Name: _____

CSCI 291 — Spring 2023

Final Exam

This exam should have six pages and contain six problems, some with multiple parts. Feel free to write helper functions or relations as part of any of your solutions. Closed book and notes.

Problem 1: [13 points]

In English, describe what the Haskell function below does. What would it return when passed the input list [3,3,4,4]?

```
mystery lst = (filter (==(head lst)) lst) == lst
```

Problem 2: [15 points]

At a recent family reunion your grandfather, Leverett, asked about your classes. When you told him that Richards went on and on about Declarative Programming all semester, he says “what's so great about that — why would anyone want to express computations declaratively?” What did you tell him? (Keep in mind that Leverett's a big fan of brevity.)

Problem 3: [16 points]

The "files" on the final Prolog assignment contained information on test subjects and the sample groups they were in. Define the Prolog predicate `commonGroup(S1,S2,Pairs)`, which is true if `S1` and `S2` are subjects that appear in the same group at least once in `Pairs`, a list of `in` functors of the sort you worked with on the final Prolog assignment.

```
?- commonGroup(5, 3, [in(5,a), in(3,a)]).
true .

?- commonGroup(5, 3, []).
false.

?- commonGroup(0, 4, [in(0,a), in(2,a), in(3,a), in(7,a), in(4,b), in(5,b),
        in(6,b), in(1,b), in(0,c), in(3,c), in(4,c), in(1,c), in(5,d)]).
true .

?- commonGroup(5, S, [in(0,a), in(2,a), in(3,a), in(7,a), in(4,b), in(5,b),
        in(6,b), in(1,b), in(0,c), in(3,c), in(4,c), in(1,c), in(5,d)]).
S = 4 ;
S = 5 ;
S = 6 ;
S = 1 ;
S = 5.
```

Problem 4: [16 points]

Define the Prolog predicate `allGroups(Sub,Ps,Groups)`, which succeeds if `Groups` is a list of *all* groups from `Ps` in which subject `Sub` appears. For full credit, there shouldn't be any duplicates in `Groups`.

```
?- allGroups(5, [in(5,f), in(5,a), in(5,b)], Groups).
Groups = [a, b, f].

?- allGroups(7, [in(7,c), in(5,c), in(7,b), in(7,c), in(2,b)], Groups).
Groups = [b, c].

?- allGroups(3, [], Groups).
Groups = [] .
```

Problem 5: [16 points]

Now define a Haskell version of `commonGroup`: Write a Haskell function that takes two subjects and a list of tuples denoting group assignments, and returns `True` if the two subjects occur together in at least one group, `False` otherwise.

```
*Main> commonGroup 5 3 [(5,'a'), (3,'a')]
True

*Main> commonGroup 5 3 []
False

*Main> commonGroup 0 4 [(0,'a'), (2,'a'), (3,'a'), (7,'a'), (4,'b'), (5,'b'),
       (6,'b'), (1,'b'), (0,'c'), (3,'c'), (4,'c'), (1,'c'), (5,'d')]
True
```

Problem 6: [24 points]

The questions below refer to the following Prolog predicate. Some relevant family facts are on the next page for reference. (Feel free to detach that page.)

```
fullSister(Sis, Sib) :-
    parent(P1,Sis), parent(P1,Sib),
    parent(P2,Sis), parent(P2,Sib),
    female(Sis),
    Sis \= Sib, P1 \= P2.
```

a) Consider what would happen if we moved the female goal so that it became the *first* goal in the rule. Would the revised predicate have the same meaning? That is, would it return true for all of the same cases as the original, and generate all of the same variable bindings when `fullSister` queries included variables? For full credit, justify your answer.

b) Consider the size and shape of the search tree when running the query `fullSister(X,Y)`. If the female goal got moved as described in the previous problem, would the search tree be wider than for the original rule, narrower, or the same? (Consider the width to be the maximum number of nodes across any level of the search tree.) Justify your answer.

c) Assume we left the `female` goal where it was originally, but put a cut (`!`) immediately after the `female` goal. Would the revised predicate have the same meaning? That is, would it return true for all of the same cases as the original, and generate all of the same variable bindings when `fullSister` queries included variables? For full credit, justify your answer.

```
father(david, holly).
father(david, heather).
father(durkee, brad).
father(durkee, trevor).
father(leverett, durkee).
father(leverett, elmo).
father(brad, charlie).
father(brad, flora).
father(reuben, virginia).
father(reuben, dorothy).

mother(nancy, holly).
mother(nancy, heather).
mother(mary, brad).
mother(mary, trevor).
mother(holly, charlie).
mother(holly, flora).
mother(virginia, durkee).
mother(virginia, elmo).

male(david).
male(durkee).
male(leverett).
male(brad).
male(charlie).
male(trevor).
male(elmo).
male(reuben).

female(nancy).
female(mary).
female(virginia).
female(dorothy).
female(holly).
female(flora).
female(heather).

parent(P,Kid) :- mother(P,Kid).
parent(P,Kid) :- father(P,Kid).
```