

```

1 //Marc Pfeiffer
2 //Homework # 5 - Family Tree
3 #include <iostream>
4 #include <fstream>
5 using namespace std;
6
7 ifstream dataIn("infile.txt");
8 ofstream dataOut("outfile.txt");
9
10 struct Node{
11
12     string name;
13     string father;
14     Node *son, *brother;
15
16 };
17 void inorder(Node *tree){
18
19     if(tree){
20         dataOut<<tree->name<<endl;
21         inorder(tree->son);
22
23         inorder(tree->brother);
24
25     }
26
27 }
28 Node* findPerson(Node* tree, string name){
29
30
31     if (name == tree->name){
32
33         return tree;
34     }
35     Node* temp = NULL;
36     if(tree->son!= NULL){
37
38         temp = findPerson(tree->son, name);
39
40     }
41     if(temp != NULL){
42         return temp;
43     }
44     if(tree->brother != NULL){
45         temp = findPerson(tree->brother, name);
46     }
47     if(temp != NULL){
48         return temp;
49     }
50     return NULL;
51 }
52
53
54 void insertPerson(Node *&tree, string name, string father){
55
56     if(!tree){
57         Node* temp = new Node;
58         temp->name = name;
59         temp->father= father;
60         temp->son= NULL;
61         temp->brother= NULL;
62         tree = temp;
63     }
64     else if(father == tree->name){
65
66         insertPerson(tree->son,name, father);

```

```

67     }
68     else if( father == tree->father){
69         insertPerson(tree->brother, name, father);
70     }
71     else{
72         Node* temp;
73         temp = findPerson(tree, father);
74         insertPerson(temp->son, name, father);
75     }
76 }
77
78 Node* findParent(Node* tree, string name){
79
80     Node* temp = findPerson(tree, name);
81
82     if(!temp){
83         return NULL;
84     }
85     Node* parent = findPerson(tree, temp->father);
86     if(!parent ){
87
88         return NULL;
89
90     }
91     else {
92         return parent;
93     }
94 }
95 void findSons(Node* tree){
96
97     if(tree->son == NULL){
98         dataOut<<tree<<" has no sons\n\n";
99     }
100    else{
101        Node *temp = tree->son;
102        dataOut<<tree->name<<"'s sons are:\n";
103        dataOut<<temp->name;
104        while(temp->brother!=NULL){
105            temp =temp->brother;
106            dataOut<<"\n"<<temp->name;
107        }
108        dataOut<<endl;
109
110
111    }
112
113 }
114 void oldestSon(Node *tree){
115
116     if(tree->son == NULL){
117         dataOut<<tree->name<<" has no sons\n\n";
118     }
119     else{
120         dataOut<<tree->name<<"'s oldest son is "<<tree->son->name<<endl;
121     }
122 }
123 void yongestSon(Node* tree){
124
125     if(tree->son == NULL){
126         dataOut<<tree->name<<" has no sons\n\n";
127     }
128     else{
129
130         Node* temp= tree->son;
131         while(temp->brother!= NULL){
132

```

```

133         temp = temp->brother;
134     }
135     dataOut<<"tree->name<<"'s youngest son is " <<temp->name<<endl;
136 }
137 }
138
139 void brothers(Node* tree, string name){
140
141     Node* person = findPerson(tree, name);
142     Node* father = findParent(tree, name);
143
144     if(father ==NULL){
145         dataOut<<"person-> name<<"is the root :: no brothers\n\n";
146         return;
147     }
148     Node* brother = father->son;
149     if(brother!=person){
150
151         dataOut<<"brother->name<<" is a brother of " << person->name<<endl;
152
153     }
154     else if (brother->brother == NULL){
155
156         dataOut<<"person-> name<<" has no brothers\n\n";
157     }
158     while(brother->brother!= NULL){
159
160         brother = brother->brother;
161         if(brother!= person){
162             dataOut<<"brother->name<<" is a brother of " << person->name<<endl;
163         }
164     }
165 }
166 }
167
168 void oldestBrother(Node* tree, string name){
169
170     Node* person = findPerson(tree, name);
171     Node* father = findParent(tree, name);
172
173     if(father == NULL){
174
175         dataOut<<"name<< "has no brother\n";
176     }
177     else{
178         if(father->son->name == name){
179             dataOut<<"name<< " is oldest"<<endl;
180
181         }
182         else{
183             dataOut<<"father->son->name <<" is the oldest brother of " << name <<
endl;
184         }
185     }
186 }
187 }
188
189 void youngestBrother(Node* tree, string name){
190
191     Node* person = findPerson(tree, name);
192     Node* father = findParent(tree, name);
193
194     if(father == NULL){
195
196         dataOut<<"name<< "has no brother\n";
197     }

```

```

198 Node* brother = father->son;
199 while(brother->brother!= NULL){
200     brother= brother->brother;
201 }
202 }
203 if(brother == person){
204     dataOut<<person->name<<" is youngest\n";
205 }
206 else{
207     dataOut<<brother->name<<" is "<< name<<"'s youngest brother\n";
208 }
209 }
210 }
211 }
212
213 void uncles( Node* tree, string name){
214     Node* person = findPerson(tree, name);
215     Node* father = findParent(tree, name);
216
217     if(father == NULL){
218         dataOut<<name<<"has no uncles\n";
219         return;
220     }
221     Node* grandpa = findParent(tree, father->name);
222     if(grandpa == NULL){
223         dataOut<<name<<"has no uncles\n";
224         return;
225     }
226     Node* uncle = grandpa->son;
227     if(uncle!= father){
228         dataOut<<uncle->name<<" is "<< name<<"'s uncle\n";
229     }
230     else
231     {
232         if(uncle->brother == NULL){
233             dataOut<<name<<"has no uncles\n";
234             return;
235         }
236     }
237     while(uncle->brother!= NULL){
238         uncle = uncle->brother;
239         if(uncle != father){
240             dataOut<<uncle->name<<" is "<< name<<"'s uncle\n";
241         }
242     }
243 }
244 }
245 }
246 }
247 }
248 }
249
250 Node* findGrandpa(Node* tree, string name){
251     Node* parent = findParent(tree, name);
252     if(parent == NULL){
253         return NULL;
254     }
255     Node* grandpa = findParent(tree, parent->name);
256     if(grandpa== NULL){
257         return NULL;
258     }
259     else{
260         return grandpa;
261     }
262 }
263

```

```

264 }
265 int main()
266 {
267     Node *tree = NULL;
268     string name, father;
269
270
271     dataIn>>name;
272
273     while(name != "xxx") {
274         dataIn>> father;
275         insertPerson(tree, name, father);
276         dataIn>>name;
277     }
278     inorder(tree);
279
280     cin>>name;
281     Node* person = findPerson(tree, name);
282     Node* parent = findParent(tree, name);
283     Node* grandpa = findGrandpa(tree, name);
284     dataOut<<endl<<endl<<name<<"'s father is "<<parent->name<<endl;
285     if(parent->father == "0") {
286         dataOut<<"no known grandpa";
287     }
288     else {
289         Node* grandpa = findGrandpa(tree, name);
290         dataOut<<name<<"'s grandpa is "<<grandpa->name<<endl<<endl;
291     }
292     findSons(person);
293     oldestSon(person);
294     youngestSon(person);
295     brothers(tree, name);
296     oldestBrother(tree, name);
297     youngestBrother(tree, name);
298     uncles(tree, name);
299
300
301
302     return 0;
303 }

```