# cps721: Assignment 1 (100 points).
# Due date: Electronic file - before Monday, September 28, 2015, 21:00.

**Hand in a printed copy at start of class on Tuesday, September 29**.

*You have to work in groups of TWO, or THREE. You should not work alone.*

YOU SHOULD NOT USE "**;**" (disjunction) "**!**" (cut) AND "$->$" IN YOUR PROLOG PROGRAMS.

You can discuss this assignment only with your CPS721 group partners or with the CPS721 instructor. By submitting this assignment you acknowledge that you read and understood the course Policy on Collaboration in homework assignments stated in the CPS721 course management form.

This assignment has 3 pages. On the 3rd page, you can find the rules about submitting this assignment electronically. Improperly submitted assignments will not be marked.

**1 (40 points).** This problem is designed to get you started in Prolog. Make up a simple knowledge base (KB) of atomic sentences (10-15 atomic sentences for each predicate) about movies (not necessarily real ones), so that most of the queries below will successfully retrieve answers from your KB. If you would like to read about real movies, you can consult *http://www.imdb.com/* Keep your KB in the file **movies.pl** Use the following predicates.

*actsIn(Person,Movie)* – the *Person* is acted in the *Movie*.
*directs(Person,Movie)* – the *Person* directed the *Movie*.
*released(Movie,Year)* – the *Movie* was released during the *Year*.

Now pose the following queries to Prolog, and obtain Prolog's answers. You are not allowed to write any rules in this part of your assignment. You can only write atomic statements to implement your KB, and Prolog queries to retrieve answers from your KB. As for syntax of your queries, you can use only negation "not", conjunction, and predicates mentioned above with variables or constants as arguments. You cannot use any other predicates.

1. Did Robert De Niro star in *Awakenings*?
2. Who directed *Rain Man*?
3. Did anyone star in both *Apocalypse Now* and *The Godfather*?
4. Was there a movie released in 2014 where neither Cate Blanchett nor Leonardo DiCaprio acted?
5. Does any movie have more than one director? Have Prolog list all such movies and their directors by using the ; command (if you work with the command-line Prolog) or by using *more* button (if you work with a GUI version of ECLiPSe Prolog).
6. Did any director of "Forrest Gump" ever directed movies released during or after 2000? Note that $X < Y$ (and $X > Y$) is Prolog's `less than` (`greater than`, respectively) predicate: $X < Y$ succeeds if the number $X$ is less than the number $Y$. Similarly, $X =< Y$ succeeds if the number $X$ is less than or equal to the number $Y$. Both variables $X, Y$ should be instantiated before comparison $X =< Y$.
7. Has anyone acted in more than two movies released in the same year?
8. Who has worked with the same director in different years? Have Prolog list all such actors, directors, the movies they worked on together, and the years they were released using the ; command (or by clicking the "more" button on GUI).
9. Has anyone directed **only one** movie in 2015? Suppose the answer *Yes*. Add new atomic statements about this director and imaginary movie s/he directed in 2015. After this change, if answer to your query is still yes, then the query is wrong. Pay attention to the phrase **only one** and think how it can be implemented for an arbitrary large KB.
10. What movie has the earliest year of release? Note that if you add to your KB a new (real or imaginary) movie with an earlier release date (e.g., a movie directed by Charlie Chaplin that was not previously in your KB), but submit same query, then answer to your query should change too, if the query is correct.

**Handing in solutions**: (a) An electronic copy of your program with all atomic statements (the name of this file must be **movies.pl**); (b) An electronic copy of your session with Prolog that shows clearly the queries you submitted to Prolog, and the answers that were returned (the name of this file must be **movies.txt**); (c) Print and staple both files **movies.pl** and **movies.txt** and hand printouts in class.

**2 (20 points).** This problem will exercise what you have learned about rules in Prolog, by having you implement a set of rules for calculating the final letter grade in the CPS721 course.

a) Write a Prolog file **grade.pl** with the *knowledge base* (KB) about quizzes, assignments and tests in CPS721. Your KB should include a collection of atomic statements in Prolog using the following predicates:

*assignment(Number,AScore)* – a student got *AScore* for the assignment *Number*, where $0 \leq AScore \leq 100$.
*quiz(Number,QScore)* – a person has received *QScore* for the quiz *Number*, where $0 \leq QScore \leq 2$.
*midterm(Mark1)* – a student has received *Mark1* for the midterm test, where $0 \leq Mark1 \leq 20\%$.
*exam(Mark2)* – a person received *Mark2* for the final exam, where $0 \leq Mark2 \leq 50\%$.

b) Subsequently, use these predicates to define new predicates as specified below. Write the rules in the same file **grade.pl**

- Write the rule calculating the total value $A$ of the five assignments. Use the new predicate *sumAssignments(A)* in the head of your rule and relate it with the atomic statements about the scores for all five assignments. Since in CPS721 each assignment is out of 100 points, but in total the assignments contribute 20% of the final grade, you have to scale down the sum of scores of all assignments to calculate the number $0 \leq A \leq 20\%$.
- Introduce new predicate *sumQuizzes(Q)* and use it to calculate the total value $0 \leq Q \leq 10\%$ contributed to the final grade by all the ten quizzes. Since each quiz is marked out of 2 points, the sum of scores for quizzes has to be scaled down.
- Let predicate *grade(G)* is true if $G$ is the final grade in % obtained by a student in the CPS721 course. Write a single rule defining this predicate using the predicates introduced before: *sumAssignments(A)*, *sumQuizzes(Q)*, *midterm(M)*, *exam(E)*. The CPS721 grading scheme is defined in the CPS721 course management form.
- Let predicate *letter(L,G)* holds if $L$ is the Ryerson letter grade representing the total numerical course grade $0 \leq G \leq 100\%$. Consult the Ryerson university Web site for an exact definition of the final letter grades in the Faculty of Sciences. Write several Prolog rules implementing this predicate *letter(L,G)* according to the Faculty of Science university grading scheme. Make sure you do rounding correctly when calculating the letter grade.

(c) Once you have implemented all these predicates, test your rules by submitting queries to Prolog, obtain answers, and store a copy of your interaction with Prolog in a file. Try at least 5 queries to demonstrate your program works correctly. It is up to you to decide which queries you would like to try. In this and in the next question, when you write rules (conditional clauses), you are not allowed to use the following commands: "**;**" (disjunction),

**Handing in solutions.** An electronic copy of: (a) your program (the name of the file must be **grade.pl**) that includes all your Prolog rules and your atomic statements; (b) your session with Prolog, showing the queries you submitted and the answers returned (the name of the file must be **grade.txt**). It is up to you to formulate a range of queries that demonstrates that your program is working properly. (c) Hand in printouts of both **grade.pl** and **grade.txt** in class.

**3 (40 points).** As you may know, Geographic Information Systems (GIS) technology is based on cartography, databases and statistical analysis. The world's first fully-operational GIS was developed in 1962 in Ottawa, Ontario, Canada by the federal Department of Forestry and Rural Development. For the purposes of this assignment, imagine a prototype of a GIS that can answer queries about a city through a hand-held mobile device. This imaginary system would have information about all streets, subway stations, buildings and other locations in the city. Below, we speak of "entities" to use a generic term that may represent city streets, subway stations, universities, libraries or any other locations of importance. For the GIS to operate correctly, it should be provided with numerous logical rules for reasoning about directions and entities in the city. In this question, you are asked to implement recursive rules about the four geographic cardinal directions only. More specifically, you are given the following predicates (east includes proper east, south-east and north-east directions):

*eastOf(Entity1,Entity2)* – *Entity1* is east of *Entity2*, *westOf(Entity1,Entity2)* – *Entity1* is west of *Entity2*
*southOf(Entity1,Entity2)* – *Entity1* is south of *Entity2*, *northOf(Entity1,Entity2)* – *Entity1* is north of *Entity2*.

(a) Start with implementing a small *knowledge base* (KB) of *atomic* statements. Your KB can be real, e.g., it include information about streets in Toronto, or it can be imaginary; this is up to you. Your atomic statements can use 4 new predicates describing relations between the regions that are immediately adjacent to each other in one of the 4 cardinal directions.

(b) Next, add logical *rules* (conditional statements). Given the predicates mentioned above and your new predicates from (a), write all recursive rules one might need to reason about cardinal direction of any entity with respect to others. Do not introduce any other predicates. Remember that you have to write **all** correct rules, and **nothing but correct rules**. For example, if we know that $X$ is east of $Y$, we should be able to conclude that $Y$ is west of $X$. In addition, if we know that $X$ is south of $Y$, and $Z$ is east of $Y$, then we know that $X$ is south of $Z$, and so on. Warning: as stated on the top of this assignment, when you write rules in Prolog, you are not allowed to use the following commands: "**;**" (disjunction), "**!**" (cut) and "$->$" (if-then).

(c) Test your rules on your atomic statements using queries about entities named in your KB. Try as many testing queries as needed to make sure that you got a complete set of correct rules. In particular, make sure you get all correct and only correct answers when you submit a query $?- \ \ westOf(X,Y)$. and similar queries with variables. Repetitions of correct answers are allowed. The queries up to you; they should convince the TA that you have implemented a correctly working recursive program.

**Handing in solutions.** An electronic copy of: (a) your program (the name of the file must be **gis.pl**) that includes all your correct rules. (b) An electronic copy of your session with Prolog that shows clearly the queries you submitted to Prolog, and the answers that were returned (the name of this file must be **gis.txt**); (c) Hand in a printout of **gis.pl** and **gis.txt**) in class.

**How to submit this assignment.** Hand in printouts of all files before beginning of class. On the first page of your printout **write the name of your electronic ZIP file** to make sure the TA can locate it easily among other ZIP files. **Staple all pages!** Read regularly *Frequently Answered Questions* and replies to them that are linked from the Assignments Web page at

```
http://www.scs.ryerson.ca/~mes/courses/cps721/assignments.html
```

If you write your code on a Windows machine, make sure you save your files as plain text that one can easily read on Linux machines. Before you submit your Prolog code electronically make sure that your files do not contain any extra binary symbols: it should be possible to load `movies.pl` or `grade.pl` or `gis.pl` into a recent release 6 of ECLiPSe Prolog, compile your program and ask testing queries. TA will mark your assignment using ECLiPSe Prolog. If you run any other version of Prolog on your home computer, it is your responsibility to make sure that your program will run on ECLiPSe Prolog (release 6 or any more recent release), as required. For example, you can run a command-line version of *eclipse* on moon remotely from your home computer to test your program (read handout about running *ECLiPSe Prolog*). To submit files electronically do the following. First, create a **zip** archive:

```
zip yourLoginName.zip movies.pl movies.txt grade.pl grade.txt gis.pl gis.txt
```

where `yourLoginName` is the login name of the person who submits this assignment from a group. Remember to mention at the beginning of each file *student, section numbers* and *names* of all people who ever participated in discussions (see the course management form). You may be penalized for not doing so. Second, run the command

```
submit-cps721 yourLoginName.zip
```

on one of the `moons`. If you need more information about **zip**, read manual pages on *moon*. Make sure you have **ssh** software on your home computer. You might need it to establish a remote connection with departmental servers, if you are going to submit your assignment from home. Otherwise, submit your assignment from labs at any time. Improperly submitted assignments will not be marked. In particular, you are **not** allowed to submit your assignment by email to a TA or to the instructor: no exceptions. Make sure that your Computer Science account at Ryerson is properly configured and you can easily access it from Labs (or from home, if you wish). Talk to one of the system administrators in ENG246 or ENG248 *in advance* if needed.

**Revisions**: If you would like to submit a revised copy of your assignment, then run simply the submit command again. (The same person must run the submit command.) A new copy of your assignment will override the old copy. You can submit new versions as many times as you like. You do not need to inform anyone about this. Don't ask your team members to submit your assignment, because TA will be confused which version to mark. Only one person from a group should submit different revisions of the assignment. The time stamp of your last ZIP file determines whether you have submitted your work on time.