Implementation of Binary Search Tree

#include<stdio.h>

#include<conio.h>

struct node

{

int data;

struct node \*leftChild;

struct node \*rightChild;

}\*root=NULL;

struct node\* insert(struct node \*root, int data)

{

if(root == NULL)

{

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

newNode->data = data;

newNode->leftChild = NULL;

newNode->rightChild = NULL;

return newNode;

}

else if(data< root->data)

{

root->leftChild = insert(root->leftChild,data);

}

else

{

root->rightChild = insert(root->rightChild,data);

}

//original root

return root;

}

void printPreorder(struct node\* node)

{

if(node== NULL)

return;

else

{

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

// first recur on left subtree

printPreorder(node->leftChild);

// then recur on right subtree

printPreorder(node->rightChild);

}

}

void printInorder(struct node\* node)

{

if(node== NULL)

return;

else

{

// first recur on left subtree

printInorder(node->leftChild);

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

// then recur on right subtree

printInorder(node->rightChild);

}

}

void printPostorder(struct node\* node)

{

if(node== NULL)

return;

else

{

// first recur on left subtree

printPostorder(node->leftChild);

// then recur on right subtree

printPostorder(node->rightChild);

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

}

}

int main()

{

int c=0,value;

while(c!=7)

{

printf("\nEnter 1 for Insertion");

printf("\nEnter 2 for Deletion");

printf("\nEnter 3 for Searching");

printf("\nEnter 4 for Postorder");

printf("\nEnter 5 for Preorder");

printf("\nEnter 6 for Inorder");

printf("\nEnter 7 for Exit");

printf("\nEnter your choice: ");

scanf("%d",&c);

switch(c)

{

case 1:

{

printf("\nEnter value: ");

scanf("%d",&value);

root = insert(root,value);

break;

}

case 2:

{

printf("\nEnter value to delete: ");

scanf("%d",&value);

// delete(value);

break;

}

case 3:

{

printf("\nEnter value to search: ");

scanf("%d",&value);

//search(value);

break;

}

case 4:

{

printPostorder(root);

break;

}

case 5:

{

printPreorder(root);

break;

}

case 6:

{

printInorder(root);

break;

}

case 7:

{

break;

}

default:

{

printf("\nInvalid Choice");

break;

}

}

}

return 0;

}