**Program for BST insertion, deletion and traversal**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*lchild,\*rchild;

}\*ptr,\*ptr1,\*root,\*new1,\*parent,\*ptr2,\*ptr3,\*ptr4;

void insertion();

void preorder (struct node \*);

void inorder (struct node \*);

void postorder (struct node \*);

void deletion(int);

struct node \* succ(struct node \*);

void main()

{

int ch,c,item;

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

root->data=NULL;

root->lchild=NULL;

root->rchild=NULL;

ptr=root;

while(1)

{

printf("\n enter your choice of operation");

printf("\n 1. insertion \t 2. Deletion 3.Traverse");

scanf("%d",&ch);

switch(ch)

{

case 1:

insertion();

break;

case 2:

printf("\n Enter node to be delete");

scanf("%d",&item);

deletion(item);

break;

case 3:

printf("\n Enter your traversal");

printf("\n 1. Preorder 2. Inorder 3. Postorder");

scanf("%d",&c);

if(c==1)

{

printf("\n Nodes in BST are");

preorder(root);

}

else if(c==2)

{

printf("\n Nodes in BST are");

inorder(root);

}

else if(c==3)

{

printf("\n Nodes in BST are");

postorder(root);

}

break;

default : exit(0);

}

}

getch();

}

void insertion()

{

int item,flag=0;

ptr=root;

printf("\n Enter data part of node to be insert");

scanf("%d",&item);

while(ptr!=NULL && flag==0)

{

if(item==ptr->data)

{

printf("\n Item already exist in BST");

flag=1;

}

else if(item<ptr->data)

{

ptr1=ptr;

ptr=ptr->lchild;

}

else if(item>ptr->data)

{

ptr1=ptr;

ptr=ptr->rchild;

}

}

if(ptr==NULL)

{

new1=(struct node\*)malloc(sizeof(struct node));

new1->data=item;

new1->lchild=NULL;

new1->rchild=NULL;

if(root->data==NULL)

{

root=new1;

printf("\n Node %d is inserted successfully as ROOT Node",item);

}

else if(item<ptr1->data) /\* inserting new node as left child to its parent\*/

{

ptr1->lchild=new1;

printf("\n Node %d is inserted successfully LEFT child",item);

}

else /\* inserting new node as right child to its parent\*/

{

ptr1->rchild=new1;

printf("\n Node %d is inserted successfully Right Child",item);

}

}

}

void preorder(struct node \*p)

{

if(p!=NULL)

{

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

preorder(p->lchild);

preorder(p->rchild);

}

}

void inorder(struct node \*p)

{

if(p!=NULL)

{

inorder(p->lchild);

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

inorder(p->rchild);

}

}

void postorder(struct node \*p)

{

if(p!=NULL)

{

postorder(p->lchild);

postorder(p->rchild);

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

}

}

void deletion(int item)

{

int flag=0,ch,item1;

ptr=root;

while((ptr!=NULL) && (flag==0))

{

if(item==ptr->data)

{

flag=1;

break;

}

else if(item < ptr->data)

{

parent=ptr;

ptr=ptr->lchild;

}

else if(item > ptr->data)

{

parent = ptr;

ptr=ptr->rchild;

}

}

if(flag==0)

{

printf("\n The given item %d is not exist in BST, So Deletion is not possible",item);

}

if((ptr->lchild==NULL) && (ptr->rchild==NULL))

{

ch=1;

}

else if((ptr->lchild!=NULL) && (ptr->rchild!=NULL))

{

ch=3;

}

else

{

ch=2;

}

/\* DELETION FROM CASE 1 \*/

if(ch==1)

{

if(parent->lchild==ptr)

{

parent->lchild=NULL;

printf("\n Node is deleted successfully");

}

else

{

parent->rchild=NULL;

printf("\n Node is deleted successfully");

}

free(ptr);

}

/\* DELETION FROM CASE 2 \*/

if(ch==2)

{

if(parent->lchild==ptr) /\* Node to be deleted is left child of parent node \*/

{

if(ptr->lchild==NULL)

{

parent->lchild=ptr->rchild;

printf("\n Node is deleted successfully");

}

else

{

parent->lchild=ptr->lchild;

printf("\n Node is deleted successfully");

}

}

else /\* Node to be deleted is right child of parent node \*/

{

if(ptr->rchild==NULL)

{

parent->rchild=ptr->lchild;

printf("\n Node is deleted successfully");

}

else

{

parent->rchild=ptr->rchild;

printf("\n Node is deleted successfully");

}

}

free(ptr);

}

/\* DELETION FROM CASE 3 \*/

if(ch==3)

{

ptr4=ptr; /\* to store the address of the node to be delete \*/

ptr1=succ(ptr);

item1=ptr1->data;

deletion(item1);

ptr4->data=item1;

}

}

struct node \* succ(struct node \*ptr2)

{

ptr3=ptr2->rchild;

if(ptr3!=NULL)

{

while(ptr3->lchild!=NULL)

{

ptr3=ptr3->lchild;

}

}

return(ptr3);

}

**OUTPUT:**

enter your choice of operation

1. insertion 2. Deletion 3.Traverse1

Enter data part of node to be insert10

Node 10 is inserted successfully as ROOT Node

enter your choice of operation

1. insertion 2. Deletion 3.Traverse1

Enter data part of node to be insert5

Node 5 is inserted successfully LEFT child

enter your choice of operation

1. insertion 2. Deletion 3.Traverse1

Enter data part of node to be insert15

Node 15 is inserted successfully Right Child

enter your choice of operation

1. insertion 2. Deletion 3.Traverse3

Enter your traversal

1. Preorder 2. Inorder 3. Postorder2

Nodes in BST are 5 10 15

enter your choice of operation

1. insertion 2. Deletion 3.Traverse3

Enter your traversal

1. Preorder 2. Inorder 3. Postorder1

Nodes in BST are 10 5 15

enter your choice of operation

1. insertion 2. Deletion 3.Traverse3

Enter your traversal

1. Preorder 2. Inorder 3. Postorder3

Nodes in BST are 5 15 10

enter your choice of operation

1. insertion 2. Deletion 3.Traverse1

Enter data part of node to be insert2

Node 2 is inserted successfully LEFT child

enter your choice of operation

1. insertion 2. Deletion 3.Traverse1

Enter data part of node to be insert12

Node 12 is inserted successfully LEFT child

enter your choice of operation

1. insertion 2. Deletion 3.Traverse1

Enter data part of node to be insert18

Node 18 is inserted successfully Right Child

enter your choice of operation

1. insertion 2. Deletion 3.Traverse2

Enter node to be delete5

Node is deleted successfully

enter your choice of operation

1. insertion 2. Deletion 3.Traverse3

Enter your traversal

1. Preorder 2. Inorder 3. Postorder2

Nodes in BST are 2 10 12 15 18

enter your choice of operation

1. insertion 2. Deletion 3.Traverse2

Enter node to be delete15

Node is deleted successfully

enter your choice of operation

1. insertion 2. Deletion 3.Traverse3

Enter your traversal

1. Preorder 2. Inorder 3. Postorder2

Nodes in BST are 2 10 12 18

enter your choice of operation

1. insertion 2. Deletion 3.Traverse