CZ3005 Artificial Intelligence - Assignment 3

Implementing a Talking Box with Prolog

Learning Objective:	To reinforce the understanding on the use of Prolog as a tool of logic programming and the specification of declarative knowledge – semantic information.
Learning Outcome:	a) Understand the separation of Knowledge from the inference mechanism in Knowledge-Based System (KBS). b) Appreciate how the KBS can be updated based on interactive responses from the users.
Tools:	SWI-Prolog - http://www.swi-prolog.org/
Learning Activities:	1. Download the sample Prolog program "**TalkingBoxPrologOnly.pl**" and familiarize with how to interactively query a KBS. You can extend the code from this file to produce your solution. 2. Watch the briefing video on NTULearn: a. "**Assignment 3 Briefing.mp4**"
Deliverables:	This assignment consists of 4 problems. You will only need to do ONE of them. You are free to choose which of the 4 questions you want to work on. The design, creativity, quality of your report, and how well documented your code is will be taken into account when marking.
Submission Instruction:	Your submission should contain: 1) The Prolog file for the talking box named as: "<your_name>_qn_<number>.pl", and 2) A brief report in PDF format "<your_name>_Assignment3.pdf" to include: a. Discussions on how you designed your solution, b. Screenshot of your interaction with the Prolog program you have written. You should include your name, tutorial group number (e.g., TSR1) and matriculation card number in the cover page of your report. You should clearly state in your report which question you chose. The submission is through NTULearn. The deadline for submission is one week after your respective lab session.

[Hints]

How to run the "**TalkingBoxPrologOnly.pl**" program:

1. Double click on the file to open it in the Prolog command line interface.
2. As the code uses the assert() function to TELL the KBS what the user likes and dislikes in order to dynamically select subsequent questions to ask, you need to use the "dynamic" command to let Prolog allow such assertions to modify the KBS on the fly:

```
?- dynamic(like/1).
true.

?- dynamic(dislike/1).
true.
```

3. Then, start interacting with the KBS with the "ask(0)." command and follow the instructions on the screen (y – "yes", n – "no", q – "quit").

```
?- ask(0).
"Hey... Do you like "trekking'? y/n/q: 'y.
"Great... Do you like "picnic'? y/n/q: '|: n.
"Huh... May be you like "night'? y/n/q: '|: y.
"Great... Do you like "gifts'? y/n/q: '|: y.
"Great... Do you like "wine'? y/n/q: '|: y.
"Great... Do you like "dinner'? y/n/q: '|: y.
"Great... Do you like "candlelight'? y/n/q: '|: y.
"Great... Do you like "rains'? y/n/q: '|: y.
"Great... Do you like "tea'? y/n/q: '|: y.
"Great... Do you like "concert'? y/n/q: '|: y.
"Great... Do you like "poetry'? y/n/q: '|: y.
"Great... Do you like "music'? y/n/q: '|: n.
"Huh... May be you like "sleeping'? y/n/q: '|: n.
"Huh... May be you like "flowers'? y/n/q: '|: y.
"Great... Do you like "coffee'? y/n/q: '|: y.
"Great... Do you like "friends'? y/n/q: '|: y.
"Great... Do you like "party'? y/n/q: '|: y.
"Great... Do you like "beer'? y/n/q: '|: y.
"Great... Do you like "event'? y/n/q: '|: n.
"Huh... May be you like "movie'? y/n/q: '|: n.
"Huh... May be you like "action_movies'? y/n/q: '|: n.
"Huh... May be you like "soccer'? y/n/q: '|: n.
"Huh... May be you like "gardens'? y/n/q: '|: n.
"Huh... May be you like "water'? y/n/q: '|: n.
"Huh... May be you like "roses'? y/n/q: '|: y.
"Great... Do you like "dating'? y/n/q: '|: q.
```

4. To see what new likes and dislikes the KBS has learned through the interactions, you can use the following instructions:

```
?- like(X).
X = nothing ;
X = trekking ;
X = night ;
X = gifts ;
X = wine ;
X = dinner ;
X = candlelight ;
X = rains ;
X = tea ;
X = concert ;
X = poetry ;
X = flowers ;
X = coffee ;
X = friends ;
X = party ;
X = beer ;
X = roses.
```

```
?- dislike(X).
X = nothing ;
X = picnic ;
X = music ;
X = sleeping ;
X = event ;
X = movie ;
X = action_movies ;
X = soccer ;
X = gardens ;
X = water.
```

[Choose only ONE of the following Four questions to answer]

Question 1: Ten Questions

In the two-people game of ten questions, one person is answerer and the other person is questioner. They decide one topic to confine the scope, say Olympics games. The answerer decides an Olympic game in his/her mind and does not reveal the game to the questioner. Questioner is supposed to ask a maximum of ten questions to guess the answer.

Design a Prolog script that plays the role of an answerer. For example, queries can be related to:

- Team size (2 for badminton doubles, 1 for swimming)
- Number of teams in a game (2 for badminton doubles, many for swimming)
- Arena type (court for badminton doubles, pool for swimming)
- Play device (shuttle cock or racket for badminton doubles, water for swimming, ball or racket for tennis, javelin for javelin throw, etc.)
- Game mode (knock out, timed)
- Performance type (score, win)

There should be only one form of query, for example: has(ball), has(pool), has(team), has(teamsize,2), has (teams_per_game, 2), etc. And one form of decision: is(badminton).

Prolog script should be able to play the game 5 times, choosing a different game each time.

(25 marks)

Question 2: Kid's Day at School

Assume that the Prolog script is a parent, trying to know about a kid's day at school. The Prolog script should converse intelligently with the kid as follows. The Prolog script should ask a question to the kid that kid can answer yes or no. Depending on whether the answer is yes or no, Prolog script should ask a related question or another random question. For example, if the kid says yes to whether it ate or not, the query can be a food item, or whether the kid used fork or spoon, or whether the kid washed hand. Similarly, related to games.

(25 marks)

Question 3: Subway Sandwich Interactor

The Prolog script offers different meal options, sandwich options, meat options, salad options, sauce options, top-up options, sides options etc. to create a customized list of person's choice. The options should be intelligently selected based on previous choices. For example, if the person chose a veggie meal, meat options should not be offered. If a person chose healthy meal, fatty sauces should not be offered. If a person chose vegan meal, cheese top-up should not be offered. If a person chose value meal, no top-up should be offered.

(25 marks)

Question 4: Patient with a Sympathetic Doctor

Assume that the Prolog script is a sympathetic doctor, conversing with a patient who can answer only yes or no. The doctor should be able to diagnose the patient's condition while asking question sensitively depending upon patient's pain level and mood level. You can choose 5 or more different mood considerations (calm, angry, etc.) and 5 or more levels of pain. Five or more diseases should be diagnosable, each disease characterized by 5 or more symptoms.

(25 marks)

- End of Assignment 4 -