COMP 348: Principles of Programming Languages

Assignment 1 on Logical Programming

Summer 2020, sections AA and AB

May 3, 2020

Achoura Bague (27877986)

Caleb Hoyne (480298)

Shadi Jiha (40131284)

**Question 1:**

/\* Question 1 \*/

team(['40131284','27877986', '480298']).

student\_info('40131284', 'Shadi', 'Jiha').

student\_info('27877986', 'Achoura', 'Bague').

student\_info('480298', 'Caleb', 'Hoyne').

takes\_course('40131284', 'comp', '348', 'aa').

takes\_course('40131284', 'comp', '348', 'aaaf').

takes\_course('40131284', 'comp', '352', 'aa').

takes\_course('40131284', 'comp', '352', 'aaae').

takes\_course('27877986', 'comp', '348', 'aa').

takes\_course('27877986', 'comp', '348', 'aaaf').

takes\_course('27877986', 'comp', '333', 'cc').

takes\_course('480298', 'comp', '348', 'aa').

takes\_course('480298', 'comp', '348', 'aaaf').

course\_schedule('comp', '348', 'aa', 'mon', '1445', '1715').

course\_schedule('comp', '348', 'aa', 'wed', '1445', '1715').

course\_schedule('comp', '348', 'aaaf', 'mon', '1315', '1405').

course\_schedule('comp', '348', 'aaaf', 'wed', '1315', '1405').

course\_schedule('comp', '352', 'aa', 'tue', '1830', '2100').

course\_schedule('comp', '352', 'aaae', 'wed', '1315', '1415').

course\_schedule('comp', '333', 'cc', 'tue', '1315', '1545').

course\_schedule('comp', '333', 'cc', 'thu', '1315', '1545').

**Question 2:**

Output:

40131284 has only taken 4 courses and tutorials in summer 2020.  
27877986 has only taken 3 courses and tutorials in summer 2020.  
480298 has only taken 2 courses and tutorials in summer 2020.  
false

**Question 3:**

1. Using the code from question 1 we can write:

/\* Question 3 A \*/

get\_sections(CNAME, CNUM, L) :- findall(L, course\_schedule(CNAME, CNUM, L, \_, \_, \_), L).

all\_sections(CNAME, CNUM, L) :- get\_sections(CNAME, CNUM, T), list\_to\_set(T, L).

1. Using code from Q1:

/\* Question 3 B \*/

has\_taken(S, [CNAM|[CNUM|[SEC|[]]]]) :- takes\_course(S, CNAM, CNUM, SEC).

1. Using code from Q1:

/\* Question 3 C \*/

has\_taken2(S, [CNAM|[CNUM|[]]]) :- takes\_course(S, CNAM, CNUM, \_).

1. Using code from Q1:

/\* Question 3 D \*/

get\_courses(S, L):- findall(T, takes\_course(S, T, \_, \_), L).

all\_subjects(S, L) :- get\_courses(S, T), list\_to\_set(T, L).

1. For this question we have to redefine the predicates:

/\* Question 3 E \*/

takes\_course\_id('40131284', '10').

takes\_course\_id('40131284', '11').

takes\_course\_id('40131284', '12').

takes\_course\_id('40131284', '13').

takes\_course\_id('27877986', '10').

takes\_course\_id('27877986', '11').

takes\_course\_id('27877986', '14').

takes\_course\_id('480298', '10').

takes\_course\_id('480298', '11').

course\_by\_id('10', ['comp', '348', 'aa']).

course\_by\_id('11', ['comp', '348', 'aaaf']).

course\_by\_id('12', ['comp', '352', 'aa']).

course\_by\_id('13', ['comp', '352', 'aaae']).

course\_by\_id('14', ['comp', '333', 'cc']).

get\_course\_by\_id(ID, L) :- course\_by\_id(ID, L).

get\_student\_courses(S, L) :- findall(T, takes\_course\_id(S, T), L).

id\_to\_list([], R).

id\_to\_list([L|T], [R|S]) :- get\_course\_by\_id(L, R), id\_to\_list(T, S).

all\_courses(S, L) :- get\_student\_courses(S, R), id\_to\_list(R, L).

1. For this question all we have to do is remove the section from the course\_by\_id and everything else can stay the same.

**Question 4:**

team(X), member(S, X),

findall(S,(takes\_course(S, \_, \_,aa)),LL),

length(LL, NN),

write(S), write(' has only taken '), write(NN),

write(' courses and tutorials in summer 2020.'), nl, fail.

**Question 5:**

Instead of getting the desired list, i.e. L = [ ['comp', '348', 'aa'], ['comp', '348', 'ab'] ] we get a Boolean result. This is mainly due to the fact that '4000123' is a quoted atom while 4000123 is a number. The situation would be different if the id starts with a lowercase letter. Because in prolog ‘i4000123’ and i4000123 are pretty much the same thing.

**Question 11:**

add\_to\_start\_list(X, [], [X]). % this function inserts an element to the beginning of the list

add\_to\_start\_list(X, [H|T], [H|T2]) :-

add\_to\_start\_list(X, T, T2).

reverse\_list([], []). % this function reverses the order of a list

reverse\_list([H|T], Out) :-

reverse\_list(T, Out1),

add\_to\_start\_list(H, Out1, Out).

lucas(1, 2). % Here we have the first base case (first number of lucas is 2)

lucas(2, 1). % Here we have the second base case (first second of lucas is 1)

lucas(N, F) :-

N > 1,

N1 is N - 1,

N2 is N - 2,

lucas(N1, F1),

lucas(N2, F2),

F is F1 + F2.

lucas\_sequence(0, [0]).

lucas\_sequence(N, [H|T]) :- % This functions produces a lucas sequence inside a list but in reverse order

lucas(N, H),

N1 is N - 1,

lucas\_sequence(N1, T).

lucas\_sequence\_to\_list(In, Out) :- % This functions produces a lucas sequence inside a list in correct order

lucas\_sequence(In, Out1),

reverse\_list(Out1, [H|Out]).