NANYANG TECHNOLOGICAL UNIVERSITY

CZ3005 ARTIFICIAL INTELLIGENCE

Lab 2 REPORT

HENG WEILIANG

U1620593L

TS3

Sympathetic Doctor

Objective:

To create a Prolog based program that assume the role of a sympathetic doctor. By asking a series of questions that is from the knowledge base. The doctor would be able to give a diagnosis of what the patient may have or inform the patient that the knowledge base is not enough to give an diagnosis.

Overview:

The assignment has been done with Prolog together with Python and QT. The program will include a CLI base interactive script as well as a Python GUI program.
The CLI base script `doctorCLI.pl` can be initial by typing `main/0` in the prolog console. The patient should answer `yes` and `no` only.
The GUI program can be started by loading the `App.py` depending on the operating system.

How it works:

This program is basically a linear search following by a matching with predefined result included in the knowledge.

1. Creating a list of facts about symptoms in the knowledge base. Such as pain, mood etc.
2. Creating a list of facts for matching the symptoms with possible diseases.
3. Creating predicates to ask the user about the symptoms.
4. Once an affirm result is obtained. The result is store in a dynamic predicate.
5. Diagnose the diseases by unification of the affirm result that are now in the dynamic predicates.

Model:

Dynamic predicates:

```
:- dynamic pain(X).
:- dynamic mood(X).
:- dynamic fever(X).
:- dynamic bowel(X).
:- dynamic miscellaneous(X).
:- dynamic not_pain(X).
:- dynamic not_mood(X).
:- dynamic not_fever(X).
:- dynamic not_bowel(X).
:- dynamic not_miscellaneous(X).
```

List of dynamic predicate to store the affirm result. They need to be able to modify during the program runtime hence define as dynamic.

Diseases:

```
c(cancer).
a(amoebiasis).
b(barbados).
r(rabies).
h(heart_trouble).
d(dehydration).
o(others).
```

List of possible diseases that the program can diagnose. This is added so that the GUi program can receive the feedback from the Prolog script. The CLI script does not this to be included.

Symptoms Library:

```
painLibrary([severe,moderate,mild,no]).
moodLibrary([gloomy, grumpy, guilty, indifferent, irritated, mad,
melancholy, pessimistic, rejected, restless, sad, stressed, weird]).
feverLibrary([extremely_high,high,mild,low,no]).
bowel_movementsLibrary([hard,loose,tarry,bloody,normal]).
miscellaneousLibrary([fatigue,coughing,itchy,giddy,hallucinating,twitches,p
alpitations,no_special_symptoms]).
```

List of possible symptoms that exists as the form of knowledge. This can be easily expanded in the further.

Disease Matching Rule:

```
cancer(X):-pain(strong),mood(gloomy),fever(mild),bowel(bloody),
          miscellaneous(giddy),c(X).
barbados(X):-
pain(mild),mood(grumpy),fever(low),miscellaneous(fatigue),b(X).
amoebiasis(X):-pain(unbearable),mood(weepy),fever(high),bowel(loose),
              miscellaneous(itchy),a(X).
rabies(X):-pain(strong),mood(angry),fever(high),bowel(normal),
          miscellaneous(hallucinating),r(X).
heart_trouble(X):-pain(unbearable),mood(stressed),fever(no),bowel(normal),
                 miscellaneous(palpitations),h(X).
dehydration(X):-pain(no),mood(calm),fever(no),bowel(hard),
               miscellaneous(giddy),d(X).
```

List of matching for diseases from the affirm result that are previously stored in the dynamic predicates. This too can be easily expanded in the further.

GUI Symptoms Extraction:

```
readpain(X):- painLibrary(Y), member(X,Y).
readmood(X):- moodLibrary(Y), member(X,Y).
readfever(X):- feverLibrary(Y), member(X,Y).
readbowel(X):- bowel_movementsLibrary(Y), member(X,Y).
readmiscellaneous(X):- miscellaneousLibrary(Y), member(X,Y).
```

These predicates are called by the GUI program. By calling the build in member function from Prolog. One element from the knowledge base of symptoms library is extracted and formatted into question and present to the patient.

Diagnose:

```
diagnose(X):-nl,
            cancer(X)->write("You may have cancer.");
            amoebiasis(X)->write("You may have amoebiasis.");
            barbados(X)->write("You may have barbados");
            rabies(X)->write("You may have rabies.");
            heart_trouble(X)->write("You may have heart trouble.");
            dehydration(X)->write("you may have dehydration.");
            o(X)->write("The current knowledge base cannot determine your
diseases").
```

This `diagnose/1` predicate is to output the first disease that return a True value. Diseases look up is by combining the diseases matching rule with disjunctions.

CLI Symptoms Passing:

```
readpain():-painLibrary(X),readpain(X).
readmood():-moodLibrary(X),readmood(X).
readfFever():-feverLibrary(X),readfever(X).
readbowel():-bowel_movementsLibrary(X),readbowel(X).
readmiscellaneous():-miscellaneousLibrary(X),readmiscellaneous(X).
```

Unlike the GUI version, CLI version make use of these predicates with arity 0 to help passig the symptoms list to the predicates that are responsible for asking question and affirming the extract symptom.

CLI Symptom Extraction

```
readpain(List):-
                  [H|T]=List,
                  format("Do you experience pain that is ~w",[H]),
                  read(X),
                  ((input(X);emptyList(T))->assert(pain(H));
                  readain(T)).
```

This predicate split the list into head and tail, with head containing only one element. If the patient affirm the element. That it is assert into the dynamic predicate other it will recursively extracting the element til empty.

Limitation

Since this is a linear searching and match. The position of the defining disease matching rule will have an impact on the result. For instance, if a disease `d` which has symptoms `a` and `b` is positioned before a disease `E` that has symptoms `a, b` and `c`, then disease `E` will never be found.

Appendix A

Instruction on running the GUI on various operating system.

Archlinux

```
pacman -S python-pyswip
pacman -S pyqt5-common
pacman -S swi-prolog
python App.py
```

Ubuntu

```
apt-get install python3-pip
apt-get install swi-prolog
pip install pyswip_alt pyqt5
python3 App.py
```

Windows

Due to limitation of Windows, the GUI Version can only be run with Python2.7.
Download from this link https://www.python.org/ftp/python/2.7.14/python-2.7.14.amd64.msi

Next due to limitation of Windows again, PyQT4 need to be installed separately with SIP pre-compiled.
Download from this link
https://sourceforge.net/projects/pyqt/files/PyQt4/PyQt-4.9.4/PyQt-Py2.7-x64-gpl-4.9.4-1.exe/download

Lastly, SWI-Prolog version 5.6 the shared library is no longer picked by the system even after environment variable added. Hence older version of SWI-Prolog is needed.
Download from this link http://www.swi-prolog.org/download/stable/bin/w64pl5661.exe

Add a new variable SWI_HOME_DIR as C:\Program Files\pl
Append c:\python27;C:\Program Files\pl\lib;C:\Program Files\pl\library;C:\Program Files\pl\bin to path variable
Install the pyswip package with pip, python -m pip install pyswip
Finally, run App.py