; j < v; j++)

adjMat[i][j] = 0;

Node \*\*adjList = malloc(v \* sizeof(Node \*));

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

adjList[i] = NULL;

printf("Enter the number of edges : \n");

scanf("%d", &e);

int start, end;

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

{

printf("Edge %d : \n", i + 1);

printf("Enter start vertex : ");

scanf("%d", &start);

printf("Enter end vertex : ");

scanf("%d", &end);

adjMat[start][end] = 1;

adjList[start] = insert(adjList[start], end);

}

printf("Adjacency Matrix : \n");

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

{

for (j = 0; j < v; j++)

printf("%d\t", adjMat[i][j]);

printf("\n");

}

printf("Adjacency List : \n");

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

{

printf("%d : ", i);

print(adjList[i]);

}

}

**OUTPUT**

