#include<stdio.h>

#include<malloc.h>

//structure definition

struct node

{

int info;

struct node \*lchild;

struct node \*rchild;

};

typedef struct node node;

//function declaration

void bst\_insert(int);

void displayPre(node\*);

void displayPost(node\*);

void displayIn(node\*);

void minTerm();

void maxTerm();

void search(int);

node \*root;

int main()

{

int c;

int ikey;

while(c!=9)

{

//menu driven program to choose from various functions

printf("\n\n1.Insert 3.Display preorder 4.Search\n5.Display postorder 6.Display Inorder\n7.Min Term 8.Max Term\n9.Exit\n");

scanf("%d",&c);

switch(c)

{

case 1:{

printf("Enter the value of ikey\n");

scanf("%d",&ikey);

bst\_insert(ikey);

}break;

case 2:{

printf("Enter the value to be deleted\n");

scanf("%d",&ikey);

// bst\_del(ikey);

}break;

case 3:{

if(root==NULL)

printf("The binary search tree is empty\n");

else

displayPre(root);

}break;

case 4:{

printf("Enter the ikey to be searched\n");

scanf("%d",&ikey);

search(ikey);

}break;

case 5:{

if(root==NULL)

printf("The binary search tree is empty\n");

else

displayPost(root);

}break;

case 6:{

if(root==NULL)

printf("The binary search tree is empty\n");

else

displayIn(root);

}break;

case 7:{

minTerm();

}break;

case 8:{

maxTerm();

}break;

case 9:{

printf("Exiting......\n");

}

}

}

return 0;

}

void displayPre(node \*ptr) //displays the pre order traversal

{

if(ptr==NULL)

return;

printf("%d ",ptr->info);

displayPre(ptr->lchild);

displayPre(ptr->rchild);

}

void displayIn(node \*ptr) //displays the inorder traversal

{

if(ptr==NULL)

return;

displayIn(ptr->lchild);

printf("%d ",ptr->info);

displayIn(ptr->rchild);

}

void displayPost(node \*ptr) //displays the post order traversal

{

if(ptr==NULL)

return;

displayPost(ptr->lchild);

displayPost(ptr->rchild);

printf("%d ",ptr->info);

}

void minTerm() //prints the min term

{

if(root==NULL)

{

printf("The binary tree is empty\n");

return;

}

else

{

node \*ptr=root;

while(ptr->lchild!=NULL)

{

ptr=ptr->lchild;

}

printf("%d is the min term of the binary tree\n",ptr->info);

return;

}

}

void maxTerm() //prints the max term

{

if(root==NULL)

{

printf("The binary tree is empty\n");

return;

}

else

{

node \*ptr=root;

while(ptr->rchild!=NULL)

{

ptr=ptr->rchild;

}

printf("%d is the max term of the binary tree\n",ptr->info);

return;

}

}

void bst\_insert(int ikey) //inserts the nodes in the binary tree

{

node \*tmp;

node \*ptr;

tmp=(node\*)malloc(sizeof(node));

tmp->info=ikey;

tmp->lchild=NULL;

tmp->rchild=NULL;

// tmp->lthread=false;

// tmp->rthread=false;

//when tree is empty

if(root==NULL)

{

root=tmp;

return;

}

//otherwise

else

{

ptr=root;

while(ptr!=NULL)

{

if(ptr->info==ikey)

{

printf("%d already exists\n",ikey);

return;

}

if(ikey<ptr->info)

{

if(ptr->lchild==NULL)

{

ptr->lchild=tmp;

printf("%d has been inserted \n",ikey);

return;

}

ptr=ptr->lchild;

}

else if(ikey>ptr->info)

{

if(ptr->rchild==NULL)

{

ptr->rchild=tmp;

printf("%d has been inserted \n",ikey);

return;

}

ptr=ptr->rchild;

}

}

return;

}

}

void search(int ikey) //used to find an element in the binary tree

{

node \*ptr;

int count=0;

if(root==NULL) //if no nodes exist

{

printf("The binary tree is empty\n");

return;

}

else //if nodes exist

{

ptr=root;

while(ptr!=NULL)

{

if(ptr->info==ikey)

{

printf("%d is found after %d iteration(s)\n",ikey,count);

return;

}

else

{

if(ikey<ptr->info)

{

ptr=ptr->lchild;

}

else if(ikey>ptr->info)

{

ptr=ptr->rchild;

}

count++;

}

}

printf("%d is not found the in binary tree\n",ikey);

}

}

OUTPUT:

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

1

Enter the value of ikey

4

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

1

Enter the value of ikey

7

7 has been inserted

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

1

Enter the value of ikey

78

78 has been inserted

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

1

Enter the value of ikey

56

56 has been inserted

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

1

Enter the value of ikey

2

2 has been inserted

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

3

4 2 7 78 56

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

5

2 56 78 7 4

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

6

2 4 7 56 78

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

1

Enter the value of ikey

89

89 has been inserted

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

3

4 2 7 78 56 89

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

4

Enter the ikey to be searched

6

6 is not found the in binary tree

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

4

Enter the ikey to be searched

4

4 is found after 0 iteration(s)

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

7

2 is the min term of the binary tree

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit

8

89 is the max term of the binary tree

1.Insert 3.Display preorder 4.Search

5.Display postorder 6.Display Inorder 7.Min Term

8.Max Term

9.Exit