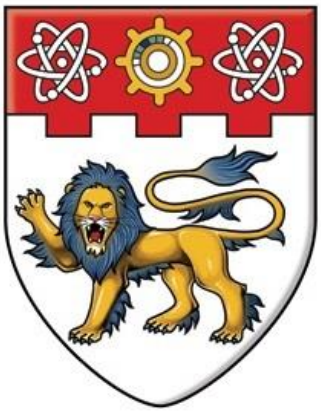

NANYANG TECHNOLOGICAL UNIVERSITY

School of Computer Science and Engineering

Artificial Intelligence Assignment 4 Report

Name (Matriculation):	Munirah Binte Mohamad (U1822418F)
Module Code:	CZ3005
Tut/Lab Group:	TSP2
Question Number:	1 (Ten Questions)

1 Introduction

This report is on Assignment 4 Question 1 “Ten Questions”. It is about a Prolog program that plays the role of the answerer while the player is the questioner. The topic is on Olympic Games. Therefore, the program decides on an Olympic sport and does not reveal the game to the player. The player is supposed to ask a maximum of ten questions to guess the answer.

2 Program Facts

There are several facts in the game that the program must keep track. We have facts such as the sport itself, the individual characteristic of each sports etc.

2.1 Static Facts

Static facts are facts that remain the same for the whole game. Static facts in the games are:

- The names of all the sports

```
sports([football, badminton, tennis, volleyball, karate, swimming, diving, golf]).
```

Figure 1: Names of all the sports

- The facts(characteristics) of each sport

```
football([ball, penalty, captain, referee, score, grass, pitch, teamsize=11, teams_per_game=2]).
badminton([racket, shuttlecock, referee, court, doubles, singles, teams_per_game=2]).
tennis([court, ball, racket, net, serve, doubles, singles, baseline, teams_per_game=2]).
volleyball([ball, sand, net, poles, empire, score, teamsize=6]).
karate([belts, uniform, judges, headgear, mouthguard, dojo, score]).
swimming([pool, referee, googles, water, lane, swimsuit, teamsize=1, teams_per_game=many]).
diving([water, pool, judges, swimsuit, googles, teamsize=1, teams_per_game= many]).
golf([ball, caddy, holes, clubs, score, course, grass, teamsize=1, teams_per_game=many]).
```

Figure 2: Facts of each sport

2.2 Dynamic facts

Dynamic facts are facts in the game that would change during the execution of the game. Dynamics facts in the game includes:

- counter() keep track of the number of questions that player has asked.
- gameselection() keeps track of the current sport that the program chose.
- already_played() keep tracks of all the sports that the player has guessed correctly

```
gameselection(empty).  /*Keep track of current sport guessing for*/
already_played().   /*Keep track of whic sports have already been played*/
counter(0).   /*Counter for no. of questions the player has asked for the current game*/

:- dynamic gameselection/1. :- dynamic gameselection/0.
:- dynamic counter/1.   :- dynamic counter/0.
:- dynamic already_played/1.:- dynamic already_played/0.
```

Figure 3: Dynamic Facts

3 Game Rules and Functions

The program must follow certain set of rules to execute the game properly.

3.1 init_var(List)

Init_var(List) is invoked when start_game() is entered. The purpose of this method is to assert all the dynamic facts of the program. 'List' contains all the sports that were initialised under sports.

This method starts by inserting all sports from already_played() into a list called Played. By subtracting Played from List, it returns a list that contain all the sports that the player has not guessed correctly. It is stored into a list called RemainingSports.

The program terminates if RemainingSports is empty. However, if it is not empty, a random sport from RemainingSports will be chosen by the program and placed into Sport. Sport is then asserted into already_played, so it cannot be randomly selected the next time. Counter is reset.

```
init_var(List) :-
    findnsols(100, A, already_played(A), Played),
    subtract(List, Played, RemainingSports),
    RemainingSports\==[],
    random_member(Sport, RemainingSports),
    assert(already_played(Sport)),
    retractall(gameselection(_)),
    assertz(gameselection(Sport)),
    retractall(counter(_)),
    assertz(counter(0));

    print('You guessed all the sports!'), nl,
    abort.
```

Figure 4: init_var(List) code

3.2 retrieve_list(A,B)

Retrieve_list(A,B) puts all facts(characteristics) of the given sport A into List B. This method checks if player input matches to the corresponding characteristic.

```
retrieve_list(A, B) :-
    (A == football, football(B));
    (A == badminton, badminton(B));
    (A == tennis, tennis(B));
    (A == volleyball, volleyball(B));
    (A == karate, karate(B));
    (A == swimming, swimming(B));
    (A == diving, diving(B));
    (A == golf, golf(B)).
```

Figure 5: retrieve_list(A,B) code

3.3 check_counter()

Check_counter() is used to check that the player can ask a maximum of ten questions only. This is done by loading the value of counter() and checking its value. If the value is larger than 10, the correct answer will be printed and game_over() will be invoked.

```
check_counter() :-
    counter(A),
    (A < 10);

    nl,
    print('No more guessing! Reached maximum 10 questions!'), nl,
    print('The answer is '),
    gameselection(A),
    print(A), nl, nl,
    game_over().
```

Figure 6: check_counter() code

3.4 increment()

increment() is used to increase the counter.

```
increment() :-
    counter(A),
    B is (A+1),
    retractall(counter(_)),
    assertz(counter(B)).
```

Figure 7: increment() code

3.5 guess_game()

This is the main portion of the game. Once queries have been evaluated, the game will return here.

This method starts by checking the check_counter(). If check_counter() is successful, this method will read an input from the player and continues the game. However, if it is unsuccessful, game_over() is invoked.

```
guess_game() :-
    check_counter(),
    read(Input),
    Input;

    game_over().
```

Figure 8: guess_game() code

3.6 concat_all_facts(L)

Concat_all_facts(L) concatenates all the facts about all sports in the list TotalList.

Each sport has been assigned with a list to place their facts. These lists will be concatenated into a single large List.

```
concat_all_facts(L) :-
    football(LLL1),
    badminton(LLL2),
    tennis(LLL3),
    volleyball(LLL4),
    karate(LLL5),
    swimming(LLL6),
    diving(LLL7),
    golf(LLL8),
    append(LLL1,LLL2, LL1),
    append(LLL3,LLL4, LL2),
    append(LLL5,LLL6, LL3),
    append(LLL7,LLL8, LL4),
    append(LL1,LL2,L1 ),
    append(LL3,LL4, L2),
    append(L1,L2, L).
```

Figure 9: concat_all_facts(L) code

3.7 Game_over()

game_over() ends the game. When it is invoked, the program asks the player if they want to keep playing and read the players's input. This input will be used as an argument in restart_game() method.

```
game_over() :-
    print('Would you like to play again? (y/n)'), read(Input), restart_game(Input).
```

Figure 10: game_over() code

3.8 restart_game(A)

restart_game (A) accepts the player's input. If player input is 'y', it will restart the game. If the player input is 'n', it will abort the program. However, if the player input is invalid, it will invoke game_over() again.

```
restart_game(A) :-
    (A==y, start_game());
    (A==n, abort);
    game_over().
```

Figure 11: restart_game(A) code

3.9 all_options()

all_options() is used to display all the facts that they can query about in has(Guess) query. A list to Set S conversion is needed to remove duplicate facts. This is invoked when the player starts the game so that they will know what options(facts) are available to query.

```
all_options() :-
    concat_all_facts(TotalList),
    list_to_set(TotalList, S),
    print('All possible options: '), nl,
    print(S), nl, nl,
    print('Player can start guessing: '), nl
    guess_game().
```

Figure 12: all_options() code

4 Player Interaction

There are two queries available to the player: has(Guess) and is(Guess). The figure below shows how the program is being designed and implemented into the code.

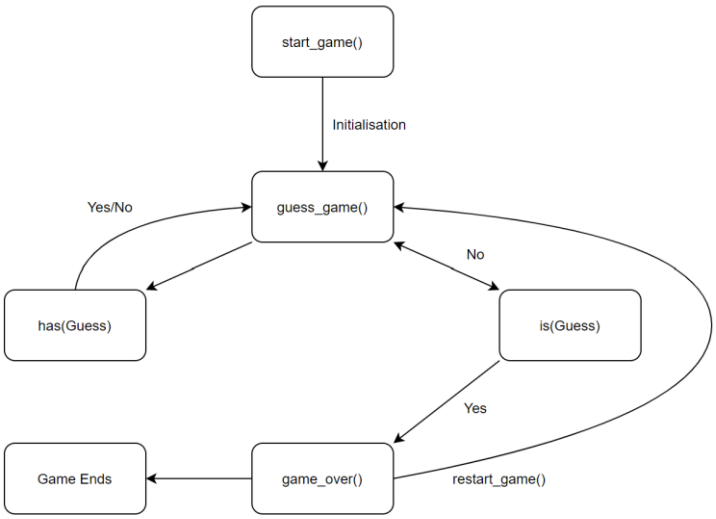

Figure 13: Representation of how the program works

4.1 start_game()

The game is started by entering 'start_game().' in the console. This method starts by inserting all sports into a single list 'List' and set the game variables by invoking 'init_var()'. Lastly, it will officially start the guessing game by invoking guess_game(), which lets the player to start asking questions.

```
start_game() :-
    sports(List),
    init_var(List), nl,
    print('Game has started!'), nl, nl,
    all_options(), nl,
    guess_game().
```

Figure 14: start_game() code

4.2 has(Guess)

Has(guess) is the query that answers if the current sport selected by the program has that certain fact.

gameselection() loads the sport selected into Sport. Retrieve_list(List) will find the corresponding list to the sport selected in List. If the query is a member of Lista, the guess is correct. If it is not in List, the program will print 'No', the counter will not be increased and returns to guess_game().

```
has(Guess) :-
    gameselection(Sport),
    retrieve_list(Sport, List),
    member(Guess, List),
    print('Yes·'), nl,
    increment(),
    guess_game();

    concat_all_facts(TotalList),
    member(Guess, TotalList),
    print('No·'), nl,
    increment(),
    guess_game();

    print('Invalid option'),nl,
    guess_game()·
```

Figure 15: has(Guess) code

4.3 is(Guess)

is(Guess) is the query that checks if the player makes a correct guess.

The program loads the selected sport into A and compare if the player guess is equal to the sport A. If it is successful, the player is correct and the game is over. If it is unsuccessful, the counter will increase by one and the program returns to guess for another query.

```
is(Guess) :-
    gameselection(A),
    A==Guess,
    print('Yes it was '),
    print(Guess), nl, nl,
    game_over();

    print('No, it is not '),
    print(Guess), nl,
    increment(),
    guess_game()·
```

Figure 16: is(Guess) code

5 Program Run

This is how the program behaves during its execution.

5.1 Game ends when correct answer is guessed

The figure below shows how the game is being played. The player is able to guess the correct answer after a few queries ad terminates the game.

```
?- start_game().
'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'No.'
|    has(shuttlecock).
'No.'
|    has(caddy).
'No.'
|    has(doubles).
'No.'
|    has(belts).
'Yes.'
|    has(dojo).
'Yes.'
|    is(basketball).
'No, it is not 'basketball
|    is(karate).
'Yes it was 'karate
'Would you like to play again? (y/n)'|    n.

% Execution Aborted
```

Figure 17: Game ends when correct answer is guessed

5.2 Player reached maximum 10 questions

The figure below shows the game stops the player from entering more guesses once the player has reached the tenth question.

```
?- start_game().
'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'No.'
|    has(shuttlecock).
'Yes.'
|    is(tennis).
'No, it is not 'tennis
|    has(dojo).
'No.'
|    has(teamsize=11).
'No.'
|    has(court).
'Yes.'
|    is(football).
'No, it is not 'football
|    has(lane).
'No.'
|    has(doubles).
'Yes.'
|    has(net).
'No.'

'No more guessing! Reached maximum 10 questions!'
'The answer is 'badminton

'Would you like to play again? (y/n)'|    n.

% Execution Aborted
```

Figure 18: Player reached maximum 10 questions

5.3 Player can play at least 5 times

The figure below shows that the player is able to play the game at least 5 times.

```
?- start_game().

'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'No.'
|    has(shuttlecock).
'No.'
|    has(water).
'Yes.'
|    is(swimming).
'Yes it was 'swimming

'Would you like to play again? (y/n)'|    y.

'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'No.'
|    has(shuttlecock).
'No.'
|    has(water).
'No.'
|    is(karate).
'Yes it was 'karate

'Would you like to play again? (y/n)'|    y.

'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'No.'
|    has(shuttlecock).
'Yes.'
|    is(badminton).
'Yes it was 'badminton

'Would you like to play again? (y/n)'|    y.


'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'No.'
|    has(water).
'Yes.'
|    is(diving).
'Yes it was 'diving

'Would you like to play again? (y/n)'|    y.

'Game has started!'

'All possible options: '
[ball,penalty,captain,referee,score,grass,pitch,teamsize=11,teams_per_game=2
,racket,shuttlecock,court,doubles,singles,net,serve,baseline,sand,poles,empi
re,teamsize=6,belts,uniform,judges,headgear,mouthguard,dojo,pool,googles,wat
er,lane,swimsuit,teamsize=1,teams_per_game=many,caddy,holes,clubs,course]

'Player can start guessing: '
|    has(ball).
'Yes.'
|    has(court).
'No.'
|    has(grass).
'Yes.'
|    is(football).
'Yes it was 'football

'Would you like to play again? (y/n)'|    n.

% Execution Aborted
```

Figure 19: Game can reach 5 rounds