**PROGRAM**

```
#include <stdio.h>
#include <stdlib.h>
struct node{
 int data;
 struct node *right;
 struct node *left;};
struct node *insertBST(struct node *, int);
void Inorder(struct node *);
void Preorder(struct node *);
void Postorder(struct node *);
struct node*minvalue(struct node*);
struct node*deletenode(struct node*, int
key);
struct node *temp = NULL;
int main()
{
 struct node *root = NULL;
 int choice, item, n, i=0, key;
 do
 {
 printf("\n\nBinary Search Tree
Operations\n");
 printf("\n1. Insertion");
printf("\n2. Traverse in Inorder");
 printf("\n3. Traverse in Preorder");
 printf("\n4. Traverse in Postorder");
 printf("\n5. Exit");
 printf("\n6. deletion of node");
 printf("\nEnter Choice : ");
 scanf("%d",&choice);
 switch(choice)
 {
 case 1:
 printf("\nEnter data for node %d : ",++i);
 scanf("%d",&item);
 root = insertBST(root,item);
 break;
 case 2:
 printf("\nBST Traversal in INORDER \n");
 Inorder(root);
 break;
 case 3:
 printf("\nBST Traversal in PREORDER \n");
 Preorder(root);
 break;
 case 4:
 printf("\nBST Traversal in POSTORDER \n");
 Postorder(root);
break;
 case 5:
 printf("\n\n Terminating \n\n");
 break;

 case 6:printf("\nEnter which node is
delete:");
 scanf("%d",&key);
 deletenode(root,key);
 break;
 default:
 printf("\n\nInvalid Option !!! Try Again !!
\n\n");
 break;
 }
 } while(choice != 5);

 return 0;

}

struct node *insertBST(struct node *root, int
item)

{

 if(root == NULL)

 {

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

 root->left = root->right = NULL;

 root->data = item;

 return root;

 }

 else

 {

 if(item < root->data )

 root->left = insertBST(root->left,item);

 else if(item > root->data )

 root->right =insertBST(root->right,item);

 else

 printf(" Duplicate Element !! Not Allowed
!!!");

 return(root);

}

}

void Inorder(struct node *root)

{

 if( root != NULL)

 {

 Inorder(root->left);

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

 Inorder(root->right);

 }

}

void Preorder(struct node *root)

{

 if( root != NULL)

 {

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

 Preorder(root->left);

 Preorder(root->right);

 }

}

void Postorder(struct node *root)

{

 if( root != NULL)

 {

 Postorder(root->left);

 Postorder(root->right);

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

 }}


 struct node*deletenode(struct node*root, int
key){


if(root==NULL)

return root;

if(key<root->data)

root->left=deletenode(root->left,key);


else if(key>root->data)

root->right=deletenode(root->right,key);
```

```
else{


if((root->left==NULL)&& (root->right==NULL)){
free(root);
root=NULL;
return root;
}
 else if(root->left==NULL){
temp=root->right;
free(root);
return temp;
}
else if(root->right==NULL){
temp=root->left;
free(root);
return temp;
}
else{


temp=minvalue(root->right);
root->data=temp->data;
root->right=deletenode( root->right,temp-
>data);


}
}
return root;


}
struct node*minvalue(struct node*root){
temp=root;
while((temp!=NULL) &&(temp->left!=NULL))
 temp=temp->left;


return temp;
}
```