**CS 81, Logic and Computability**
**Problem Set 3: First-Order Predicate Logic and Prolog**

Please watch the Prolog video lecture before embarking on the Prolog part of this assignment. Online Prolog interpreters such as SWISH (https://swish.swi-prolog.org/) are available. Some students prefer those environments to running `swipl` locally.

### Challenge 1: Translating from English to Logic with Moose and Aardvarks [20 Points]

This problem is sponsored by the International Society of Moose and Aardvarks Owners (ISMAO). Let $L$ denote a binary predicate representing the *Likes* relation (so that $L(x, y)$ means that $x$ likes $y$) on a domain of people. Let $A$ be a unary predicate denoting whether its argument owns an aardvark and let $M$ be a unary predicate indicating whether its argument owns a moose. Express each of the following English statements using these three predicates and the operators of first-order predicate logic. **Keep your expressions as simple as possible and push your quantifiers as far to the left as possible.**

1. Everyone likes some person who owns an aardvark and a moose.
2. There is someone who owns a moose who likes every aardvark owner.
3. For every pair of people who don't own aardvarks, if the first likes the second then the second likes the first.
4. For every pair of people who own different animals (one a moose only, the other an aardvark only) those people either both like one another or both dislike one another.
5. For every three moose owners, if the first likes the second and the second likes the third then the first likes the third. (Transitive liking of moose owners!)

## Answer 1

1. $\forall x \exists y (L(x, y) \rightarrow (A(y) \wedge M(y)))$
2. $\forall x \exists y ((M(y) \wedge A(x)) \rightarrow L(y, x))$
3. $\forall x, y ((\neg A(x) \wedge \neg A(y)) \rightarrow L(x, y) \rightarrow L(y, x))$

4. $\forall x, y(((A(x) \wedge A(y))) \vee (M(x) \wedge M(y))) \rightarrow ((L(x,y) \leftrightarrow L(y,x))) \vee (\neg(L(x,y)) \leftrightarrow \neg L(y,x)))$
5. $\forall x, y, z((M(x) \wedge M(y) \wedge M(z) \wedge L(x,y) \wedge L(y,z)) \implies L(x,z))$

**Challenge 2: Translating from Logic to English [10 Points]**

Translate each of the following predicate logic statements into English with the same meaning for predicates $L$, $A$, and $M$ as in the previous problem. Keep your English statements as simple and concise as possible.

1. $\forall x \exists y\ (L(x,y) \wedge \neg L(y,x))$
2. $\exists x\ (M(x) \wedge \forall y\ (A(y) \rightarrow \neg L(x,y)))$
3. $\forall x \forall y\ (L(x,y) \rightarrow \forall z(L(x,z) \rightarrow L(y,z)))$
4. $\forall x \forall y\ (L(x,y) \rightarrow \exists z(L(x,z) \rightarrow L(y,z)))$
5. $\exists x\ (M(x) \rightarrow \forall y\ M(y))$

## Answer 2

1. Everyone likes someone who does not like them back.
2. There is some mooose owner who dislikes all ardvark owners.
3. For all pairs of people if the first likes the second means that the first likes all other people, then the second person also likes everyone that the first likes.
4. For all pairs of individuals, if the first likes the second, then there exists some individual that if the first person likes, then the second also likes that person.
5. If there is someone who owns a moose, then everyone owns a moose.

**Challenge 3: Validity! [15 Points]**

For each of the following statements, explain in one sentence why it is true.

1. $\neg \forall x\ P(x) \rightarrow \exists x\ \neg P(x)$
2. $\neg \exists x\ P(x) \rightarrow \forall x\ \neg P(x)$
3. $(\exists x\ A(x)) \rightarrow ((\forall x\ M(x)) \rightarrow (\exists x\ M(x)))$

## Answer

1. If we cannot say that for all $x$ that $P(x)$ holds, then there must be some element in our domain for which $P(x)$ does not hold, and thus, that there is some element for which $\neg P(x)$ holds.

2. If there does not exist a single element for which $P(x)$ holds, then it must be true that $\neg P(x)$ holds for all elements since $P(x)$ either takes on false or true.

3. If our first predicate is true, meaning that we can find someone who owns an ardvark, then we get to know that everyone owns a moose, which naturally implies that we can come up with someone who owns a moose (choose someone from our domain which we know is non-empty and that everyone owns a moose).

### Challenge 4: A Matter of Interpretation! [15 Points]

For each of the statements below, describe two interpretations: One in which the statement is true and the other for which it is not true. If possible, the domain that you describe should be finite and as small as possible. State the domain (a set) and interpretation of each predicate (the elements in the domain for which that predicate is true) explicitly and then explain clearly why the statement is either true or not true in the given interpretation (remember, you'll do two interpretations for each of these three subproblems):

1. $(\forall x\ A(x)) \vee \neg\exists x\ M(x)$

2. $(\exists x\ A(x)) \rightarrow (\forall y\ A(y))$

3. $((\forall x\ A(x)) \rightarrow (\forall x\ M(x))) \rightarrow \forall x\ (A(x) \rightarrow M(x))$

## Answer

1. For the true case, consider the set $\{x\}$ with the property that $A(x)$ holds, which means that the whole statement holds since it is an or statement. Now, for the false case, consider the same set, except for if $A(x)$ does not hold, but that $M(x)$ does hold. Then we can see that neither of our parts of the or statement hold, and thus, the whole statement does not hold.

2. For the true case, consider the set $\{x\}$ with the property that $A(x)$ holds and that then, since $x$ is the only element of the set, $A(y)$ holds for elements in the set. However, for the false case, consider the set $\{x, y\}$ such that $A(x)$ is true but $A(y)$ is false. Then we can see the whole expression does not hold and evaluates to false since $1 \to 0$ evaluates to false.

3. For the true case, consdier the set $\{x\}$ with the property that $A(x), M(x)$ both hold. For the false case, consider the set $\{x, y\}$ with the property that $A(x)$ is true, $A(y)$ is false, $M(x)$ is false and $M(y)$ is true. We can see then that the first part of our expression will evaluate to true since $0 \implies 0$, but then that our second part of the statement will be false since we can see clearly that the implication does not hold for all elements in our set, and thus, our statement evaluates to false overall.

**Challenge 5: A First Taste Prolog! [20 Points]**

Write a predicate called `count(X, L, N)` that is true if and only if item `X` occurs exactly `N` times in list `L`. For example:

```
count(spam, [oh, spam, spam, we, love, spam], N).  % user query
N = 3 ;    % prolog replies.  We type ; to ask if there's any more
false      % nope, no  more!

count(X, [oh, spam, spam, we, love, spam], N).  % user query
X = oh,
N = 1 ;    % any more?
X = spam,
N = 3 ;    % any more?
X = we,
N = 1 ;    % any more?
X = love,
N = 1 ;    % any more?
N = 0.
```

There's a good chance that your code will report more options, including that "spam" occurs 3 times but also twice and also once. That's an error and it's likely due to the order in which things appear. This is a good opportunity to think carefully about why order matters. By the way, the "N = 0" is fine, but it's OK if your code also doesn't produce this.

Use the starter file `count.pl` to write your code, and do not change the filename when you upload to Gradescope.

### Challenge 6: Prolog in Depth! [20 Points]

Write a predicate called `depth(E, Tree, N)` that is true if and only if element `E` is at depth `N` in the given binary search tree `Tree` in `[Root, Left, Right]` format. Here's more explanation: A binary search tree can be represented as either an empty list, `[]`, or a list of the form `[Root, Left, Right]` where `Root` is a number and `Left` and `Right` are themselves binary search trees in which all of the items in `Left` are less than or equal to `Root` and all items in `Right` are greater than `Root`. For example, here's a predicate that defines a small binary search tree represented in this way:

```
tree([42, [5, [], []], [47, [],[50, [], []]]]).
```

The depth of an item in the tree is 0 if its at the root, 1 if its parent is the root, 2 if its parent's parent is at the root, and so forth. Here are some examples using the tree defined above:

```
?- tree(T), depth(42, T, N).
T = [42, [5, [], []], [47, [], [50, [], []]]],
N = 0 ; % any more?
false.

?- tree(T), depth(50, T, N).
T = [42, [5, [], []], [47, [], [50, [], []]]],
N = 2 ; % any more?
false.
```

You may assume that we'll only test the `depth` predicate on items that are in the tree and that each item in the tree only appears once.

Use the starter file `depth.pl` to write your code, and do not change the filename when you upload to Gradescope.