In a Knights and Knaves puzzle, the following information is given: Each character is either a knight or a knave. A knight will always tell the truth: if knight states a sentence, then that sentence is true. Conversely, a knave will always lie: if a knave states a sentence, then that sentence is false.

The objective of the puzzle is, given a set of sentences spoken by each of the characters, determine, for each character, whether that character is a knight or a knave.

For example, consider a simple puzzle with just a single character named A. A says “I am both a knight and a knave.”

Logically, we might reason that if A were a knight, then that sentence would have to be true. But we know that the sentence cannot possibly be true, because A cannot be both a knight and a knave – we know that each character is either a knight or a knave, but not both. So, we could conclude, A must be a knave.

That puzzle was on the simpler side. With more characters and more sentences, the puzzles can get trickier! Your task in this problem is to determine how to represent these puzzles using propositional logic, such that an AI running a model-checking algorithm could solve these puzzles for us.

**Understanding**

take a look at puzzle.py. At the top, we’ve defined six propositional symbols. AKnight, for example, represents the sentence that “A is a knight,” while AKnave represents the sentence that “A is a knave.” We’ve similarly defined propositional symbols for characters B and C as well.

What follows are four different knowledge bases, knowledge0, knowledge1, knowledge2, and knowledge3, which will contain the knowledge needed to deduce the solutions to the upcoming Puzzles 0, 1, 2, and 3, respectively. Notice that, for now, each of these knowledge bases is empty. That’s where you come in!

The main function of this puzzle.py loops over all puzzles, and uses model checking to compute, given the knowledge for that puzzle, whether each character is a knight or a knave, printing out any conclusions that the model checking algorithm is able to make.

**Specification**

Add knowledge to knowledge bases knowledge0, knowledge1, knowledge2, and knowledge3 to solve the following puzzles.

* Puzzle 0 is the puzzle from the Background. It contains a single character, A.
  + A says “I am both a knight and a knave.”
* Puzzle 1 has two characters: A and B.
  + A says “We are both knaves.”
  + B says nothing.
* Puzzle 2 has two characters: A and B.
  + A says “We are the same kind.”
  + B says “We are of different kinds.”
* Puzzle 3 has three characters: A, B, and C.
  + A says either “I am a knight.” or “I am a knave.”, but you don’t know which.
  + B says “A said ‘I am a knave.’”
  + B then says “C is a knave.”
  + C says “A is a knight.”

In each of the above puzzles, each character is either a knight or a knave. Every sentence spoken by a knight is true, and every sentence spoken by a knave is false.

Once you’ve completed the knowledge base for a problem, you should be able to run python puzzle.py to see the solution to the puzzle.

**Hints**

* For each knowledge base, you’ll likely want to encode two different types of information: (1) information about the structure of the problem itself (i.e., information given in the definition of a Knight and Knave puzzle), and (2) information about what the characters actually said.
* Consider what it means if a sentence is spoken by a character. Under what conditions is that sentence true? Under what conditions is that sentence false? How can you express that as a logical sentence?
* There are multiple possible knowledge bases for each puzzle that will compute the correct result. You should attempt to choose a knowledge base that offers the most direct translation of the information in the puzzle, rather than performing logical reasoning on your own. You should also consider what the most concise representation of the information in the puzzle would be.
  + For instance, for Puzzle 0, setting knowledge0 = AKnave would result in correct output, since through our own reasoning we know A must be a knave. But doing so would be against the spirit of this problem: the goal is to have your AI do the reasoning for you.