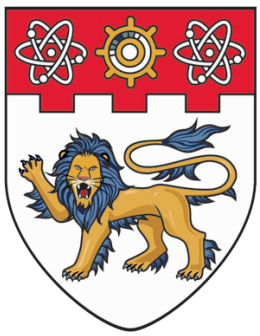


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science and Engineering

CZ3005 - Artificial Intelligence

Lab 2 TSP4

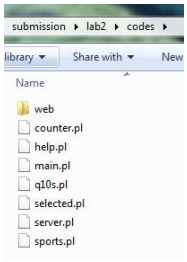
Done By:

Ching Jia Chin U1620237E

Table of Contents:

Setup:

1. Add all files, “counter.pl, help.pl, main.pl, q10s.pl, selected.pl, sports.pl, server.pl and web”, to a folder.



Files inside lab2 folder

2. Inside Prolog, change working directory to folder mentioned in step 1.

Using WebGUI

3. Inside Prolog, type “[‘server.pl’].”
4. Open http://localhost:5000/ on your browser. Refer to Appendix A below to see a screenshot of web GUI.

4.1.

```
?- working_directory(_, 'C:/Users/Jiachin/Desktop/CZ3005/submission/lab2').
true.

?- ['server.pl'].
Resetting 10 Questions

Theme is Sports!
Each question increases your score by 1.
Each wrong guess increases your score by 1.
Try to get the lowest score!

Starting round 1 of 5
Type 'help.'(no qoutes) to receive help or instructions.
Warning: c:/users/jiachin/desktop/cz3005/submission/lab2/server.pl:182:
        Singleton-marked variable appears more than once: _Request
Warning: c:/users/jiachin/desktop/cz3005/submission/lab2/server.pl:227:
        Singleton-marked variable appears more than once: _Request
% Started server at http://localhost:5000/
true.

[debug]  ?-
```

Loading server.pl

Using Prolog CMD line

5. Inside Prolog, type “[‘main.pl’].”

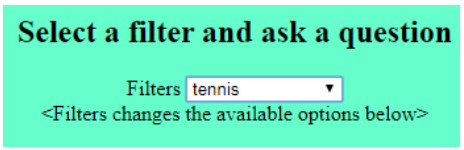
```
?- working_directory(_, 'C:/Users/Jiachin/Desktop/CZ3005/submission/lab2').
true.

?- ['main.pl'].
Resetting 10 Questions

Theme is Sports!
Each question increases your score by 1.
Each wrong guess increases your score by 1.
Try to get the lowest score!

Starting round 1 of 5
Type 'help.'(no qoutes) to receive help or instructions.
true.

?-
```

Loading main.pl

How to play (Web GUI):

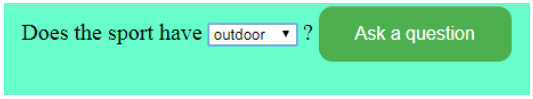
Asking a question

1. Select a filter. It shows what are the attributes of each sport and filters the questions you can ask. Refer to step 2 below.



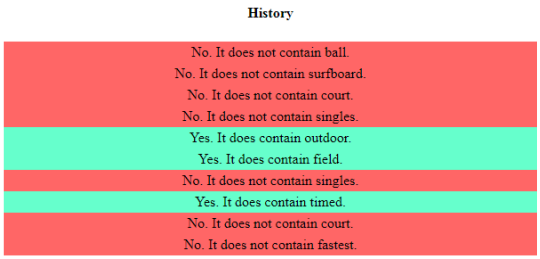
Select a filter

2. Choose a question to ask. Options are determined by the filter you select above. For example, outdoor is an attribute of tennis, so it appears in the question.



Select a question

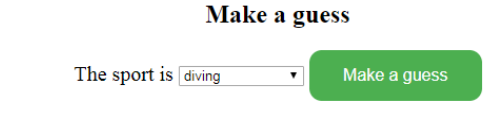
3. Your results will be shown in the history table.



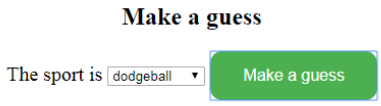
History table

Making a guess

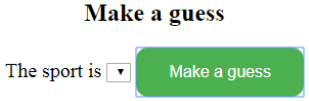
4. Choose a sport and make a guess. Wrong guesses are automatically removed. Refer to below.



Wrong Guess



Correct Guess



Game over <5 rounds are complete>

How to play (Using Prolog CMD line):

1. Type “help.” to see available commands.

```
?- help.
Type the following commands for options:

list(sports).  - Prints a list of all sports
                 This is a tree of depth 2. You can call list() on items of depth 1. E.g. list(tennis).
list(options). - Prints a list of all options.
                 This is a tree of depth 2. You can call list() on items of depth 1. E.g. list(equipment).
special. - Prints a list of special commands. E.g. has(maxteamsize(2))
                This is a tree of depth 2. You can call list() on items of depth 1. E.g. list(equipment).
getscore.        - Prints your score.
has(X).        - Ask if selected game has X. Replies Yes or No.
is(X).           - Guess the selected game.
true.
```

1.1. Type “list(sports).” To see a list of sports.

```
?- list(sports).
tennis              diving              golf                baseball
soccer              rugby               basketball          ultimate_frisbee
volleyball          badminton           swimming            skiing
surfing             archery             karate              bowling
dodgeball           table_tennis        hockey
true
```

1.2. Type “list(soccer)” to see the contents of soccer.

```
?- list(golf).
field               outdoor             singles             outdoor
scores              ball                performance         club
maxteamsize(1)
true .

?- list(soccer).
field               outdoor             ball                scores
timed               maxteamsize(11)
true .
```

2. Use “has(myquestion).” to ask a question.

```
?- has(ball).
Selected game does contain ball
true.

?- has(racket).
Selected game does not contain racket
true.
```

3. Use “is(myguess).” to guess the answer.

```
?- has(scores).
Selected game does contain scores
true.

?- is(soccer).
Sorry, that was the wrong guess. Try again!
true.

?- is(rugby).
You have guessed correctly!
Your score is 6. Your total score is 6. Try to get the lowest score!

Round ended. Starting new round in 3 seconds.

Starting round 2 of 5
Type 'help.'(no qoutes) to receive help or instructions.
true.
```

How it works:

Counters

Since prolog is a declarative language, it does not have variables. Instead, counters must be implemented in a declarative way. Counters was implemented by declaring counters as 1 initially. Every time increment is called, the following happens:

1. The current value of counter is incremented by 1
2. Retract previous declaration that counter equals <old value>
3. Declare counter as <new value>

This repeats each time increment is called. The other counters such as rounds and scores are implemented the same way.

Has()

Has(X) was implemented by comparing X with every item in the selected sport. If X matches an item, it will return true. This is done in Prolog by recursively checking every item in a list in sequence. If the end of the list is reached without finding a match, it returns false.

Is()

Is(X) was implemented by comparing X with the selected game. If they are the same, return true.

Traces:

Counter()

```
[trace]  ?- increment.
   Call: (8) increment ? creep
   Call: (9) counter(_4680) ? creep
   Exit: (9) counter(6) ? creep
^  Call: (9) retractall(counter(_4666)) ? creep
^  Exit: (9) retractall(counter(_4666)) ? creep
   Call: (9) succ(6, _4686) ? creep
   Exit: (9) succ(6, 7) ? creep
^  Call: (9) assertz(counter(7)) ? creep
^  Exit: (9) assertz(counter(7)) ? creep
   Exit: (8) increment ? creep
true.
```

Has()

```
[trace]  ?- has(ball).
   Call: (8) has(ball) ? creep
   Call: (9) increment ? creep
   Call: (10) counter(_4682) ? creep
   Exit: (10) counter(1) ? creep
^  Call: (10) retractall(counter(_4668)) ? creep
^  Exit: (10) retractall(counter(_4668)) ? creep
   Call: (10) succ(1, _4688) ? creep
   Exit: (10) succ(1, 2) ? creep
^  Call: (10) assertz(counter(2)) ? creep
^  Exit: (10) assertz(counter(2)) ? creep
   Exit: (9) increment ? creep
   Call: (9) score(_4690) ? creep
   Exit: (9) score(0) ? creep
   Call: (9) 0=<10 ? creep
   Exit: (9) 0=<10 ? creep
   Call: (9) incrementscore ? creep
   Call: (10) score(_4690) ? creep
   Exit: (10) score(0) ? creep
^  Call: (10) retractall(score(_4676)) ? creep
^  Exit: (10) retractall(score(_4676)) ? creep
   Call: (10) succ(0, _4696) ? creep
   Exit: (10) succ(0, 1) ? creep
^  Call: (10) assertz(score(1)) ? creep
^  Exit: (10) assertz(score(1)) ? creep
   Exit: (9) incrementscore ? creep
   Call: (9) selected(_4698) ? creep
   Exit: (9) selected(badminton) ? creep
   Call: (9) badminton(_4698) ? creep
   Exit: (9) badminton([court, indoor, singles, doubles, scores, knockout, shuttlecock, racket|...]) ? creep
   Call: (9) hasitem([court, indoor, singles, doubles, scores, knockout, shuttlecock, racket|...], ball) ? creep
   Call: (10) court=ball ? creep
   Fail: (10) court=ball ? creep
   Redo: (9) hasitem([court, indoor, singles, doubles, scores, knockout, shuttlecock, racket|...], ball) ? creep
   Call: (10) hasitem([indoor, singles, doubles, scores, knockout, shuttlecock, racket, maxteamsize(...)], ball) ? creep

   Call: (11) indoor=ball ? creep
   Fail: (11) indoor=ball ? creep
   Redo: (10) hasitem([indoor, singles, doubles, scores, knockout, shuttlecock, racket, maxteamsize(...)], ball) ? creep
```

```
      Call: (11) hasitem([singles, doubles, scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (12) singles=ball ? creep
      Fail: (12) singles=ball ? creep
      Redo: (11) hasitem([singles, doubles, scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (12) hasitem([doubles, scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (13) doubles=ball ? creep
      Fail: (13) doubles=ball ? creep
      Redo: (12) hasitem([doubles, scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (13) hasitem([scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (14) scores=ball ? creep
      Fail: (14) scores=ball ? creep
      Redo: (13) hasitem([scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (14) hasitem([knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (15) knockout=ball ? creep
      Fail: (15) knockout=ball ? creep
      Redo: (14) hasitem([knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (15) hasitem([shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (16) shuttlecock=ball ? creep
      Fail: (16) shuttlecock=ball ? creep
      Redo: (15) hasitem([shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Call: (16) hasitem([racket, maxteamsize(2)], ball) ? creep
      Call: (17) racket=ball ? creep
      Fail: (17) racket=ball ? creep
      Redo: (16) hasitem([racket, maxteamsize(2)], ball) ? creep
      Call: (17) hasitem([maxteamsize(2)], ball) ? creep
      Call: (18) maxteamsize(2)=ball ? creep
      Fail: (18) maxteamsize(2)=ball ? creep
      Redo: (17) hasitem([maxteamsize(2)], ball) ? creep
      Call: (18) hasitem([], ball) ? creep
      Call: (19) false ? creep
      Fail: (19) false ? creep
      Fail: (18) hasitem([], ball) ? creep
      Fail: (17) hasitem([maxteamsize(2)], ball) ? creep
      Fail: (16) hasitem([racket, maxteamsize(2)], ball) ? creep
      Fail: (15) hasitem([shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Fail: (14) hasitem([knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Fail: (13) hasitem([scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Fail: (12) hasitem([doubles, scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Fail: (11) hasitem([singles, doubles, scores, knockout, shuttlecock, racket, maxteamsize(2)], ball) ? creep
      Fail: (10) hasitem([indoor, singles, doubles, scores, knockout, shuttlecock, racket, maxteamsize(...)], ball) ? creep

      Fail: (9) hasitem([court, indoor, singles, doubles, scores, knockout, shuttlecock, racket|...], ball) ? creep
      Redo: (8) has(ball) ? creep
   ^  Call: (9) format("Selected game does not contain ~w", [ball]) ? creep
   Selected game does not contain ball
   ^  Exit: (9) format("Selected game does not contain ~w", [ball]) ? creep
      Exit: (8) has(ball) ? creep
   true.
```

Is()

```
[trace]  ?- is(tennis).
   Call: (8) is(tennis) ? creep
   Call: (9) selected(_4692) ? creep
   Exit: (9) selected(basketball) ? creep
   Call: (9) tennis=basketball ? creep
   Fail: (9) tennis=basketball ? creep
   Redo: (8) is(tennis) ? creep
   Call: (9) writeln("Sorry, that was the wrong guess. Try again!") ? creep
Sorry, that was the wrong guess. Try again!
   Exit: (9) writeln("Sorry, that was the wrong guess. Try again!") ? creep
   Call: (9) incrementscore ? creep
   Call: (10) score(_4720) ? creep
   Exit: (10) score(5) ? creep
^  Call: (10) retractall(score(_4706)) ? creep
^  Exit: (10) retractall(score(_4706)) ? creep
   Call: (10) succ(5, _4726) ? creep
   Exit: (10) succ(5, 6) ? creep
^  Call: (10) assertz(score(6)) ? creep
^  Exit: (10) assertz(score(6)) ? creep
   Exit: (9) incrementscore ? creep
   Exit: (8) is(tennis) ? creep
true.
```

Failed Is()

```
[trace]  ?- is(basketball).
   Call: (8) is(basketball) ? creep
   Call: (9) selected(_4694) ? creep
   Exit: (9) selected(basketball) ? creep
   Call: (9) basketball=basketball ? creep
   Exit: (9) basketball=basketball ? creep
   Call: (9) endround ? creep
   Call: (10) writeln('You have guessed correctly! ') ? creep
You have guessed correctly!
   Exit: (10) writeln('You have guessed correctly! ') ? creep
   Call: (10) getscore ? creep
   Call: (11) score(_4694) ? creep
   Exit: (11) score(6) ? creep
   Call: (11) addscoretotal ? creep
   Call: (12) score(_4694) ? creep
   Exit: (12) score(6) ? creep
   Call: (12) scoretotal(_4694) ? creep
   Exit: (12) scoretotal(0) ? creep
   Call: (12) _4700 is 6+0 ? creep
   Exit: (12) 6 is 6+0 ? creep
^  Call: (12) retractall(scoretotal(_4686)) ? creep
^  Exit: (12) retractall(scoretotal(_4686)) ? creep
^  Call: (12) assertz(scoretotal(6)) ? creep
^  Exit: (12) assertz(scoretotal(6)) ? creep
   Exit: (11) addscoretotal ? creep
   Call: (11) scoretotal(_4708) ? creep
   Exit: (11) scoretotal(6) ? creep
^  Call: (11) format("Your score is ~a. Your total score is ~a. Try to get the lowest score! ~n~n", [6, 6]) ? creep
Your score is 6. Your total score is 6. Try to get the lowest score!

^  Exit: (11) format("Your score is ~a. Your total score is ~a. Try to get the lowest score! ~n~n", [6, 6]) ? creep
   Exit: (10) getscore ? creep
   Call: (10) round(_4764) ? creep
   Exit: (10) round(1) ? creep
   Call: (10) 1<5 ? creep
   Exit: (10) 1<5 ? creep
   Call: (10) writeln('Round ended. Starting new round in 3 seconds. \n') ? creep
Round ended. Starting new round in 3 seconds.
```

Successful is()

Appendix:

Web GUI