cps721: Assignment 4 (100 points).
Due date: Electronic file - Monday, November 11, 2019, 21:00 (sharp).

YOU SHOULD NOT USE ";" , "!" AND "–>" IN YOUR PROLOG RULES.

You must work in groups of TWO, or THREE. You cannot work alone. 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 will exercise what you have learned about natural language processing. In this assignment, we imagine a robot that is capable of picking up and moving blocks located on a table in front of it. Ultimately, we would like to be able to tell the robot what to do using ordinary English imperative sentences like "Pick up the pyramid beside the small green cube and put it directly in front of the yellow block". As in class, to interpret English sentences like these, we need to construct a *database* of facts about our world, a *lexicon* with all of our English vocabulary, and a *parser/interpreter* to process the English expressions. As in class, we will focus mainly on the English noun phrases like "the pyramid beside the small green cube."

Before we start using English words, we will build in Prolog a simple deductive database of facts about this blocks world and rules that define semantics of simple spatial relations. Each block can be assumed to have a colour, a size, and be of a certain shape (a cube, a pyramid, a wedge, ...). Also, each block is located somewhere in the scene. To simplify matters, we imagine that our robot lives in a two-dimensional world with these properties:

- a block is located either on the table or on another block;
- each block is on at most one other block (which must be a cube), and has at most one other block on it (i.e., nothing can be on top of a wedge or on top of a pyramid, but there can be towers of cubes with any block on the top);
- the surface of the table is divided into some number (say, 5 or 6) of contiguous areas ordered from left to right;
- each block on the table is located in exactly one of these areas (i.e., blocks cannot be on the borders between areas), and each table area contains at most one block (or at most one pile of blocks).

With these assumptions, the locations of blocks can then be represented using a predicate locatedOn(*Block*,*Loc*) where *Block* is a block and *Loc* is either a cube or one of the table areas. Notice a block cannot be located on a wedge or on a pyramid. We can then use a second predicate justLeftOf($Area_1$,$Area_2$) to state that $Area_1$ is the area on the table immediately to the left of $Area_2$.

1. Implement a database that expresses basic facts about 10-15 blocks, and colours, sizes, shapes, and locations of all blocks in a scene.
2. Write rules defining the predicates beside(X, Y), which holds when blocks X and Y are both on the table in two adjacent areas, and above(X, Y), which holds when block X is somewhere above block Y. Implement also the predicate leftOf(X, Y), which is true is X is somewhere to the left from

Y. Note that X can be higher or lower than Y and blocks X and Y can be in piles of blocks which are not next to each other. Write also a rule defining the predicate rightOf(X, Y) that is opposite to the predicate leftOf(X, Y).

3. Show that your database works properly by formulating some Prolog queries (similar to the 1st assignment) about the scene and obtaining suitable answers. These queries should *not* use English noun phrases! However, you must use as queries the Prolog versions of your own noun phrases that you will be using to test your program in Part 6 of this assignment. The purpose of this is to show to the TA what answers should be expected from your database when it will be subsequently queried using English noun phrases. Keep your queries and answers computed by Prolog in the file **nlu.txt**

Once you have your database working, you are ready to consider English noun phrases and the blocks they refer to in your scene. Here are some example queries using English noun phrases that we would like to be able to answer:

- what([a,huge,blue,wedge,on,the,table], B).
- what([any,small,green,block], B).
- what([any,yellow,pyramid,on,a,big,cube], B).
- what([a,cube,beside,the,orange,wedge], B).
- what([any,cube,below,a,pink,wedge,on,a,large,red,block], B).
  Note that this is ambiguous: is it the cube or the wedge that is on the red block?
- what([a,green,wedge,above,a,blue,block,beside,the,small,red,pyramid], B).
- what([the,pink,medium,wedge,above,a,block,beside,a,medium,blue,cube], B).

4. Build a Prolog lexicon, as we did in class, of articles, adjectives, common nouns, and prepositions, including **all the words** in the seven example noun phrases above. Your lexicon should include all well-known colors, shapes, sizes, static spatial relations like those considered above (in total, **20 words or more**, apart from articles). The word "any" should be treated as an article. There are no proper nouns since blocks usually do not have English names. Remember that it is easy to defeat a language understanding program by using a word that it does not know about. Vocabulary is important in these systems. Make yours as smart as you can. Keep both the database and your lexicon in the file **nlu.pl**

5. Copy the Prolog parser/interpreter for noun phrases given in class (or write your own), and define the what predicate used above. The parser must be also in the same file **nlu.pl**

6. Test the what predicate on a variety of noun phrases, like those above, showing that it is capable of identifying the blocks being referred to in your scene. It is up to you to choose noun phrases for testing, but you must convincingly demonstrate that your program works properly. Try at least **10 new** different noun phrases (in addition to the phrases given to you). Remember that testing your program is very important part of the software development cycle. You lose marks if you do not test your program as required. Copy all results of your tests into **nlu.txt** Copy all results of your tests into **nlu.txt**

7. Modify your parser to handle the article "the" properly. The idea here is that a noun phrase like "the red pyramid" should only succeed in naming a block if there is a *unique* block of the appropriate kind (i.e., if there are two red pyramids, then the query with a noun phrase like "the red pyramid" should fail). Examples:

   what([the,pink,wedge,above,a,red,cube], B).
   what([the,yellow,pyramid,above,a,green,block], B).

8. Handle prepositional phrases of the form "between X and Y" as in

   - ?- what([a,medium,yellow,pyramid,between,an,orange,wedge,and,a,green,wedge], B).
   - ?- what([a,green,cube,between,the,large,blue,block,and,the,medium,blue,cube], B).
   - ?- what([a,pink,block,between,an,orange,wedge,and,a,red,pyramid], B).

   You can assume that in all prepositional phrases "between X and Y" about block B, blocks X, B, Y are arranged in a scene from left to right. For example, in the 1st query given above, "an orange wedge" is located to the left from "a medium yellow pyramid", and this pyramid is to the left from "a green wedge". You must also demonstrate that your program works properly. Include your testing session in the file **nlu.txt**

Note that the TA who will be testing your program can submit other Prolog and natural language queries: Provide a hand-drawing of the scene encoded in your database to make sure that TA will understand correctly what answers he can expect to get from his queries. Save an image of your drawing as a PDF file **scene.pdf** and submit it together with other files of your assignment.

Handing in solutions. (a) An electronic copy of your program (**nlu.pl**) with all defined predicates (you also must provide brief comments); (b) an electronic copy (**nlu.txt**) of your queries and answers computed by Prolog (provide your drawing of your scene in this file); (c) A copy of your drawing **scene.pdf**

Bonus work (0-40 points):

To make up for a grade on another assignment that was not what you had hoped for, or simply because you find this area of Artificial Intelligence interesting, you may choose to do extra work on this assignment. Bonus marks will be given at the discretion of the TA: minor variations will not be awarded any extra marks. Demonstrate your creativity to get bonus marks! *Do not attempt any bonus work until the regular part of your assignment is complete. Bonus work is individual.*

The English noun phrases described above are quite limited. For bonus work, generalize your program to handle some additional features of English and add words to your lexicon. The more advanced your program will be, the higher will be your mark. You must test your program extensively. Your elaborated program should handle at least the following:

- Handle *prepositional idioms*, that is, groups of words that behave like prepositions, as in, "next to", "to the left of".
- Generalize the location property to the three-dimensional world. In this case, areas on the table should be ordered from left to right *and* from front to back. The lexicon should then be augmented with new prepositions and prepositional idioms to reflect the richer notion of location.

Handing in solutions. An electronic copy of: (a) your working program **bonus.pl** (you also must provide brief comments in the program); (b) your session with Prolog, showing the queries you submitted and the answers returned (the name of the file must be **bonus.txt**). If you work in a group, write the name of the person who submits bonus work. Otherwise, bonus marks (if any) will be divided between all students in your group.

How to submit this assignment. Read regularly *Frequently Answered Questions* and replies to them that are linked from the Assignments 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 either `nlu.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 nlu.pl nlu.txt scene.pdf
```

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 participated in discussions (see the course management form). You may be penalized for not doing so. Second, upload your file **yourLoginName.zip**
(make sure it includes **all** files) to D2L into "Assignment 4" folder.

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.

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 and 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 the last file you submit will determine whether you have submitted your assignment on time.