CSCI 561: Foundations of Artificial Intelligence
Instructor: Sheila Tejada
Homework #2: Inference in First Order Logic
Due Date: June 30, 11:59 pm

Introduction:

To check the spread of the infectious disease Ebola, the National Center for Epidemic Control (NCEC) is preparing a large task force in identifying early symptoms of the disease. After a series of meetings with the experts, the NCEC has concluded that a practical way to set forth would be to train the task force in using a medical knowledge base for early identification of symptoms and infected people. This would allow expert knowledge to be widely available through the knowledge base and minimize human error.

You have been hired by the NCEC to facilitate the training of the task force in using the knowledge base. Each trainee is presented with disease related information in form of first order logic clauses. The trainee can read the clauses in the knowledge base and provide a logical conclusion that is provable from the clauses. You are required to develop a program that can check the conclusion made by the trainee and provide a quick feedback about whether the conclusion is right or not. The trainee can thus be trained to make conclusions using the knowledge base.

Problem:

You are given a knowledge base and one query. Your job is to determine if the query can be inferred from the knowledge base or not and report all the constants that satisfy the query if the query has a variable. You have to use backward chaining algorithm (AIMA 3rd edn. Figure 9.6) to solve this problem.

Input format:

You will be given an input file. Read the input file name from the command line. The first line of the input file will be the query. Next line of the input file will contain the number of clauses in the knowledge base (m). Following, there will be m lines each containing a statement in the knowledge base. Each clause is in one of these two formats:

- *an implication of the form p1 ∧ p2 ∧... ∧ pn => q, whose premise is a conjunction of atomic sentences and whose conclusion is a single atomic sentence*
- *a fact which is an atomic sentence: q*

Each atomic sentence is a predicate applied to a certain number of arguments. Note that negation is not used in this homework.

Sample:

```
Works(x,AidAgency)
7
Works(x,AidAgency)&HasTraveled(x,Timbuktu)=>Diagnosis(x,Infected)
Diagnosis(x,Fever)=>HasSymptom(John,Fever)
HasSymptom(x,Fever)&HasTraveled(x,Timbuktu)=>Works(x,AidAgency)
Works(Mary,AidAgency)
HasTraveled(Mary,Timbuktu)
HasTraveled(John,Timbuktu)
HasSymptom(Alice,Fever)
```

Notes:

- Variables are denoted by a single lowercase letter (For this homework, you can assume that only variable 'x' will be used. No other variables are used.)
- All predicates (such as Works) and constants (such as Fever) are case-sensitive alphabetical strings that begin with uppercase letters.
- Each predicate has at least one and at most two arguments. Same predicate will not appear with different number of arguments.
- All of the arguments of the facts are constants, i.e. you can assume that there will be no fact such as HasSymptom(x,Fever) (which says that everyone has fever!) in the knowledge base.
- You can assume that the input format is exactly as it is described. There are no errors in the given input.
- The knowledge base that you get is consistent. So there are no contradicting rules or facts in the knowledge base.
- There won't be any loop in the knowledge base.
- There will be at most 100 clauses in the knowledge base.
- Each implication clause will have only one implication in it.

Output format:

You have to create a file named **"output.txt"**. In the output file, you will log your backward chaining process as shown in the sample output.

Sample:

```
Query: Works(x,AidAgency)
Query: HasSymptom(x,Fever)&HasTraveled(x,Timbuktu)=>Works(x,AidAgency)
Query: HasSymptom(x,Fever)
Query: Diagnosis(x,Fever)=>HasSymptom(John,Fever)
Query: Diagnosis(x,Fever)
Diagnosis(x,Fever): False
HasSymptom(x,Fever): True: ['Alice']
Query: HasTraveled(x,Timbuktu)
HasTraveled(x,Timbuktu): True: ['John', 'Mary']
Works(x,AidAgency): True: ['Mary']
```

Output file starts with logging the query in the input file in the given format. Each subsequent query that you make in order to determine whether the parent query can be satisfied, will be logged in the same format (e.g. Query: HasSymptom(x,Fever)). Once you have evaluated a query, you will log whether the query can be satisfied or not by printing the query, the truth value and the list of constants that satisfies the query, each separated by ':' as shown above (e.g. HasTraveled(x,Timbuktu): True: ['John', 'Mary']). If the query is not satisfied then no constants are logged (e.g. Diagnosis(x,Fever): False). The evaluation order the left hand side of a clause is from left to right. For e.g., for a clause x&y=>z, you will first log the evaluation of 'x' and then the evaluation of 'y'. The order of checking the knowledge base for satisfying a query is top to bottom. For e.g., to evaluation the query HasSymptom(x,Fever), you will first evaluate the clause Diagnosis(x,Fever)=>HasSymptom(John,Fever) and then evaluate the clause HasSymptom(Alice,Fever). The constants that satisfies a query are listed in alphabetical order (e.g. ['John', 'Mary']).

Make sure you follow the spacing requirements as given in the sample outputs. We will be using 'diff' to match your output with the standard output.

Guidelines:

- You can use Python2.7 or Python3.4 to implement your code. Make sure that the code runs on Vocareum.
- The name of the code should be **"inference.py"** for Python 2.7 and **"inference3.py"** for Python3.4.
- The command to run your code would be "python inference{3}.py *inputfile*", where you will read the input file name from the command line argument.
- Input file is a text file ending with ".txt" extension.
- **Late submissions:** Late submissions will be allowed for additional 24 hours, i.e. till **July 1, 11:59 pm**. The late submission **penalty will be 50%** of the score you get. Homework will not be accepted after July 1.
- Homework-2 is worth 10% of the total grade for the course.
- Multiple submissions are allowed, and your last submission will be graded. So you are encouraged to submit early and often in order to iron out any problems, especially issues with the format of the final output. The performance of you program will be measured automatically; failure to format your output correctly may result in very low scores, which will not be changed.
- This is an individual assignment. You may not work in teams or collaborate with other students. You must be the sole author of 100% of the code you turn in.
- You may not look for solutions on the web, or use code you find online or anywhere else.
- You may use external resources to learn basic functions of Python (such as reading and writing files, handling text strings, and basic math), but the computation performed by the program must be your own work.
- Failure to follow the above rules is considered a violation of academic integrity, and is grounds for failure of the assignment, or in serious cases failure of the course.