**Program:**

#include <stdio.h>

#include <malloc.h>

struct node

{

int data;

struct node \*left;

struct node \*right;

};

typedef struct node node;

node \*tree = NULL;

void insertElement(node \*, int);

node \*deleteElement(node \*, int);

void deletetree(node \*);

node \* searchsmallest(node \*);

node \* searchlargest(node \*);

void preorderTraversal(node \*);

void inorderTraversal(node \*);

void postorderTraversal(node \*);

int totalnodes(node \*);

int externalnodes(node \*);

int internalnodes(node \*);

int height(node \*);

int main()

{

int num, opt, val, ans, ans1;

node \*ptr;

printf("\n Press 1 to Insert");

printf("\n Press 2 to Delete node");

printf("\n Press 3 to delete tree");

printf("\n Press 4 to search the smallest element");

printf("\n Press 5 to search the largest element");

printf("\n Press 6 to do PreOrder Traversal");

printf("\n Press 7 to do InOrder Traversal");

printf("\n Press 8 to do PostOrder Traversal");

printf("\n Press 9 to find total nodes");

printf("\n Press 10 to find leaf nodes");

printf("\n Press 11 to find internal nodes");

printf("\n Press 12 to find Height");

do

{

printf("\nEnter your option : ");

scanf("%d", &opt);

switch (opt)

{

case 1:

do

{

printf("\n Enter the value of the new node : ");

scanf("%d", &val);

insertElement(tree, val);

printf("\nPress 1 to add more numbers");

scanf("%d", &ans1);

} while (ans1 == 1);

break;

case 2:

printf("\n Enter node to be deleted : \n");

scanf("%d",&num);

tree=deleteElement(tree, num);

break;

case 3:

printf("\n Deleting tree : \n");

deletetree(tree);

break;

case 4:

ptr=searchsmallest(tree);

printf("\n Smallest elemnt is %d ",ptr->data);

break;

case 5:

ptr=searchlargest(tree);

printf("\n Largest elemnt is %d ",ptr->data);

break;

case 6:

printf("\n PreOrder Traversal : \n");

preorderTraversal(tree);

break;

case 7:

printf("\n InOrder Traversal : \n");

inorderTraversal(tree);

break;

case 8:

printf("\n PostOrder Traversal : \n");

postorderTraversal(tree);

case 9:

printf("\n Total nodes : %d ",totalnodes(tree));

break;

case 10:

printf("\n Total external nodes : %d ",externalnodes(tree));

break;

case 11:

printf("\n Total internal nodes : %d ",internalnodes(tree));

break;

case 12:

printf("\n Height of tree : %d ",height(tree));

break;

default:

printf("Wrong answer");

}

printf("\n Press 1 to choose another option");

scanf("%d", &ans);

} while (ans == 1);

}

void insertElement(node \*tree, int val)

{

node \*temp, \*parent, \*current;

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

temp->data = val;

temp->right = NULL;

temp->left = NULL;

parent = tree;

if (tree == NULL)

{

tree = temp;

tree->left = NULL;

tree->right = NULL;

}

else

{

current = tree;

while (current != NULL)

{

parent = current;

if (val > current->data)

current = current->right;

else

current = current->left;

}

if (val > parent->data)

parent->right = temp;

else

parent->left = temp;

}

}

node \*deleteElement(node \*, int x)

{

node \*parent, \*current, \*psuc, \*suc, \*ptr;

if (tree->left == NULL)

{

printf("\n Empty tree");

return (tree);

}

parent = tree;

current = tree->left;

while (current != NULL && x != current->data)

{

parent = current;

current = (x < current->data) ? current->left : current->right;

}

if (current == NULL)

{

printf("Value not present");

return (tree);

}

if(current->left==NULL)

ptr=current->right;

else if(current->right == NULL)

ptr=current->left;

else

{

psuc =current;

current=current->left;

while(suc->left!=NULL)

{

psuc = current;

suc = suc->left;

}

if(current==psuc)

{

suc->left=current->right;

}

else

{

suc->left=current->left;

psuc->left=suc->right;

suc->right=current->right;

}

ptr=suc;

}

if(parent->left==current)

parent->left=ptr;

else

parent->right=ptr;

free(current);

return tree;

}

void deletetree(node \*tree)

{

if(tree!=NULL)

{

deletetree(tree->left);

deletetree(tree->right);

free(tree);

}

}

node \* searchsmallest(node \* tree)

{

if( (tree == NULL) || (tree->left==NULL))

return tree;

else

return searchsmallest(tree->left);

}

node \* searchlargest(node \* tree)

{

if( (tree == NULL) || (tree->right==NULL))

return tree;

else

return searchlargest(tree->right);

}

void preorderTraversal(node \*tree)

{

if (tree != NULL)

{

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

preorderTraversal(tree->left);

preorderTraversal(tree->right);

}

}

void inorderTraversal(node \*tree)

{

if (tree != NULL)

{

inorderTraversal(tree->left);

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

inorderTraversal(tree->right);

}

}

void postorderTraversal(node \*tree)

{

if (tree != NULL)

{

preorderTraversal(tree->left);

preorderTraversal(tree->right);

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

}

}

int totalnodes(node \*)

{

if(tree==NULL)

return 0;

else

return(totalnodes(tree->left) + totalnodes(tree->right) + 1);

}

int externalnodes(node \*)

{

if(tree==NULL)

return 0;

else if((tree->left==NULL) && (tree->right==NULL))

return 1;

else

return(externalnodes(tree->left) + externalnodes(tree->right));

}

int internalnodes(node \*)

{

if((tree==NULL) || ((tree->left==NULL) && (tree->right==NULL)))

return 0;

else

return(internalnodes(tree->left) + internalnodes(tree->right) + 1);

}

int height(node \*)

{

int left,right;

if(tree==NULL)

return 0;

else

{

left=height(tree->left);

right=height(tree->right);

if(left > right)

return(left +1);

else

return(right +1);

}

}

**Output:**

Press 1 to Insert

Press 2 to Delete node

Press 3 to delete tree

Press 4 to search the smallest element

Press 5 to search the largest element

Press 6 to do PreOrder Traversal

Press 7 to do InOrder Traversal

Press 8 to do PostOrder Traversal

Press 9 to find total nodes

Press 10 to find leaf nodes

Press 11 to find internal nodes

Press 12 to find Height

Enter your option : 1

Enter the value of the new node : 12

Press 1 to add more numbers1

Enter the value of the new node : 5

Press 1 to add more numbers1

Enter the value of the new node : 20

Press 1 to add more numbers0

Press 1 to choose another option1

Enter your option : 4

Smallest element is 5

Press 1 to choose another option1

Enter your option : 5

largest element is 20

Press 1 to choose another option1

Enter your option : 6

PreOrder Traversal- 12 5 20

Press 1 to choose another option1

Enter your option : 7

PostOrder Traversal- 5 20 12

Press 1 to choose another option1

Enter your option : 8

InOrder Traversal- 5 12 20

Enter your option : 3

Deleting tree

Press 1 to choose another option 0