Exercises

Define your answers to Exercises 1 through 6 in terms of the base relations `parent(X,Y)`, `female(X)`, and `male(X)`. To test your code, define some `parent` facts like those in this chapter, along with appropriate `female` and `male` facts. But your solutions should be general enough to work with any set of such facts.

Exercise 1 Define a `mother` predicate so that `mother(X,Y)` says that `X` is the mother of `Y`.

Exercise 2 Define a `father` predicate so that `father(X,Y)` says that `X` is the father of `Y`.

Exercise 3 Define a `sister` predicate so that `sister(X,Y)` says that `X` is a sister of `Y`. Be careful, a person cannot be her own sister.

Exercise 4 Define a `grandson` predicate so that `grandson(X,Y)` says that `X` is a grandson of `Y`.

Exercise 5 Define the `firstCousin` predicate so that `firstCousin(X,Y)` says that `X` is a first cousin of `Y`. Be careful, a person cannot be his or her own cousin, nor can a brother or sister also be a cousin.

Exercise 6 Define the `descendant` predicate so that `descendant(X,Y)` says that `X` is a descendant of `Y`.

Exercise 7 Define a `third` predicate so that `third(X,Y)` says that `Y` is the third element of the list `X`. (The predicate should fail if `X` has fewer than three elements.) *Hint:* This can be expressed as a fact.

Exercise 8 Define a `firstPair` predicate so that `firstPair(X)` succeeds if and only if `X` is a list of at least two elements, with the first element the same as the second element. *Hint:* This can be expressed as a fact.

Exercise 9 Define a `del3` predicate so that `del3(X,Y)` says that the list `Y` is the same as the list `X` but with the third element deleted. (The predicate should fail if `X` has fewer than three elements.) *Hint:* This can be expressed as a fact.

Exercise 10 Define a `dupList` predicate so that `dupList(X,Y)` says that `X` is a list and `Y` is the same list, but with each element of `X` repeated twice in a row. For example, if `X` is `[1,3,2]`, `Y` should be `[1,1,3,3,2,2]`. If `X` is `[]`, `Y` should be `[]`. Check that your predicate works in both directions—that is, check that it works on queries like `dupList(X,[1,1,3,3,2,2])` as well as on queries like `dupList([1,3,2],Y)`.

Exercise 11 Define a predicate `isDuped` so that `isDuped(Y)` succeeds if and only if `Y` is a list of the form of the lists `Y` in Exercise 10. That is, the predicate should succeed if and only if the first and second elements are equal, and the third and fourth elements are equal, and so on to the end of the list. It should fail for all odd-length lists.

Exercise 12 Define the `oddSize` predicate so that `oddSize(X)` says that `X` is a list whose length is an odd number. *Hint:* You do not need to compute the actual length, or do any integer arithmetic.

Exercise 13 Define the `evenSize` predicate so that `evenSize(X)` says that `X` is a list whose length is an even number. *Hint:* You do not need to compute the actual length, or do any integer arithmetic.

Exercise 14 Define the `prefix` predicate so that `prefix(X,Y)` says that `X` is a list that is a prefix of `Y`. That is, each element of `X` is equal to (unifies with) the corresponding element of `Y`, but `Y` may contain additional elements after that. Check that your predicate works when `X` is uninstantiated: given a query like `prefix(X,[1,2,3])`, it should find all the prefixes of the list `[1,2,3]`.

In Exercises 15 through 20, you should implement sets as lists, where each element of a set appears exactly once in its list, but in no particular order. Do not assume you can sort the lists. Do assume that input lists have no duplicate elements, and do guarantee that output lists have no duplicate elements.

Exercise 15 Define the `isMember` predicate so that `isMember(X,Y)` says that element `X` is a member of set `Y`. Do not use the predefined list predicates.

Exercise 16 Define the `isUnion` predicate so that `isUnion(X,Y,Z)` says that the union of `X` and `Y` is `Z`. Do not use the predefined list predicates. Your predicate may choose a fixed order for `Z`. If you query `isUnion([1,2],[3],Z)` it should