Test 3

NAME:_____ STUDENT NO:_____

1. (20%) Given the following set of family relations `parent`, `father`, `mother`, `sibling`, `male` and `female`:

```
% parent(X,Y) if X is a parent of Y
parent(X,Y) :- mother(X,Y).
parent(X,Y) :- father(X,Y).

% sibling(X,Y) if X and Y are siblings.
sibling(X,Y) :-
     father(F,X), father(F,Y),
     mother(M,X), mother(M,Y),
     X \= Y.

female(jane).
female(lucy).
female(susan).

male(john).
male(eric).
male(bill).

father(john,susan).
father(john,eric).
father(eric,bill).

mother(lucy,jane).
mother(jane,susan).
mother(jane,eric).
```

   (a) (5%) Write a predicate `uncle(X,Y)` which is true if `X` is an uncle of `Y`.

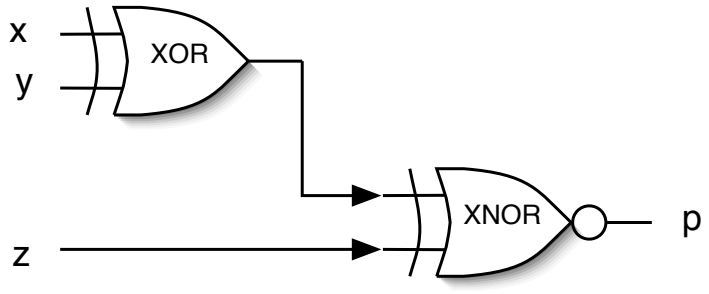
(b) (5%) Write a predicate `grandfather(X,Y)` which is true if `X` is a grandfather of `Y`.

(c) (5%) Write a predicate `immediate_family(X,Y)` which is true if `X` is an *immediate family* member of `Y` (i.e., including parents, children, brothers and sisters).

(d) (5%) Write down a query that expresses "Who are the grandchildren of lucy?"

2. (20%)

   (a) (5%) A 3-bit *odd*-parity generator can be specified as follows. That is, given three inputs `X`, `Y`, and `Z`, it produces a parity-bit `P` which gives an *odd* number of 1's when they are counted together as a binary string. (e.g., 1,1,1 gives 0; 1,0,1 gives 1.) Write a predicate `oddparity(X,Y,Z,P)` which is true if the number of 1's in



   all bits `X`, `Y`, `Z` and `P` is *odd*. You `must` use the circuit above as a template. You are given the following predicates.

```
% xor(X,Y,Z) if Z is the exclusive-or of bit X and Y
xor(0,0,0).
xor(0,1,1).
xor(1,0,1).
xor(1,1,0).

% invert(X,Y) if Y and X are inverted.
invert(1,0).
invert(0,1).
```

(b) (5%) Write down a query that expresses "If the odd-parity output is 0, and one input bit is 1, what are the other two inputs?"

(c) (5%) Explain how you could use the `oddparity` predicate to *check* for *odd-parity*, i.e., instead of a parity generator?

(d) (5%) Could you suggest another definition of `oddparity` without following the circuit diagram?

3. (20%) Given the following predicate "`concat(X,Y,Z)`" which holds if `Z` is the list concatenation of lists `X` and `Y`.

```
concat( [], Y, Y ).
concat( [H|X], Y, [H|Z] ) :- concat( X, Y, Z ).
```

Use this `concat` predicate for the following questions.

(a) (6%) Write a predicate `suffix( S, L )` which is true if the list `S` is a suffix of the list `L`, e.g., `suffix( [3,4], [1,2,3,4] )`, and `suffix( [4], [1,2,3,4] )` are true.

(b) (6%) Write a predicate `thrice( L, L3 )` which is true if the list `L` occurs in the list `L3` three times consecutively, e.g., `thrice( [1], [1,1,1] )`, and `thrice( [1,2], [1,2,1,2,1,2] )` are true.

(c) (8%) Write a predicate `adjacent( X, Y, L )` which is true if two elements `X` and `Y` are adjacent in the list `L`, e.g., `adjacent( 3,4,[5,1,3,4,2] )` and `adjacent( 4,3,[5,1,3,4,2] )` are true.

4. (20%)

   (a) (15%) Write a Prolog program that solves the following crypto-arthmetic puzzle. That is, all letters must denote a different digit from 0 to 9, and when they are added together will produce a *correct* answer.

```
          E   A   T
+     T   H   A   T
----------------------
  A   P   P   L   E
```

   (b) (5%) Using this problem as an example, what does *non-determinism* mean?

5. (20%)

   (a) (5%) What is a *logical (deduction) inference* step? Explain. You may use the family database in Question 1 as an example.

   (b) (5%) What is *unification*? Explain.

(c) (5%) Use the Computer Science Degree Requirements example (i.e., based on propositional logic). Explain why it is *not* adequate to express the requirements for many students, e.g., john, mary, jane, etc. Show how you could modify it to address all students.

```
csmajor :- csreq, mathreq, english, electives.
csreq :- csc1, csc2, csc3, csc4.
csc1 :- csc110, csc115, csc212.
csc2 :- csc230, csc225, seng265.
...
```

(d) (5%) How does Prolog *search* for an answer? Explain. Again, you may use Question 1 as an example.