**Programs**

1. Write a program to create a binary search tree(BST)..

Ans:

#include<stdio.h>

#include<conio.h>

struct node

{

int data;

struct node \*left,\*right;

};

struct node \*createtree(struct node \*root,int data);

struct node \*deletetree(struct node \*root,int data);

struct node \*findmin(struct node \*root);

void preorder(struct node \*root);

void inorder(struct node \*root);

void postorder(struct node \*root);

struct node \*root=NULL;

void main()

{

int data,ch,i,n;

clrscr();

while(ch!=6)

{

printf("\n-----------------------------Menu--------------------------------");

printf("\n1.Insert \t2.Delete \t3.Inorder \t4.Preorder");

printf("\n5.Postorder \t6.Exit");

printf("\n-----------------------------------------------------------------");

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("Enter how many values do you want to insert:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("Enter element:");

scanf("%d",&data);

root=createtree(root,data);

}

break;

case 2:printf("Enter element to delete:");

scanf("%d",&data);

root=deletetree(root,data);

break;

case 3:printf("Inorder Traversal:");

inorder(root);

break;

case 4:printf("Preorder Traversal:");

preorder(root);

break;

case 5:printf("Postorder Traversal:");

postorder(root);

break;

case 6:exit(0);

default:printf("Invalid choice..");

}

}

}

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

{

if(root==NULL)

{

struct node \*temp;

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

temp->data=data;

temp->left=NULL;

temp->right=NULL;

return(temp);

}

if(data<(root->data))

root->left=createtree(root->left,data);

else if(data>root->data)

root->right=createtree(root->right,data);

return root;

}

void preorder(struct node \*root)

{

if(root!=NULL)

{

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

preorder(root->left);

preorder(root->right);

}

}

void inorder(struct node \*root)

{

if(root!=NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

void postorder(struct node \*root)

{

if(root!=NULL)

{

postorder(root->left);

postorder(root->right);

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

}

}

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

{

struct node \*temp;

if(root==NULL)

printf("Element not found..");

else if(data<root->data)

root->left=deletetree(root->left,data);

else if(data>root->data)

root->right=deletetree(root->right,data);

else

{

if(root->right&&root->left)

{

temp=findmin(root->right);

root->data=temp->data;

}

else

{

temp=root;

if(root->left==NULL)

root=root->right;

else if(root->right==NULL)

root=root->left;

free(temp);

printf("Element deleted..");

}

}

return root;

}

struct node \*findmin(struct node \*root)

{

if(root==NULL)

return NULL;

if(root->left)

return findmin(root->left);

else

return root;

}

Output:





