We pulled code from the AI online repository. We used the Eclipse IDE in order to program. The files we worked with included aima.core.logic.propositional.kb.KnowledgeBase.java and the file we created, WumpusWorld.java. The main classes we worked with included *knowledgebase, aSentence* and *queryString*. There were three ways we could have implemented this solution: table lookup, forward chaining, and resolution. We chose table lookup because we believed it was the easiest solution. Forward chaining would have required our statements be in horn form, and resolution would have required that our statements be in CNF form. We were dealing with a small number of symbols for each query, so table-lookup was fine. We did not need to convert to any standard form in order to perform an algorithm that would have saved computation time. We would use the *tell* function to add statements to the knowledge base and then use a function we made called *ttEntails* in order to see if a statement is true or not based on the knowledge base.

“ttEntails” takes in the KnowledgeBase and a series of strings, and appends them to a Stringbuilder. For each string, it then uses the knowledgebase function “askwithTTEntails” whether the string (now a queryString) is valid or not. From there, we print out whether each statement is valid or invalid. “askwithTTEntails” uses the TTEntails algorithm to see if a queryString entails the knowledge base or not.

From there, for each problem, we used the *tell* function of the knowledge base to add to create the knowledge base for each individual problem. We gave as few rules to the knowledge base as possible for simplicity. Because we were constrained by the use of propositional logic, and not first-order logic, we had to make each rule specific to each square or combination of squares, and not have more general rules that can apply to each square individually.

For the 7.3a part of the assignment, we told it that a breeze in square 1,1 biconditionally implies a pit in 1,2. We then told it that if 1,1 is ok, that biconditionally implies no pit in 1,1. We then did the same for squares 1,2 and 2,1. Finally, we told it that 1,1 was ok and that there was no breeze. We then asked it if 1,2 and 2,1 were safe. Using this information, the program found that no breeze in 1,1 implied that there was no pit in 1,2 or 2,1. It then implied that if there was no pit in 1,2 or 2,1 that those squares are safe.

For the 7.3b part of the assignment, we told the knowledge base that a breeze in 2,1 biconditionally implies a pit in 11, 31, or 22. We then told the knowledge base that if 11, 12, and 21 are safe, then there is no pit in them. Finally, we told the knowledge base that there was a breeze in 21, and that 11 is safe. From there, it determined that a pit in 3,1 or 2,2 could be possible, but is not necessarily true.

For the 7.4a part of the assignment, we told the knowledge base that a stench in 12 biconditionally implied that there is a wumpus in 11, 22, or 13. We told the knowledge base that is there is a breeze in 21, there is a pit in 11, 31, or 22. We then told the knowledge base that an ok in 11, 12, 21, or 22 biconditionally implies that there is no wumpus and no pit in the respective squares. Finally, we told the knowledge base that 12 had a stench, 11, 12, 21, 22 were ok, and that there was a breeze in 21. Finally, we asked the knowledge base if a wumpus in 13, ok in 22, or a pit in 31 were entailed. It then found that there was a wumpus in 13 because there was a stench in 12, but 22 was safe. It found that 22 was safe because we told it so. Finally, it a pit in 31 entailed the knowledge base, because there was a breeze in 21, but 11 and 22 were ok.