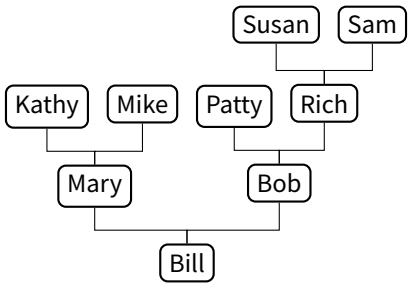
Overview

As we saw in class, defining an *imperative program* that would allow us to ask and answer questions about a family tree such as the one given below is a non-trivial task.



This is particularly true if we wish the resulting program to possess both a high degree of *clarity* and *elaboration tolerance*. This assignment is intended to build off of the family tree example discussed in lecture for the purpose of familiarizing yourself with the SWI-Prolog interpreter.

Assignment Specification

Read Chapter 4 Sections 1–5 of the textbook. Play with some of the examples to familiarize yourself with the basics of the interpreter as you go, as well as the initial family tree representation from the lecture notes. You'll recall that in class, we defined the `ancestor` relation as follows:

```
1  ancestor(X,Y) :- parent(X,Y).
2
3  ancestor(X,Y) :- parent(Z,Y), ancestor(X,Z).
```

An alternative representation that is *more natural*[1] could be as follows:

```
1  ancestor(X,Y) :- parent(X,Y).
2
3  ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

The textbook presents yet another alternative representation of the relation:

```
1  ancestor(X,Y) :- parent(X, Z), ancestor(Z, Y).
2
3  ancestor(X,X).
```

1. Compare this last representation to the two prior ones. Is is it different? If so, provide an example illustrating the difference.
2. Building off of the definitions of various familial relations discussed in the lecture, extend the program to allow us to ask queries regarding: *cousin*, *grandfather*, *grandmother*, *grandparent*, as well as *paternal/maternal* variations of grandfather and grandmother. Use the traditional understandings of the terms.

[1] based on how the the base and recursive cases fit together when read.

Submit your solution to the Course Canvas as a Prolog program (i.e., a plain text file containing your Prolog source code), with your answer to question 1 included in the comments. Note that in Prolog, single-line comments are preceded with the `%` symbol. As an example, here is one way to comment the `parent` relation discussed in lecture:

```
% parent/2 - read as X is a parent of Y
parent(bob,bill).
parent(mary,bill).
...
```

Note that in the comment describing the relation, `parent/2` gives the *name* of the relation being defined, its *arity*, and a plain reading of it in English.

Grading Criteria

Overall, this programming assignment is worth a total of 100 points according to the following breakdown:

- 75 points will awarded on the basis of your program's ability to correctly answer queries regarding the family relationships discussed in lecture and specified in this assignment.
- 25 points will be awarded based on the clarity and elaboration tolerance of your program.