/\* Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers

a. Create a BST of N Integers

b. Traverse the BST in Inorder, Preorder and Post Order

c. Search the BST for a given element (KEY) and report the appropriate message \*/

#include <stdio.h>

#include <stdlib.h>

int flag=0;

struct BST

{

int data;

struct BST \*left,\*right;

};

/\*FUNCTION PROTOTYPE\*/

void insert(struct BST \*, struct BST \*);

void inorder(struct BST \*);

void preorder(struct BST \*);

void postorder(struct BST \*);

struct BST \*search(struct BST \*, int, struct BST \*\*);

int main()

{

int choice;

int ans =1;

int key;

struct BST \*newnode, \*root, \*tmp, \*parent;

struct BST \*get\_node();

root = NULL;

printf("\nProgram For Binary Search Tree ");

do

{

printf("\n 1.Create");

printf("\n 2.Search");

printf("\n 3.Recursive Traversals");

printf("\n 4.Exit");

printf("\n Enter your choice :");

scanf("%d", &choice);

switch (choice)

{

case 1:

do

{

newnode = get\_node();

printf("\n Enter The Element ");

scanf("%d", &newnode->data);

if (root == NULL) /\* Tree is not Created \*/

root = newnode;

else

insert(root, newnode);

printf("\n Want To enter More Elements?(1/0)");

scanf("%d",&ans);

} while (ans);

break;

case 2:

printf("\n Enter Element to be searched :");

scanf("%d", &key);

tmp = search(root, key, &parent);

if(flag==1)

{

printf("\n Parent of node %d is %d", tmp->data, parent->data);

}

else

{

printf("\n The %d Element is not Present",key);

}

flag=0;

break;

case 3:

if (root == NULL)

printf("Tree Is Not Created");

else

{

printf("\nThe Inor\der display :\n");

inorder(root);

printf("\nThe Preorder display : \n");

preorder(root);

printf("\nThe Postorder display : \n");

postorder(root);

}

break;

}

}

while (choice != 4);

}

/\*Get new Node \*/

struct BST \*get\_node()

{

struct BST \*temp;

temp = (struct BST \*) malloc(sizeof(struct BST));

temp->left = NULL;

temp->right = NULL;

return temp;

}

/\*This function is for creating a binary search tree \*/

void insert(struct BST \*root, struct BST \*newnode)

{

if (newnode->data < root->data)

{

if(root->left==NULL)

root->left=newnode;

else

insert(root->left, newnode);

}

if (newnode->data > root->data)

{

if (root->right == NULL)

root->right = newnode;

else

insert(root->right, newnode);

}

}

/\*This function is for searching the node from binary Search Tree\*/

struct BST \*search(struct BST \*root, int key, struct BST \*\*parent)

{

struct BST \* temp;

temp = root;

while (temp != NULL)

{

if (temp->data == key)

{

printf("\nThe %d Element is Present", temp->data);

flag=1;

return temp;

}

\*parent = temp;

if (temp->data > key)

temp = temp->left;

else

temp = temp->right;

}

return NULL;

}

/\*This function displays the tree in inorder fashion \*/

void inorder(struct BST \*temp)

{

if (temp != NULL)

{

inorder(temp->left);

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

inorder(temp->right);

}

}

/\*This function displays the tree in preorder fashion \*/

void preorder(struct BST \*temp)

{

if (temp != NULL)

{

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

preorder(temp->left);

preorder(temp->right);

}

}

/\*This function displays the tree in postorder fashion \*/

void postorder(struct BST \*temp)

{

if (temp != NULL)

{

postorder(temp->left);

postorder(temp->right);

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

}

}