Starting with Assignment 5

Step 1: Develop your solutions locally:

Include the CLP(FD) library as follows in your program:

```
    :- use_module( library(clpfd) ) .
```

Assignments 5.1., …, 5.6. are consecutive assignments which all together shall implement the domain of verbs modelling change of ownership.

The assignment requires to combine procedures of different example programs, and to re-implement some procedures and some grammar rules, i.e.

- move
- setVal
- getVal
- getAnswer
- sentence
- question

while other procedures shall remain unchanged, i.e., dialog, tellask, sORq, and syntaxError, etc..

Your complete dialog system should work correctly, although the tester5 will check the procedures that shall be reimplemented by you. Therefore, your procedures must have the same names and numbers of parameters as required in the file a5.pl.

For each assignment 5.X. there are done some local tests that check the procedures that you shall reimplement (e.g. sentence, move, question, getAnswer, …).

Like with Assignments 2, 3, and 4, you shall first solve the queries locally by using the **SWI-Prolog-Interpreter -**

either by double-clicking the file a5.pl (or by opening it from command line, i.e. using

swipl-win -s a5.pl in **Windows**

or using by using

./swipl -s a5.pl in **MacOS** or **Linux**)

and the **SWI-Prolog-Editor .**

Step 2: Complete the assignments before you upload your solutions:

Upload your solution files not earlier than when you have done all assignments and commented your solutions as described below.

Before you upload your solutions, do a double-check using the following tester5 program.
As before, the version of the test program differs depending on the operating system:

1. for Windows use **tester5**.exe
2. for MacOS use **tester5**
3. for OpenSuse Linux use **tester5suse**

1. Assuming that your program is called **a5.pl** and that you want to check your solution to Assignment 5.**1** (i.e. sentences with 'owns'), in **Windows** use a CMD-Shell to call:

```
> tester5   a5   1                    for testing your procedures
```

   (here, you shall use the Windows version tester**5**.exe)

2. Assuming that your program is called **a5.pl** and that you want to check your solution to Assignment 5.**1** (i.e. sentences with 'owns'), in **MacOS** use a shell to call:

```
> ./tester5   a5   1                  for testing your procedures
```

   (here, you shall use the MacOS version tester**5**)

3. Assuming that your program is called **a5.pl** and that you want to check your solution to Assignment 5.**1** (i.e. sentences with 'owns'), in **OpenSuse Linux** use a shell to call:

```
> ./tester5suse   a5  1               for testing your procedures
```

   (here, you shall use the OpenSuse Linux version tester**5**suse)

Actions to be taken depending on the feedback of the program tester5:

If you get an error message query 5.X is undefined, make sure that you have named your procedures like in Assignment 5.X, i.e. sentence, move, …, and that the number of parameters is correct. Consider also the different numbers of parameters needed for DCG rules.

If the program tester5 tells you that your solution is different from the wanted solution, look at the counter example presented to show the difference - and rewrite your query.
In case of counter examples for sentence or question, most likely the error is in the grammar. May be, you have to do this multiple times until you have a correct solution.

If the program tester5 tells you that your solution is most likely equivalent to the wanted solution, but it takes more clauses and/or more goals, try to improve your solution (i.e. find a shorter solution). May be, you have to do this multiple times until you have a short solution.
Hint: Try to find a correct solution in the first place. Once, your solution is equivalent to the wanted solution, but not short enough, try inlining (→ Video 1.10.) to get a shorter solution. The same inlining techniques can also be considered for grammar rules.
(If your solution is just a few goals longer than the wanted solution tester5 does not mention this.)

If program tester5 tells you that your solution is most likely equivalent to the wanted solution, the tester could not detect any difference in the correctness of your program (and could not detect a big difference in the complexity of the solutions). Then, you should prepare for explaining your solution.
Please do not upload your solution, before you get the answer your solution is most likely equivalent to the wanted solution **from the tester for each of the 6 checks**
> tester5 a5 N   with N in {1,2,3,4,5,6} .

You are requested to comment the following procedures and grammar rules in your code:

- getAnswer (approximately 2-3 lines) ,
- move (approximately 7-10 lines for the complete procedure, i.e. for all rules in total)
- getVal (approximately 2-3 lines) ,
- sentence (approximately 2 lines)
- question (approximately 2 lines)

Furthermore, use 8-10 lines to describe the overall behavior of the program, i.e.

- what are the strong points of this dialog program?
- what are weak points or shortcomings of this dialog program?

After commenting your solution and describing the overall behavior of the program, please check that it still compiles without warning (i.e. that it does NOT generate UTF-code warning messages)!

You can do the summarized test by one of the following commands depending on your operating system:

```
> tester5   a5   -1                    in Windows
> ./tester5   a5   -1                  in MacOS
> ./tester5suse   a5   -1              in OpenSuse Linux
```

Then, you can upload your solution.