Concordia University
Department of Computer Science and Software Engineering
COMP 348: Principles of Programming Languages
Winter 2021
Assignment 1
Evaluation: 100 pts (+20 bonus)
Due date and time: Before Friday Feb 19th 2021 at 23:59

Question 1 (15 pts) **Knowledge representation in prolog**

Create a prolog database capturing the information about your family. The information must be created in terms of a procedure named individual which should have the below mentioned syntax.

individual(tom, male, adam, eve).
individual(sandra, female, john, jenny).

The facts comprise of individual's name, sex, and parents' names. Create a minimum of 15 facts to make the database large enough.

Create the below mentioned rules to query useful information from the database.

1. offspring(X, Y) /* Hint: X is an offspring of Y which means Y is one of X's parents*/
2. niblings(X, Y) /* Hint: X is nibling of Y which means X is Y's sibling's offspring */
3. puncle(X, Y) /* Hint: X is Y's paternal uncle which means X is Y's father's brother*/
4. modrige(X, Y) /* Hint: X is Y's modrige which means X is Y's mother's sister */
5. avuncle(X, Y) /* Hint: X is Y's avuncle which means X is Y's mother's brother*/

Question 2 (15 pts) **Unifications and resolutions in Prolog**

Which of the following pairs of terms can be unified together? If they can't be unified, please provide the reason for it. If they can be unified successfully, wherever relevant, provide the variable instantiations that lead to successful unification. (Note = shows unification)

likes(jane, X) = likes(X, josh).
diSk(27, queens, sgt_pepper) = diSk(A, B, help).
[a,b,c] = [X,Y,Z|T].
ancestor(french(jean), B) = ancestor(A, irish(joe)).
characters(hero(luke), X) = characters(X, villain(vader)).
f(X, a(b,c)) = f(d, a(Z, c)).
s(x, f(x), z) = s(g(y), f(g(b)), y).
vertical(line(point(X,Y), point(X,Z))) = vertical(line(point(1,1),point(1,3))).
g(Z, f(A, 17, B), A+B, 17) = g(C, f(D, D, E), C, E).
f(c, a(b,c)) = f(Z, a(Z, c)).

Question 3 (20 pts) **Queries in Prolog**

Assume we have the following knowledge base in a Prolog program:

```
building(engineering, ev).
building(business, mb).
building(library, lb).
building(classes, h).
building(hr, fg).
department(electrical, engineering).
department(civil, engineering).
department(finance, business).
department(ibm-exams, lb).
status(engineering, accredited).
faculty(smith, electrical).
faculty(walsh, electrical).
faculty(smith, computer).
faculty(jones, civil).
faculty(james, civil).
faculty(davis, civil).
faculty(X, Y) :- department(Z, Y), faculty(X, Z).
building(X, Y) :- department(X, Z), building(Z, Y).
status(X, Z) :- department(X, Y), status(Y, Z).
faculty(X) :- faculty(X, _).
```

Determine the type of each of the following queries (ground/non-ground), and explain what will Prolog respond for each of these queries (write all the steps of unifications and resolutions for each query)?

```
? building(library, lb).
? status(finance, A).
? department(civil, Bussiness).
? faculty(X, civil).
? faculty(smith, X).
? department(X, Y).
? faculty(X, civil), department(civil, Y).
?  faculty(Smith).
?  building(_, X).
? status(X, accredited), building(X, Y).
? status(_, X), building(X, Y).
? faculty(X), faculty(X, Y), department(Y, _).
?  faculty(X), faculty(X, Y), !, department(Y, Z).      % note there is a cut (!) here
?  faculty(X), !, faculty(X, _).                        % note there is a cut (!) here
? department(X, _), \+ faculty(_, X).
```

Question 4 (25 pts) **Using Prolog for a Search Problem**

Assume, you are working with the following knowledge base:

```
family(person( john, cohen, date(17,may,1990), unemployed),
        person( lily, cohen, date(9,may,1990), unemployed),
        [ ] ).
family(person( john, armstrong, date(7,may,1988), unemployed),
        person( lily, armstrong, date(29,may,1961), unemployed),
        [ ] ).
family(person( eric, baily, date(7,may,1963), works( bbc, 2200)),
        person( grace, baily, date(9,may,1965), works( ntu, 1000)),
        [person( louie, baily, date(25,may,1983), unemployed) ] ).
family(person( eric, baily, date(7,may,1963), works( acc, 21200)),
        person( grace, baily, date(9,may,1965), works( ntnu, 12000)),
        [person( louie, baily, date(25,may,1983), unemployed) ] ).
family(person( eric, fox, date(27,may,1970), works( bbc, 25200)),
        person( grace, fox, date(9,may,1971), works( ntbu, 13000)),
        [person( louie, fox, date(5,may,1993), unemployed) ] ).
family(person( tom, cohen, date(7,may,1960), works( bcd, 15200)),
        person( ann, cohen, date(29,may,1961), unemployed),
        [person( pat, cohen, date(5,may,1983), works( bcd, 15200)),
         person( jim, cohen, date(5,may,1983), works( bcd, 15200)) ] ).
family(person( bob, armstrong, date(12,oct,1977), works( ntnu, 12000)),
        person( liz,armstrong, date(6,oct,1975), unemployed),
        [person( bob, armstrong, date(6,oct,1999), unemployed),
        person( sam,armstrong, date(8,oct,1998), unemployed) ] ).
family(person( tony, oliver, date(7,may,1960), works( bbc, 35200)),
        person( anny, oliver, date(9,may,1961), unemployed),
        [person( patty, oliver, date(8,may,1984), unemployed),
        person( jimey, oliver, date(5,may,1983), unemployed) ] ).
family(person( jack, fox, date(27,may,1940), unemployed),
        person( jane, fox, date(9,aug,1941), works( ntu, 13050)),
        [person( andy, fox, date(5,aug,1967), works( com, 12000)),
         person( kai, fox, date(5,jul,1969), unemployed) ] ).
husband(X) :- family( X, _, _).
wife(X) :- family( _, X, _).
child(X) :- family( _, _, Children), member(X, Children).
exists(Persons) :- husband( Persons); wife( Persons); child( Persons).
dateofbirth(person(_, _, Date, _), Date).
salary(person(_, _, _, works(_, S)), S).
salary(person(_, _, _, unemployed), 0).
```

Write the details of steps of search (unification, resolutions, and back tracking) and also the answer for each of the following queries. (You can show the details of your search process by drawing search trees for each of the following queries)

```
? exists(P), dateofbirth(P, date(_,_,Y)), Y<1963, salary(P, Salary), Salary<15000.
? exists(P), dateofbirth(P,date(_,_,Y)), !, Y<1998, salary(P,Salary), Salary<20000.
? wife(person(GivenName, FamilyName, _, works(_,_))).
? child(X), dateofbirth(X, date(_,_,1983)).
```

Question 5 (25 pts) **Lists and Backtracking**

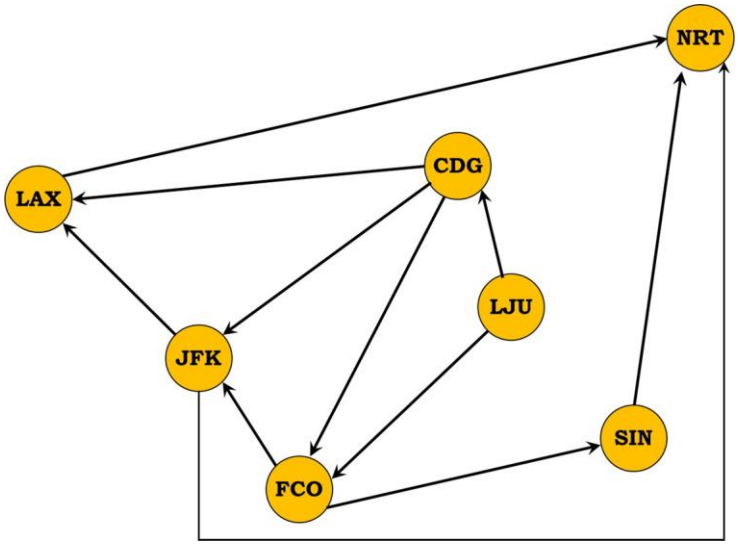
Consider the database from the previous question and answer the following

1) Write a prolog rule totalIncome/2 to compute the total income of a family.
2) Write a prolog query to print total income of each family.
3) Write a prolog query to print family details of each family that has income per family member less than 2000.
4) Write a prolog query to print family details of each family where children's total income is more than their parents.

(BONUS) Question 6 (20 pts**) Graphs in Prolog**

The Purpose of this question is to write a Prolog Program which describes a directed graph (G), with the following structure and allows us to ask some questions about this graph. From the graph described below create a prolog database with the below mentioned predicates.

(a) flightPath/4 which stores originatingAirport, destinationAirport, flightTime and distance (Make use of flightconnections.com to find out the flight times (round it off to the closest integer e.g. 1h20 mins =1 and 1h35 = 2, *etc.*)
(b) transferTime/2 which stores airport and time spent in immigration, baggage, *etc.* (You can choose a suitable rounded integer for time spent)



A. Write the below mentioned prolog rules and show some sample results by querying them.

1) connection(Start, Destination) to check whether destination can be reached from the starting airport or not. You should consider direct as well as indirect paths.
2) flightTime(Start, Destination, Time, Path) to compute the flight time for all possible paths.
3) pathLength(Path, Length) to compute the length of a given path (path will be a list).
4) shortestPath(Start, Destination) to print the shortest path between two airports.

B. Create a knowledge base (KB) composed of all the facts and rules that you have written in parts (1), and (2), and save it in a prolog program file called "flights.pl", and show a print of your program here.

C. Run this Prolog program, and write 4 queries about each of those rules that you have created in part (A) in order to test those rules, and show your queries and their results, and test the correctness of your results. **(Note: you should write and test two ground quires (one with positive and one with negative answer) and two non-ground quires for each of the rules that you have created.)**

Submission:

- **Assignment must be done individually (no groups are permitted).**
  - Create one zip file, containing all files for your assignment.
  - Name your zip file this way:*a1_studentID*, where *studentID* is your ID number, for the first assignment, student 123456 would submit a zip file named a1_123456.zip
- Assignments must be submitted through Moodle.

Note:

- **Assignment not submitted by the due date and in the correct format will not be graded NO EXCEPTIONS!!!!**
- **Copying, plagiarism, cheating whatever you name it is STRICTLY prohibited and result in receiving ZERO in the assignment and reporting your name to the school for further disciplinary actions. This includes copying from your classmates, Internet repos or anything that is NOT genuinely your own work. All submissions will be automatically cross checked for plagiarism.**