SSN COLLEGE OF ENGINEERING (Autonomous)
DEPARTMENT OF CSE
UCS308 Data Structures Lab
Assignment 8
Binary search tree

Register Number : 185001131
Name : Sai Charan B
Class : CSE – B

Data file: tree.h

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
typedef struct node{
   char name[100];
   struct node *left,*right;
}tree;
```

Function file: treefunc.h

```
#include “tree.h”
tree* makeempty(tree* t)
{
if (t!=NULL){
makeempty(t -> left);
makeempty(t -> right);
free(t);}
return NULL;
}

tree* find(char x[100],tree* t)
{
if (t==NULL){
   return NULL;}
if (strcmp(x , t -> name)<0){
   return find (x , t -> left);}
else if (strcmp(x , t-> name)>0){
   return find(x , t -> right);}
else{
   return t;}
}
```

```
tree* insert(char x[100], tree* t)
{
if(t==NULL)
     t=malloc(sizeof(tree));
else
{
strcpy(t -> name,x);
t -> left = NULL;
t -> right = NULL;}}
else if (strcmp(x , t -> name)<0)
t -> left = insert(x , t -> left);
else if(strcmp(x , t -> name)>0)
t -> right =insert (x , t -> right);
return t;}

void display(tree *t){
    if(t!=NULL){
        display(t -> left);
        printf("\n%s",t -> name);
        display(t -> right);}}

 tree* findmin(tree* t)
{
if(t==NULL)
    return NULL;
else if(t -> left == NULL)
    return t;
else
return findmin( t -> left);}
tree* delete(char x[100], tree* t){
tree* tmpcell = (tree*)malloc(sizeof(tree));
if(t==NULL)
    printf("Empty tree");
else if (strcmp(x , t -> name)<0)
    t -> left = delete(x , t -> left);
else if (strcmp(x ,t -> name)>0)
    t -> right = delete(x , t -> right);
else if(t -> left && t -> right){
 tmpcell = findmin(t -> right);
 strcpy(t -> name , tmpcell -> name);
 t -> right = delete(t -> name,t -> right);
}
else{
```

```
tmpcell=t;
if(t -> left==NULL)
   t = t -> right;
else if(t->right==NULL)
   t = t -> left;
free(tmpcell);
}
return t;}

void findgp(tree* t,char name[100]){
         if(t == NULL || (!t -> left && !t -> right) )
                  return;
         if(t -> left){
                  if(t -> left -> left){
                           if(strcmp(t -> left -> left -> name,name) == 0){
                                    printf("\n\nGrandparent of %s is %s",name,t -> name);
                                    return;}}
                  if(t -> left -> right){
                           if(strcmp(t -> left -> right -> name,name) == 0){
                                             printf("\n\nGrandparent of %s is %s",name,t -> name);
                                    return;}}
                  findgp(t -> left,name);}
         if(t -> right){
                  if(t -> right -> left){
                           if(strcmp(t -> right -> left -> name,name) == 0){
                                             printf("\n\nGrandparent of %s is %s",name,t -> name);
                                    return;}}
                  if(t -> right -> right){
                           if(strcmp(t -> right -> right -> name,name) == 0){
                                             printf("\n\nGrandparent of %s is %s",name,t -> name);
                                    return;}}
                  findgp(t -> right,name);}}

void findgc(tree *t , char name[100]){
   if(t==NULL)
      return;
   if(strcmp(t -> name,name)==0){
      if(t -> left){
         if(t -> left -> left){
            printf("\n\nGrandchild of %s is %s",name,t -> left -> left -> name);}}
   if(t -> left){
      if(t -> left -> right){
         printf("\n\nGrandchild of %s is %s",name,t -> left -> right -> name);}}
```

```
    if(t -> right){
        if(t -> right -> left){
            printf("\n\nGrandchild of %s is %s",name,t -> right -> left -> name);}}
    if(t -> right){
        if(t -> right -> right){
            printf("\n\nGrandchild of %s is %s",name,t -> right -> right -> name);}}
    }
        findgc(t -> left , name);
        findgc(t -> right,name);
}
void findsg(tree* t,char name[100]){
            if(t == NULL)
                        return;
            if(t -> left && t -> right){
                        if(strcmp(t -> left -> name,name) == 0){
                                    printf("\n\nSibling of %s is %s",name,t -> right -> name);
                                    return;}
                        if( strcmp(t -> right ->name,name) == 0){
                                    printf("\n\nSibling of %s is %s",name,t -> left -> name);
                                    return;
                        }}
            findsg(t -> left,name);
            findsg(t -> right,name);
}
```

Main file

```
#include “treefunc.h”
void main(){
    tree *t = (tree*)malloc(sizeof(tree));
    tree *temp = (tree*)malloc(sizeof(tree));
    int n = 12;
    for(int i=0;i<n;i++){
        char s[100];
        printf("Enter name : ");
        scanf("%s",s);
        insert(s,t);}
    printf("\nNames in alphabetical order\n");
    display(t);
    findgp(t,"kartika");
    findgp(t,"lakshmi");
    findgc(t,"charan");
    findsg(t,"swetha");
    findsg(t,"chitra");
    delete("ram",t);
    printf("\n\nAfter deleting Ram");
    display(t);}
```
Output :

```
Enter name : kumar
Enter name : anusha
Enter name : ram
Enter name : charan
Enter name : mohan
Enter name : kartika
Enter name : chitra
Enter name : lakshmi
Enter name : abishek
Enter name : swetha
Enter name : tarun
Enter name : Sanjana

Names in alphabetical order

abishek
anusha
charan
chitra
kartika
kumar
lakshmi
mohan
ram
sanjana
swetha
tarun

Grandparent of kartika is anusha

Grandparent of lakshmi is ram

Grandchild of charan is chitra

Sibling of swetha is mohan

After deleting Ram
```

abishek
anusha
charan
chitra
kartika
kumar
lakshmi
mohan
sanjana
swetha
tarun