CS360 -- Lab 2 - Libfdr Primer - "Famtree"

- James S. Plank
- CS360
- Url: http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/labs/Lab-2-Famtree/index.html

---

Note: When you do this lab, make directories for yourself called **src** and **bin**. Put your **famtree.c** program into **src**, and the **makefile** in this directory will compile the executable into **bin/famtree**. The gradescripts assume that you have this directory structure.

What you submit

You are to submit the file **famtree.c**. The TA will copy this into his/her **src** directory, and will use the **makefile** in the lab directory to create the executable **bin/famtree**.

Famtree

This is a lab that makes sure that you have red-black trees, dlists and fields down.

Your job in this lab is to write the program **bin/famtree**. This program takes a description of people and their relationships to one another on standard input. The descriptions must be in a special format, which will be described below. An example is in the file **data/fam1txt**, which represents a family composed of Bob and Nancy, and their three chidren Frank, Lester and Diana.

bin/famtree takes such a description, and prints out complete information on all of the people in the file. This consists of, for each person, their sex (if known), their father, mother and children. Therefore **data/fam1output.txt** contains valid output of **bin/famtree** on **data/fam1.txt**.

The format of the input file is as follows. Lines must either be blank, or have one of following first words:

- **PERSON**: This specifies a person. Following **PERSON** is the person's name, which may be any number of words. **Famtree** should assume that names consist of words with single spaces between them.
- **SEX**: This specifies the person's sex. It can be either **M** or **F**.
- **FATHER**: This specifies the person's father. Following **FATHER** is the father's name, which may be any number of words. It implies that the father is male. A person may only have one father.
- **MOTHER**: This specifies the person's mother. Following **MOTHER** is the mother's name, which may be any number of words. It implies that the mother is female. A person may only have one mother.
- **FATHER_OF**: This specifies that the person is male, and that the specified person is one of the person's children. A person may have any number of children.
- **MOTHER_OF**: This specifies that the person is female, and that the specified person is one of the person's children.

People may be specified in any order in the input file.

bin/famtree has two other features. First, it prints out the people in a structured order. That is, no person can be printed out until both of their parents have been printed out. If this is impossible (as, for example, in **data/cyclefam.txt**), then **bin/famtree** identifies this fact before it prints out people. Beyond this one rule, people may be printed in any order.

The second feature of **bin/famtree** is that it allows for redundancy, but it infers as much as it can. For example, **data/redundant.txt** has a few lines that are redundant. Line 3 is redundant because Fred must be male by virtue of being Joe's father. Moreover, line 7 is redundant because line 2 already specified Fred as Joe's father. The file **data/nonredundant.txt** is the minimal file specifying the same family.

Specifically -- the output should look as follows:

Each person should have the following information printed, each on a different line.

- Their name.
- The word **Sex:** and the person's sex: **Male**, **Female** or **Unknown**
- The word **Father:** and name of the person's father. If that is unknown, print **Unknown** for the father's name.
- The word **Mother:** and name of the person's mother. If that is unknown, print **Unknown** for the mother's name.
- The word **Children:** and if the person has no children, then **None**.
- If the person does have children then print the name of each child on its own line.
- End the person with a blank line.
- My output does some indentation. I don't care if yours does.

You should check for the following errors and print error messages on standard error that match mine:

- Specifying two fathers or two mothers.
- Specifying a father when the person is female.
- Specifying a mother when the person is male.
- Specifying that a person is female when the person is already known to be male.
- Specifying that a person is male when the person is already known to be female.
- Having a cycle in the specification.

The cycle detection should be performed after the other checks. The other checkss should be flagged as they are read in.

Working example

A working example of **bin/famtree** is in **~jplank/cs360/labs/Lab-2-Famtree/bin/famtree**. Try it out on the input files in this directory. Other input files are:

- **data/fam2.txt** a more complex family.
- **data/fam3.txt** A family with 10 generations of single parents.
- **data/fam4.txt** A file with an error in it.
- **data/fam5.txt** A file with another error in it.

You should make your output work exactly like **bin/famtree**'s.

Grading

There is a standard **gradescript** and **gradeall** in the lab directory. The gradescript makes use of the program **bin/grader** in the lab directory.

Help

You should have a **struct** for a person (mine is called a **Person**). That struct will have the following fields:

- Name
- Sex
- Father
- Mother

- List of children
- Some stuff for depth first search and topological sort (e.g. a visited field and/or number of unprinted parents).

Your program will work in three phases:

1. **Reading the information into the structs.** You should have a red-black tree (mine is called **people**) that contains all the people. It is keyed on each person's name, and the **val** fields are the person's **Person** structs. You use an **IS** to read standard input.

   Each time you read a line that has a name (i.e. **PERSON**, **FATHER**, **MOTHER**, **FATHER_OF** and **MOTHER_OF**) you test to see if the person with that name is in the **people** tree. If not, you create the struct and insert it.

   Whenever you process a line that needs to create some links (i.e. **FATHER**, **MOTHER**, **FATHER_OF** and **MOTHER_OF**), you first check to see if the link exists and is correct. If incorrect, you flag an error. If correct, you do nothing. If the link doesn't exist, you should create it in both the parent and the child.

   When you're done with this phase, you'll have a red-black tree with all people in it, and all the people will have the correct links to their parents and children.

2. **Testing the graph for cycles.** The graph is messed up if there is a cycle in it. In other words, if a person can be his/her own ancestor or descendant, then there is a problem. Testing for cycles is a simple depth-first search, which you should have learned in CS302. (Lecture notes in are in http://web.eecs.utk.edu/~jplank/plank/classes/cs302/Notes/DFS/).

3. **Printing out the graph.** This is a topological sort, because you are not allowed to print a node until you have printed nodes that are incident to it. Topological sort is described in the CS302 lecture notes: http://web.eecs.utk.edu/~jplank/plank/classes/cs302/Notes/Topological/.

Enjoy!

There is a makefile for this lab here.