#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*left,\*right;

}\*root=NULL;

typedef struct node n;

n \*insert(int x,n \*root);

void preorder(n \*root);

void inorder(n \*root);

void postorder(n \*root);

n \*deletee(int x,n \*root);

void search(int x,n \*root);

n \*create\_node(int x);

n \*find\_min(n \*root);

void main()

{

int ch,x,key;

for( ; ; )

{

printf("menu\n 1.insert \n 2.delete\n3.search\n4.inorder\n5.preorder\n6.postorder\n7.exit\n");

printf("enter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("enter element to be inserted:");

scanf("%d",&x);

root=insert(x,root);break;

case 2:printf("enter element to be deleted:");

scanf("%d",&x);

root=deletee(x,root);break;

case 3:printf("enter element to be searched:");

scanf("%d",&key);

search(key,root);break;

case 4:inorder(root);break;

case 5:preorder(root);break;

case 6:postorder(root);break;

case 7:exit(0);break;

default:printf("wrong choice");

}

}

}

n \*insert(int x,n \*root)

{

if(root==NULL)

root=create\_node(x);

else

if(x<root->data)

root->left=insert(x,root->left);

else

if(x>root->data)

root->right=insert(x,root->right);

else

printf("insertion not possible");

return root;

}

n \*create\_node(int x)

{

n \*new;

new=(n \*)malloc(sizeof(n));

if(new==NULL)

{

printf("overflow");

return NULL;

}

new->data=x;

new->left=NULL;

new->right=NULL;

return new;

}

n \*deletee(int x,n \*root)

{

n \*temp;

if(root==NULL)

{

printf("no such node");

return root;

}

if(x<root->data)

root->left=deletee(x,root->left);

else

if(x>root->data)

root->right=deletee(x,root->right);

else

{

if(root->left==NULL||root->right==NULL)

{

temp=(root->left)?root->left:root->right;

free(root);

return temp;

}

temp=find\_min(root->left);

root->data=temp->data;

root->left=deletee(temp->data,root->left);

}

return root;

}

n \*find\_min(n \*root)

{

while(root->left!=NULL)

root=root->left;

return root;

}

void search(int x,n \*root)

{

if(root==NULL)

{

printf("key not found");

return;

}

if(x==root->data)

{

printf("key found");

return;

}

if(x<root->data)

search(x,root->left);

else

search(x,root->right);

}

void inorder(n \*root)

{

if(root!=NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

void preorder(n \*root)

{

if(root!=NULL)

{

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

preorder(root->left);

preorder(root->right);

}

}

void postorder(n \*root)

{

if(root!=NULL)

{

postorder(root->left);

postorder(root->right);

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

}

}