```
//M Gayathri
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
typedef struct treenode *position;
typedef struct treenode *searchtree;
struct treenode
{
	char element[20];
	searchtree left;
	searchtree right;
};
void grand(char e[],searchtree t)
{
	if((t->left!=NULL) && (strcmp(e,t->element)<0))
	{
		if((t->left->right!=NULL) && (strcmp(e,t->left->right->element)==0))
				printf("\ngrand parent of %s is %s",e,t->element);
		else if((t->left->left!=NULL) && (strcmp(e,t->left->left->element)==0))
				printf("\ngrand parent of %s is %s",e,t->element);
		else
			grand(e,t->left);
	}

	else if((t->right!=NULL) && (strcmp(e,t->element)>0))
	{
		if((t->right->left!=NULL) && (strcmp(e,t->right->left->element)==0))
				printf("\ngrand parent of %s is %s",e,t->element);

		else if((t->right->right!=NULL) && (strcmp(e,t->right->right->element)==0))
				printf("\ngrand parent of %s is %s",e,t->element);
		else
			grand(e,t->right);
	}

}
searchtree findparent(char e[],searchtree t)
{
	if(t->left!=NULL && strcmp(t->left->element,e)==0)
		return t;
	else if(t->right!=NULL && strcmp(t->right->element,e)==0)
		return t;
```

```
        else if(strcmp(e,t->element)<0)
                return findparent(e,t->left);
        else
                return findparent(e,t->right);
}
void sib(char e[],searchtree t)
{
        t=findparent(e,t);
        printf("\n Siblings of %s are\n",e);
        if(t->left!=NULL)
                printf("\n\t%s",t->left->element);
        if(t->right!=NULL)
                printf("\n\t%s",t->right->element);
}
searchtree find(char x[],searchtree t)
{
        if(t==NULL)
                return NULL;
        if (strcmp(x,t->element)<0)
                return find(x,t->left);
        if (strcmp(x,t->element)>0)
                return find(x,t->right);
        else
                return t;
}
void inorder(searchtree t)
{
        if(t->left!=NULL)
                inorder(t->left);
        printf("\n%s",t->element);
        if(t->right!=NULL)
                inorder(t->right);
}

void grandch(char e[],searchtree t)
{
        t=find(e,t);
        printf("\ngrandchildren are");
        if(t->left!=NULL)
        {
                if(t->left->right!=NULL)
                                printf("\n%s",t->left->right->element);
                if(t->left->left!=NULL)
```

```
                                printf("\n%s",t->left->left->element);
        }

        else if(t->right!=NULL)
        {
                if(t->right->left!=NULL)
                                printf("\n%s",t->right->left->element);

                if(t->right->right!=NULL)
                                printf("\n%s",t->right->right->element);
        }

}

position findmin(searchtree t)
{
        if(t==NULL)
                return NULL;
        else if(t->left==NULL)
                return t;
        else
                return findmin(t->left);
}
position findmax(searchtree t)
{
        if(t!=NULL)
                while(t->right!=NULL)
                        t=t->right;
        return t;
}
searchtree insert(char x[],searchtree t)
{
        if(t==NULL)
        {
                t=(struct treenode*)malloc(sizeof(struct treenode));
                if(t==NULL)
                        printf("\n out of space");
                else
                {
                        strcpy(t->element,x);
                        t->left=t->right=NULL;
                }
        }
```

```
        else if(strcmp(x,t->element)<0)
                t->left=insert(x,t->left);
        else if(strcmp(x,t->element)>0)
                t->right=insert(x,t->right);
        return t;
}
searchtree delete(char x[],searchtree t)
{
        position tmpcell;
        if(t==NULL)
                printf("element not found");
        else if(strcmp(x,t->element)<0)
                t->left=delete(x,t->left);
        else if(strcmp(x,t->element)>0)
                t->right=delete(x,t->right);
        else if(t->left && t->right)
        {
                tmpcell=findmin(t->right);
                strcpy(t->element,tmpcell->element);
                t->right=delete(t->element,t->right);
        }
        else
        {
                if(t->left==NULL)
                        t=t->right;
                else if(t->right==NULL)
                        t=t->left;
                free(tmpcell);
        }
        return t;
}
int main()
{
        int n;
        char a[20];
        printf("enter no of elements");
        scanf("%d",&n);
        searchtree tree=NULL;
        printf("\n enter names\n");
        for(int i=0;i<n;i++)
        {
                scanf("%s",a);
                tree=insert(a,tree);
```

```
        }
        printf("\n The names are\n");
        inorder(tree);
        printf("\n");
        grand("lakshmi",tree);
        printf("\n");
        grand("karthika",tree);
        printf("\n");
        grandch("charan",tree);
        printf("\n");
        sib("swetha",tree);
        printf("\n");
        printf("\n");
        sib("chitra",tree);
        tree=delete("ram",tree);
        printf("\n deleted ram\n");
        printf("\n elements are");
        inorder(tree);
        return 0;
}


/*SAMPLE INPUT/OUTPUT

enter no of elements12

 enter names
kumar
anusha
ram
charan
mohan
karthika
chitra
lakshmi
abishek
swetha
tarun
sanjana

 The names are

abishek
```

anusha
charan
chitra
karthika
kumar
lakshmi
mohan
ram
sanjana
swetha
tarun

grand parent of lakshmi is ram

grand parent of karthika is anusha

grandchildren are
chitra

Siblings of swetha are

mohan
swetha

Siblings of chitra are

chitra
deleted ram

elements are
abishek
anusha
charan
chitra
karthika
kumar
lakshmi
mohan
sanjana
swetha
tarun

```
*/
```