COM1310 Assignment #4

As for the previous assignment, there are two drop boxes for this one as well: one for the Python code for problem 1 and one for the math solutions from the other problems.

1. This assignment builds upon the code from Assignment 3. Before you begin, clean up any bugs identified by your professor in your `satisfy.py` – they will plague you if you try to approach this assignment first. (Alternatively, you may start this assignment with the model code provided by Professor Kelly after the close of Assignment 3, but this is recommended only if you think you understand that code clearly.) The goal of this assignment is to evaluate the relationships between **two different** propositional logical expressions by comparing them across all possible environments.

   Requirements:

   a. Your program shall be contained in a single file called `implications.py`.

   b. Your program shall be invoked to process a data file the same way as `satisfy.py`

   c. Each file of data will contain 2 different logical expressions on the first two lines of the file. Each expression is self-contained on one line. We will refer to these expressions in these requirements as LEFT and RIGHT, where LEFT is found on the first line of the file and RIGHT is on the second. There will be no data on any subsequent lines of the file. (You are not required to follow these names in your code, but you may.)

   d. The Boolean variables contained in LEFT and RIGHT, respectively, will often overlap, though they are not required to do so. The set of variables that comprise an environment shall be the **union** of the variables from LEFT and RIGHT.

   e. Two specific variables that might appear in the logical formulas are not really variables, but constants. The constant T is always true (i.e., truthy), and the constant F is always false (i.e., falsey). All other capital letters represent normal Boolean variables that may be either true or false.

   f. If LEFT and RIGHT are logically equivalent, your program shall print "EQUIVALENT" and then exit.

   g. If LEFT and RIGHT are **not** logically equivalent but LEFT implies RIGHT, your program shall print "LEFT implies RIGHT" and then exit.

   h. Similarly, If LEFT and RIGHT are **not** logically equivalent but RIGHT implies LEFT, your program shall print "RIGHT implies LEFT" and then exit.

   i. If neither expression implies the other, your program shall print "NO IMPLICATION" and then exit.

   You could approach this problem in at least two equally reasonable ways. First, you could check the truth values of LEFT and RIGHT in each environment and check directly if these contradict the possibilities that LEFT implies RIGHT or RIGHT implies LEFT. Alternatively, you could construct two logical expressions that directly express the ideas "LEFT implies RIGHT" and "RIGHT implies LEFT" (using and, or, and not) and check whether those were *valid* expressions, i.e., true in all environments. Or you could try something else. Propositional logic is relatively simple, compared with predicate logic.

2. A popular type of logic puzzle is “Knights and Knaves”. These puzzles have been around since at least the 1930s and appear as plot points in movies and TV shows. They concern a situation where each person is either a “knight” who always tells the truth or a “knave” who always lies. (Some problems may explicitly allow “spies” who may either lie or tell the truth.) The goal is to determine who is a knight and who is a knave based on what they say.

   In most knight/knave puzzles, it is possible to determine the type of each person unambiguously; that is the desired outcome. But there are two other possibilities:

   a. Multiple solutions are possible due to insufficient information
   b. No solution is possible because the information is self-contradictory

   An example of the first kind would be a solitary person who states, “I am a knight”; that statement by itself is consistent with both a truth-telling knight and a lying knave. An example of the second kind would be a person who says, “I am a knave”. Supposing that person is a knight, then the statement would not be accurate, so the person is lying, so he cannot be a knight. Supposing that person is a knave, then the statement would be true, so that person is telling the truth, so he cannot be a knave either! As there are no other choices, there is no solution. (If spies were allowed in the second problem, the person could be a spy.)

   The most methodical way to solve knight/knave problems is with ***proof by cases*** where you propose a knight or knave identity for each person in each environment. Then each case proceeds as a ***proof by contradiction***. If a contradiction cannot be found, then you *may* have found a possible solution. The end goal is to show that exactly one *environment* leads to a solution.

   For each of the knight/knave problems below, determine if the problem has a single unique solution and **show your proof steps**. Be sure that you are not just eliminating impossible environments; you also need to constructively show that one environment actually works in order to claim a solution.

   a. [Easy:] On an island of knights and knaves, you are approached by two people, A and B. A says to you, "we are the same kind of people." B says, “We are different kinds of people” What are they actually?
   b. [Moderate:] On an island of knights and knaves, you are approached by three people: Alice, Bob, and Celia, and Don. Alice says: “Bob is a knave and Celia is a knight”. Bob says: “If Celia is a knight, then Alice is a knight.” Celia says: “Neither Alice nor Bob are knights.” Which ones are the knights and the knaves?
   c. [Hard, 5 points extra credit only:] On an island of knights and knaves **and spies**, you have a meeting with three people. One wears blue, one wears red, and one wears green. You are **warned in advance** that one is a knight, one is a knave, and one is a spy.

      "Who is the spy?" you ask. The man wearing blue says, "I am a knight." The man wearing red says, "He speaks the truth." The man wearing green says, "I am a spy." Who is really the spy? Who is the knight and who is the knave?

3. A surprise #1 hit song in late 1964 was Dean Martin’s “Everybody Loves Somebody Sometime”. It knocked the Beatles out of the #1 spot on the Top 40 charts for a few weeks. For context, listen to the song at https://www.youtube.com/watch?v=z-2_OstpR5c. [Professor Kelly says “the kids on my school bus groaned and fake-barfed whenever it came on the radio. So sappy!”]. The title line of that song contains *three* English quantifiers, and we know that English quantifiers don’t always translate 1-for-1 into logic. Consider the following logic formulas. Translate each formula into a

similar English song title (maybe an awkward one) and further state which formula is the most reasonable interpretation of Dean Martin's song title, presuming the predicate *Loves(p, q, t)* means that p loves q at time instant t.

a. $\forall p \exists t \exists q . Loves(p, q, t)$
b. $\exists p \exists q \exists t . Loves(p, q, t)$
c. $\exists p \forall t \forall q . Loves(p, q, t)$
d. $\forall p \exists q \forall t . Loves(p, q, t)$

You will get no credit for any song title that sound "mathy", e.g., that includes phrases like "for all" or "there exists" or that refers to logical variables. The words/phrases "everybody", "nobody", "somebody", "anybody", "loves", "doesn't love", "sometime", "any time", "always", "ever", "never", etc. are likely to be handy. The point is to explore the translation between logic and English.

4. How do you negate a *term* in predicate logic? (A "term" refers to something like the four examples in the previous problem: a single predicate preceded by a bunch of quantifiers.) It is not enough to negate the predicate itself. For example, the opposite of "Everybody loves somebody" is **not** "everybody doesn't love somebody" – there's nothing inconsistent with loving somebody but not loving somebody else. To negate a term in predicate logic, you have to negate the predicate **and** "flip" all the quantifiers so that "for all" becomes "there exists" and vice-versa. So, one translation of the negation of "everybody loves somebody" is "somebody doesn't love anybody". Those two are clearly inconsistent with each other.

   Create the logical negations of examples c. and d. in the previous problem and then translate them into English.