```
//Marc Pfeiffer
//Homework # 5 - Family Tree
#include <iostream>
#include <fstream>
using namespace std;

ifstream dataIn("infile.txt");
ofstream dataOut("outfile.txt");

struct Node{

    string name;
    string father;
    Node *son, *brother;

};
void inorder(Node *tree){

    if(tree){
        dataOut<<tree->name<<endl;
        inorder(tree->son);

        inorder(tree->brother);

    }

}
Node* findPerson(Node* tree, string name){


    if (name == tree->name){

        return tree;
    }
     Node* temp = NULL;
    if(tree->son!= NULL){

        temp = findPerson(tree->son, name);

    }
    if(temp != NULL){
        return temp;
    }
    if(tree->brother != NULL){
        temp = findPerson(tree->brother, name);
    }
    if(temp != NULL){
        return temp;
    }
    return NULL;
}


void insertPerson(Node *&tree, string name, string father){

    if(!tree){
        Node* temp = new Node;
        temp->name = name;
        temp->father= father;
        temp->son= NULL;
        temp->brother= NULL;
        tree = temp;
    }
    else if(father == tree->name){

        insertPerson(tree->son,name, father);
```

```cpp
    }
    else if( father  == tree->father){
        insertPerson(tree->brother, name, father);
    }
    else{
        Node* temp;
        temp = findPerson(tree, father);
        insertPerson(temp->son, name, father);
    }
}

Node* findParent(Node* tree, string name){

    Node* temp = findPerson(tree, name);

    if(!temp){
            return NULL;
    }
    Node* parent = findPerson(tree, temp->father);
    if(!parent ){

        return NULL;

    }
    else {
        return parent;
    }
}
void findSons(Node* tree){

    if(tree->son == NULL){
        dataOut<<tree<<" has no sons\n\n";
    }
    else{
        Node *temp = tree->son;
        dataOut<<tree->name<<"'s sons are:\n";
        dataOut<<temp->name;
        while(temp->brother!=NULL){
             temp =temp->brother;
             dataOut<<"\n"<<temp->name;
        }
        dataOut<<endl;


    }

}
void oldestSon(Node *tree){

    if(tree->son == NULL){
        dataOut<<tree->name<<" has no sons\n\n";
    }
    else{
    dataOut<<tree->name<<"'s oldest son is "<<tree->son->name<<endl;
    }
}
void yongestSon(Node* tree){

    if(tree->son == NULL){
        dataOut<<tree->name<<" has no sons\n\n";
    }
    else{

       Node* temp= tree->son;
       while(temp->brother!= NULL){

```

```cpp
                temp = temp->brother;
            }
            dataOut<<tree->name<<"'s youngest son is "<<temp->name<<endl;
        }
}

void brothers(Node* tree, string name){

    Node* person = findPerson(tree, name);
    Node* father = findParent(tree, name);

    if(father ==NULL){
            dataOut<<person-> name<<"is the root :: no brothers\n\n";
            return;
    }
    Node* brother = father->son;
    if(brother!=person){

        dataOut<<brother->name<<" is a brother of " << person->name<<endl;

    }
    else if (brother->brother == NULL){

            dataOut<<person-> name<<" has no brothers\n\n";
    }
    while(brother->brother!= NULL){

        brother = brother->brother;
        if(brother!= person){
            dataOut<<brother->name<<" is a brother of " << person->name<<endl;
        }

    }
}

void  oldestBrother(Node* tree, string name){

    Node* person = findPerson(tree, name);
    Node* father = findParent(tree, name);

    if(father == NULL){

        dataOut<<name<< "has no brother\n";
    }
    else{
        if(father->son->name == name){
            dataOut<<name<< " is oldest"<<endl;

        }
        else{
            dataOut<<father->son->name <<" is the oldest brother of " << name << 
endl;
        }

    }
}

void youngestBrother(Node* tree, string name){

    Node* person = findPerson(tree, name);
    Node* father = findParent(tree, name);

    if(father == NULL){

        dataOut<<name<< "has no brother\n";
    }
    Node* brother = father->son;
    while(brother->brother!= NULL){

        brother= brother->brother;

    }
    if(brother == person){

        dataOut<<person->name<<" is youngest\n";
    }
    else{
        dataOut<<brother->name<<" is "<< name<< "'s youngest brother\n";
    }
}

void uncles( Node* tree, string name){

    Node* person = findPerson(tree, name);
    Node* father = findParent(tree, name);

    if(father == NULL){

        dataOut<<name<< "has no uncles\n";
        return;
    }
    Node* grandpa = findParent(tree, father->name);
    if(grandpa == NULL){

        dataOut<<name<< "has no uncles\n";
        return;
    }
    Node* uncle = grandpa->son;
    if(uncle!= father){
        dataOut<<uncle->name<<" is "<< name<<"'s uncle\n";
    }
    else
    {
        if(uncle->brother == NULL){
            dataOut<<name<< "has no uncles\n";
            return;
        }
    }
    while(uncle->brother!= NULL){

        uncle = uncle->brother;
        if(uncle != father){
            dataOut<<uncle->name<<" is "<< name<<"'s uncle\n";
        }

    }
}

Node* findGrandpa(Node* tree, string name){

    Node* parent = findParent(tree, name);
    if(parent == NULL){
        return NULL;
    }
    Node* grandpa = findParent(tree, parent->name);
    if(grandpa== NULL){

        return NULL;
    }
    else{
        return grandpa;
    }
```

```cpp
}
int main()
{
    Node *tree = NULL;
    string  name, father;


    dataIn>>name;

    while(name != "xxxx"){
        dataIn>> father;
            insertPerson(tree, name, father);
            dataIn>>name;
        }
    inorder(tree);

    cin>>name;
    Node* person = findPerson(tree, name);
    Node* parent = findParent(tree, name);
    Node* grandpa = findGrandpa(tree, name);
    dataOut<<endl<<endl<<name<<"'s father is "<<parent->name<<endl;
    if(parent->father == "0"){
        dataOut<<"no known grandpa";
    }
    else{
        Node* grandpa = findGrandpa(tree, name);
        dataOut<<name<<"'s grandpa is "<<grandpa->name<<endl<<endl;
    }
    findSons(person);
    oldestSon(person);
    yongestSon(person);
    brothers(tree, name);
    oldestBrother(tree, name);
    youngestBrother(tree,name);
    uncles(tree, name);



    return 0;
}
```