Binary Search Tree

```
#include<stdio.h>
#include<stdlib.h>

typedef struct _BSTNode
{
int data;
struct _BSTNode *left,*right;
}BSTNode;

BSTNode *root=NULL;

typedef struct _Stack
{
BSTNode *data;
struct _Stack *next;
}Stack;

Stack *top=NULL;

Stack * createStackNode()
{
Stack *stack;
stack=(Stack *)malloc(sizeof(Stack));
return stack;
}

int isEmpty()
{
if(top==NULL) return 1;
return 0;
}

void push(BSTNode *node)
```

```
{
if(node==NULL) return;
Stack *t=createStackNode();
if(t==NULL) return;
if(top==NULL)
{
t->data=node;
t->next=NULL;
top=t;
}
else
{
t->data=node;
t->next=top;
top=t;
}
}

BSTNode * pop()
{
BSTNode *t=NULL;
Stack *s;
if(top==NULL) return t;
t=top->data;
s=top;
top=top->next;
free(s);
return t;
}

BSTNode * createNode(int data)
{
BSTNode *t;
t=(BSTNode *)malloc(sizeof(BSTNode));
t->data=data;
```

```
t->left=NULL;
t->right=NULL;
return t;
}

void insert(int data)
{
BSTNode *t,*j;
t=createNode(data);
if(root==NULL)
{
root=t;
return;
}
j=root;
while(1)
{
if(t->data<j->data)
{
if(j->left==NULL)
{
j->left=t;
break;
}
else
{
j=j->left;
}
}
else
{
if(j->right==NULL)
{
j->right=t;
break;
}
else
{
j=j->right;
}
}
}
}

void inOrderRecursive(BSTNode *t)
{
if(t==NULL) return;
inOrderRecursive(t->left);
printf("%d\n",t->data);
inOrderRecursive(t->right);
}

void preOrderRecursive(BSTNode *t)
{
if(t==NULL) return;
printf("%d\n",t->data);
preOrderRecursive(t->left);
preOrderRecursive(t->right);
}

void postOrderRecursive(BSTNode *t)
{
if(t==NULL) return;
postOrderRecursive(t->left);
postOrderRecursive(t->right);
printf("%d\n",t->data);
}

void inOrderIterative(BSTNode *t)
{
```

```
if(t==NULL) return;
while(t!=NULL)
{
push(t);
t=t->left;
}
BSTNode *j,*p;
while(!isEmpty())
{
p=pop();
j=p;
j=j->right;
while(j!=NULL)
{
push(j);
j=j->left;
}
printf("%d\n",p->data);
}
}

void preOrderIterative(BSTNode *j)
{
if(j==NULL) return;
push(j);
BSTNode *t;
while(!isEmpty())
{
t=pop();
printf("%d\n",t->data);
if(t->right!=NULL)
{
push(t->right);
}
if(t->left!=NULL)
{
push(t->left);
}
}
}

void postOrderIterative(BSTNode * t)
{
while(1)
{
while(t!=NULL)
{
if(t->right!=NULL) push(t->right);
push(t);
t=t->left;
}
t=pop();
if(t->right!=NULL && top!=NULL && t->right==top->data)
{
pop();
push(t);
t=t->right;
}else
{
printf("%d\n",t->data);
t=NULL;
}
if(isEmpty()) break;
}
}

void printLeftNodes(BSTNode *t,int b)
{
if(t==NULL) return;
printLeftNodes(t->left,1);
```

```
if(b) printf("%d\n",t->data);
printLeftNodes(t->right,0);
}

void removeFromBST(int num)
{
BSTNode *j,*t,**p2p,*k,*p;
t=root;

while(t!=NULL)
{
if(num==t->data) break;
j=t;
if(num<t->data) t=t->left;
else t=t->right;
}

if(t==NULL)
{
printf("Invalid number\n");
return;
}

if(t==root) p2p=&root;
else if(t==j->left) p2p=&(j->left);
else p2p=&(j->right);

if(t->left==NULL && t->right==NULL)
{
*p2p=NULL;
free(t);
return;
}

if(t->right!=NULL) {
```

```
for(k=t->right;k->left!=NULL;k=k->left) p=k;
k->left=t->left;
if(k!=t->right) {
p->left=k->right;
k->right=t->right;
}
}
else
{
for(k=t->left;k->right!=NULL;k=k->right) p=k;
k->right=t->right;
if(k!=t->left)
{
p->right=k->left;
k->left=t->left;
}
}
*p2p=k;
free(t);
}

int getCountOfLeafNode(BSTNode *t)
{
if(t==NULL) return 0;
if(t->left==t->right) return 1;
return getCountOfLeafNode(t->left)+getCountOfLeafNode(t->right);;
}

int getHeightOfBST(BSTNode *t)
{
if(t==NULL) return 0;
int count1;
int count2;
count1=getHeightOfBST(t->left);
count2=getHeightOfBST(t->right);
```

```
if(count1>count2)return count1+1;
else return count2+1;
}

int main()
{
int num,ch;
while(1)
{
printf("------------------------------------------\n");
printf("Binary Search Tree Operations -\n");
printf("1. Insert Data\n");
printf("2. InOrder Traversal Recursive\n");
printf("3. PreOrder Traversal Recursive\n");
printf("4. Postorder Traversal Recursive\n");
printf("5. InOrder Traversal Iterative\n");
printf("6. PreOrder Traversal Iterative\n");
printf("7. Postorder Traversal Iterative\n");
printf("8. Print left nodes\n");
printf("9. Remove from BST\n");
printf("10. Get count of leaf nodes\n");
printf("11. Get Height of BST\n");
printf("12. Exit\n");
printf("Enter your choice : ");
scanf("%d",&ch);
fflush(stdin);
if(ch==1)
{
printf("Enter data to insert : ");
scanf("%d",&num);
fflush(stdin);
insert(num);
}
if(ch==2)
{
```

```
inOrderRecursive(root);
}
if(ch==3)
{
preOrderRecursive(root);
}
if(ch==4)
{
postOrderRecursive(root);
}
if(ch==5)
{
inOrderIterative(root);
}
if(ch==6)
{
preOrderIterative(root);
}
if(ch==7)
{
postOrderIterative(root);
}
if(ch==8)
{
printLeftNodes(root,0); // 0 for false
}
if(ch==9)
{
printf("Enter data to remove from BST: ");
scanf("%d",&num);
fflush(stdin);
removeFromBST(num);
}
if(ch==10)
{
```

```
printf("Count of leaf nodes are: %d\n",getCountOfLeafNode(root));
}
if(ch==11)
{
printf("%d\n",getHeightOfBST(root));
}
if(ch==12) break;
}
printf("Bye!\n");
printf("----------------------------------------------\n");
return 0;
}
```