CSC 1800
Organization of Programming Languages
Fall 2016

Programming Assignment #2
Out: 12 September 2016
Due: 11:59:59 pm, Wednesday, 5 October 2016

Introduction

This project deals with *family trees*. A family tree is a set of individuals who are "connected" to each other either by marriage, sibling, or descendant relationships. You will write a Python program to solve the problem of determining how two people are related to each other. We will be using Python 3 in this project, NOT PYTHON 2. You can get the Python interpreter at http://www.python.org

Problem Background

Another way of looking at a family tree is to view it as a directed graph. The nodes represent individuals, connected by edges (arrows) that represent the "offspring" relationship. An edge from one node to another means that the latter node is an offspring of the former node. In general, we can say one node (individual A) is a *descendant* of another node (individual B) if and only if (iff) there is a directed sequence of edges from node B to node A. The inverse of this relationship is the *ancestor* relationship – that is, node A is the ancestor of node B in this example.

Defining the concept of "related:"
Nodes in a family tree have at most two edges in (one for each parent), and an arbitrary number of edges out (one for each offspring). The *distance* between an ancestor node and a descendant node is the number of edges in the graph between the two nodes. The distance between a parent and child is 1; the distance between a grandparent and grandchild is 2. Two people are *related* if they share a common ancestor or if one is the parent of the other. That is, if there exists a person from whom they are both descended. Note that two related people are not necessarily the same distance from their common ancestor.

For this problem, you must design and write a Java program that will read an input file containing a family graph, and then answer questions about relationships between people in the graph. Specifically, you'll be asked to answer "Yes/No" questions about whether two people "A" and "B" share the following relationships:

a) Is "A" married to "B"
b) Is "A" a parent of "B"
c) Is "A" a sibling of "B"
d) Is "A" a half-sibling of "B"
e) Is "A" an ancestor of "B"
f) Is "A" a cousin of "B"

You can also be asked to answer the questions, "What is the closest relationship between A and B?" and "Who is/are the people who are A's <relation>?" **For this problem assume relationships are ranked according to the list above, with "a" being the closest relationship, "e" being the furthest, and "unrelated" being the equivalent of "none of the above."**

Cousins

An interesting element here is the category of "first/second/third/etc-cousin, once/twice/thrice/etc. removed." To begin with, suppose the following facts:

A and B are siblings,
C is a child of A,
D is a child of B,

Then C and D are *first cousins*. Proceeding logically, if C has a child X and D has a child Y, then X and Y are *second cousins*. However, what is the relationship between C and Y? Why, C and Y are *first cousins, once removed!* For our purposes in the current project, we'll just assume that A and Y are simply cousins.

Uncles and aunts are also pretty straightforward to recognize. For the purposes of this current project, we'll lump aunts, uncles, and cousins together in the category called *cousin*. Another way of looking at the *cousin* relationship is that it is the same as the first part of the *related* definition: they share a common ancestor.

Re-Marriages

Another interesting twist in this problem is that you'll have to be careful of second marriages – do not assume that a person can have at most one spouse. In the case of second marriages, the children from each spouse's earlier marriage are NOT siblings, but new children produced after the marriage are (half-)siblings of the earlier marriages' offspring. In our project we'll have to recognize that there can be remarriages, and indicate that two people with only a single parent in common are half-siblings.

Problem I/O Specification

The family tree will be specified in a file that will be sent to your program via **standard in.** ***You must not write the Java program so that it always looks at some hard-coded data file name or so that you have to supply the name as a command line argument.***

You can assume that there will be no formatting errors to check for. Each line in the family tree will have the form

E <name1> <name2> [<name3>]

which has the meaning "Event: <name1> and <name2> are the married parents of <name3>." Note that marriage events and birth events are listed chronologically, and that not all marriages produce children. Names are just first names, have no hyphens, and are less than 20 characters long.

The family tree file will also have query lines (possibly interspersed with event lines). These will have the form

QUERY	Meaning
R <name1> <name2>	How are <name1> and <name2> most closely related?
X <name1> <relation> <name2>	Is <name1> the <relation> of <name2>?
W <relation> <name1>	List everyone who is the <relation> of <name1>.

<relation> can be **spouse, parent**, **sibling**, **half-sibling, ancestor**, **cousin,** or **unrelated**.

Whenever the program encounters a query, it must answer according to the knowledge so far collected as events in the input file. No output is required when the program encounters an event line. When a query line is encountered, the program must print out a blank line, then print the query, then on the next line print out the response. The response to "R" queries must be a relationship or **unrelated**. The response to "X" queries must be either "Yes" or "No." The response to "W" queries must be all the names, one per line, that correctly answer the question. The names must be sorted in alphabetical order.

Example I/O

File Content:

```
E John Mary Bill
E John Mary Pete
E John Mary Fred
E Alex Jean Rebecca
E Rebecca Bill Andrew
E Pete Carol Jim
W parent Pete
X Bill sibling Pete
X Bill sibling Fred
R John Mary
W ancestor Andrew
X Bill cousin Mary
R Cathy Todd
```

Output:

```
W parent Pete
John
Mary

X Bill sibling Pete
Yes

X Bill sibling Fred
Yes

R John Mary
spouse

W ancestor Andrew
Alex
Bill
Jean
John
Mary
Rebecca

X Bill cousin Mary
No

R Cathy Todd
unrelated
```

Hints

You'll first need to decide how to represent people – what kind of class definition will you use? What slots and methods should "people objects" have? You'll then have to decide how to organize all of the "people objects" into a graph. Remember that slots on an object can hold things like linked lists and arrays. You also must be careful to set things up so that you can easily add facts to the graph as you read in new events. I'd recommend some kind of hash table or array of people objects. You'll also need to write methods for answering each of the queries. Most of these will be some kind of graph-traversal algorithm (breadth-first or depth-first), although some could be simple checks of the contents of a single slot on a person object. Note that you'll need to keep around a linked list of nodes seen so far during your traversals when you're looking for common ancestors! Finally, you might find that you don't need a method for each query, but rather a smaller number of very generic methods might be able to solve all of the queries.

What Needs to Be Handed In

Each team must hand in a program source code listing, with all team members as a comment. The source code listing must be sent via email to the course TA, Andrew Keenan, at akeena03@villanova.edu. Within 24 hours of handing in the source code listing, the team must email a 4 page report (line-and-a-half spacing) describing how the project was organized, what problems were encountered, and how they were solved. The report must also describe what each team member contributed to the effort. Finally, the report must explain how the program was tested to verify that it worked.

Plagiarism Policy

This programming project is teamwork, but all other work in the course is individual. In teamwork projects I expect you to discuss any programming issues, and to share work *related to the project*. On individual projects, you cannot share work (this includes copying answers, sharing files, or paraphrasing each others' answers), although I have no problems with two or more students just verbally discussing questions from homeworks. The key here is that as long as you can justify to me that the written work you submit is the result of your own thinking and writing, you will not be subject to plagiarism accusations. If you use code from the Internet to solve parts of the problem, you must give proper attribution. You may not use code from the Internet (or from other students not on your team) that completely solves the project, only parts of it.

Plagiarism will result minimally in a "0" for the project/homework, and maximally an "F" for the course.