Starting with Assignment 6

Step 1: Develop your solutions locally:

Include the CLP(FD) library as follows in your program:
:- use_module(library(clpfd)) .

Assignments 6.1. is completely independent of the assignments 6.2. and 6.3. which belong together. The assignments cover different aspects of programming in a domain specific language.

Assignment 6.1. covers programming in (a small subset of) mathematics.

Assignments 6.2 and 6.3. cover programming in (a small subset of) English. These assignments require that you extend a program (p5.8._tell_an_ask_more.pl) by your own (small!) grammar for sentences and for questions and by your own procedure getAnswer/3.
Your complete dialog system should work correctly, although the tester6 will check the grammar rules and the procedure that shall be reimplemented by you. Therefore, your procedure and grammar rules must have the same names and numbers of parameters as required in the file a6.pl.

Like with Assignments 2, 3, 4, and 5, you shall first solve the queries locally by using the **SWI-Prolog-Interpreter -**
either by double-clicking the file a6.pl (or by opening it from command line, i.e. using
swipl-win -s a6.pl in **Windows**
or using by using
./swipl -s a6.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 the assignments and commented your solutions as described below.

Before you upload your solutions, do a double-check using the following tester6 program.
As before, the version of the test program differs depending on the operating system:

1. for Windows use **tester6**.exe
2. for MacOS use **tester6**
3. for OpenSuse Linux use **tester6suse**

1. Assuming that your program is called **a6.pl** and that you want to check your solution to Assignment 6.**1** (i.e. isolation of variables), in **Windows** use a CMD-Shell to call:
   > tester**6** a**6** **1** for testing your procedures
   (here, you shall use the Windows version tester**6**.exe)

2. Assuming that your program is called **a6.pl** and that you want to check your solution to Assignment 6.**1** (i.e. isolation of variables), in **MacOS** use a shell to call:
   > **./**tester**6** a**6** **1** for testing your procedures

(here, you shall use the MacOS version tester**6**)

3. Assuming that your program is called **a6.pl** and that you want to check your solution to Assignment 6.**1** (i.e. isolation of variables), in **OpenSuse Linux** use a shell to call:
   > **./**tester**6suse** a**6 1** for testing your procedures

(here, you shall use the OpenSuse Linux version tester**6**suse)

Actions to be taken depending on the feedback of the program tester6:

If you get an error message query 6.X is undefined, make sure that you have named your procedures like in Assignment 6.X, i.e. isolate_final, sentence, …, and that the number of parameters is correct. Consider also the different numbers of parameters needed for DCG rules.

If the program tester6 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.
May be, you have to do this step multiple times until you have a correct solution.

If the program tester6 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 step 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 tester6 does not mention this.)

If program tester6 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 3 checks**
> tester6 a6 N with N in {1,2,3} .

Step 3: Comments required before you upload your solutions:

Y**ou are requested to comment the following for Assignment 6.1.:**

1. What are the main differences between your solution to Assignment 6.1 and the demo programs for rewrite rule systems, and what are these differences good for? (approximately 3-4 lines)

2. Which trick(s) allowed you to keep the rewrite rule system small, i.e. to contain only 12 rewrite rules? (approximately 2-3 lines)

Furthermore, for the rewrite rule system implemented with your Assignment 6.1., describe the overall behavior of the program (approximately 12-18 lines), i.e.

1. What are limitations of this rewrite rule system?
2. How could this program be extended easily to a more powerful or more useful program? Sketch your ideas how this could be done (without programming it).

And you are requested to comment the following for Assignment 6.2. and 6.3.:

How does your procedure getAnswer/3 provide answers that can only be deduced from a combination of multiple input sentences (approximately 5-7 lines).

Furthermore, for the dialog system implemented with your Assignment 6.2. and 6.3., describe the overall behavior of the program (approximately 15-20 lines), i.e.

1. What are limitations this dialog program?
2. In which aspects and to which application or domain could this program be extended easily? Sketch your ideas how this could be done (without programming it).

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) by a call

> **./**tester**6** a**6** -**1** for testing all your procedures

Then, you can upload your solution.