Installation of SWI-Prolog – Step 3 and 4 – and starting with the assignments

Step 3: Starting with the assignments:

Make sure that you have done installation Step 1 and Step 2 (documented in a previous document) successfully, i.e., that you know how start and how to halt SWI-Prolog, how to call the built-in editor, how to compile files, how to submit queries, and how to see more answers of queries. *If you could not get a local installation running, then remotely use the Linux Pool at Paderborn university to do Step 1 and Step 2 (download an extra document how to do that)*.

1. Then, download the .zip file **assignment1.zip** containing the file a1.pl
2. Open the file by calling the **SWI-Prolog-Interpreter -**
   either by double-clicking the file a1.pl (or by opening it from command line, i.e.
   using **swipl-win** a1.pl in **Windows**
   or using **./swipl** a1.pl in **MacOS** or **Linux**)

3. In the **SWI-Prolog-Interpreter-Window** type
   ?- edit. RETURN
   to call the editor.

4. Then, in the **SWI-Prolog-Editor,** *remove* the comment symbol "%" before q1(Player) :- …
   and save and compile the file.

5. Then, in the **SWI-Prolog-Interpreter-Window** type in the query
   ?- q1(Player). RETURN
   and check that you get all results i.e., use ";" (semicolon) after you have seen the first solution to see more (or all) solutions, in the SWI-Prolog- Interpreter-Window.
6. Thereafter switch to the **SWI-Prolog-Editor-Window** to type in your solution rule for the query q2. Again, save and compile the file, switch to the **SWI-Prolog-Interpreter-Window** type in the query
   ?- q2(Player,Clubname). RETURN

7. If your query qN does not show the desired results, switch to the **SWI-Prolog-Editor-Window** to edit the rule for qN, compile as described above, and test qN again.

8. Once, qN is answered correctly, continue with qN+1.

9. **To leave the SWI-Prolog Interpreter** , you type in ?- halt . RETURN

How to deal with typical warnings

How to react to warnings "singleton variables …":

A singleton variable is a variable occurs only once in a clause – either because of a typo or because it could (and should) be replaced by an anonymous variable "_".

→ if it is a typo, remove the typo. Otherwise, use "_" instead of a variable name.

How to treat warnings "redefining clauses …":

You can either compile your code again or ignore this warning

Step 4: Completing the assignments before you upload your solutions:

Be prepared that you can explain your solutions. Currently, it is not clear, which form of a presentation is possible. Nevertheless, you shall upload your solutions in time (how this shall be done will be announced later, when the assignments are published).
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 test program.
The version of the test program differs depending on the operating system:

1. for Windows use tester1.exe
2. for MacOS use tester1
3. for OpenSuse Linux use tester1suse

1. Assuming that your program is called **a1.pl** and that you want to check your solution to query **q13**, in **Windows** use a CMD-Shell to call:

```
> tester1  a1  0
> tester1  a1  13
```

(here, you shall use the Windows version tester1.exe)

2. Assuming that your program is called **a1.pl** and that you want to check your solution to query **q13**, in **MacOS** use a shell to call:

```
> ./tester1  a1  0
> ./tester1  a1  13
```

(here, you shall use the MacOS version tester1)

3. Assuming that your program is called **a1.pl** and that you want to check your solution to query **q13**, in **OpenSuse Linux** use a shell to call:

```
> ./tester1suse  a1  0
> ./tester1suse  a1  13
```

(here, you shall use the OpenSuse Linux version tester1suse)

4. Hint: The call of `> ./tester1suse a1 0` is only useful to check whether you installed the correct version of SWI-Prolog

Skip this section if tester1 works locally: How to use tester1 in the Linux pool at the University

Only if using the tester1 locally fails, there is the following further option.
Try this only, if you cannot use the tester1 locally , although you can run swipl (anywhere) !

The OpenSUSE version of the tester1 can be used on the Linux machines at Paderborn University.
In order to use the tester1 on these machines, you have to do the following:

1. Do a wakeup call to wakeup a computer in the Linux pool of Paderborn University (as described in the document for using the Linux pool machines of Paderborn University for SWI-Prolog).

2. copy your solution file (e.g. a1.pl) and the file tester1suse to a directory on the Linux machines at Paderborn University.
   You can choose any directory, but let us say, you create a new directory Prolog/a1

3. Do a remote login via ssh:
   ```
   my_CDM_shell > ssh sshgate.cs.uni-paderborn.de
   sshgate2% ssh arthur.cs.uni-paderborn.de
   ```

4. Change directory to your newly created directory
   ```
   arthur% cd Prolog/a1
   ```

5. Check whether the tester1 works there
   ```
   arthur% ./tester1suse a1 0
   ```
   you should get as output:
   ```
   You have installed the desired SWI-Prolog version 8.1.26
   ```

6. Check your program solutions with the tester1suse, e.g. for query q2 in program a1.pl
   ```
   arthur% ./tester1suse a1 2
   ```
   you should get as output:
   ```
   Your solution to query q2 is most likely equivalent to the wanted solution.
   ```

If you get a different answer, look at the next section
(Actions to be taken depending on the feedback of the program tester1).

Important for all users: Actions to be taken depending on the feedback of the program tester1:

If the tester1 program returns "… FATAL ERROR … ",
double check that you have installed SWI-Prolog version **8.1.26-1**
and that you have used the correct operating system dependent version.

If you get an error message query q1/1 is undefined, make sure that you have named your queries q1, q2, … and that the number of parameters is correct (e.g. the queries q1,q3, and others must have one parameter, but q2 and others must have 2 parameters), and of course that you have removed the comment (%) before the q1(Player) :- … rule.

If the program tester1 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 multiple times, until you have a correct solution.

If the program tester1 tells you that your solution is 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.

If program tester1 tells you that your solution is most likely correct, the tester could not detect any difference. Then you should prepare for explaining your solution.
As long as oral presentations are not feasible,
you are requested to write a short (3-5 lines) comment about the programming techniques used in your solution below your code as part of the file that you upload. Your comment shall refer to the videos provided for the lecture. For example:

```
q1( Player ) :- book( Player , _ , P ) , P =< 20 .

% I am using the generate and test approach as shown in Video 1.7
% The relation book is the correct generator because it generates each book exactly once.
% The test checks that the price, i.e. the 3rd argument of the book relation, is not greater 20.
% The return argument in the rule head is Player, as only Player shall be returned as output.
% As the 2nd argument of the book relation is not needed, I used _ .
```

Note that for the subtasks 25 to 27, you do NOT need to program a query q25(…) rule, but a state procedure, e.g. state(25,…) for subtask 25.
If you get a message starting with "not OK : …" then your database state does not fulfill the requirements and you are requested to change the database state and try again.

Finally, before you upload your solution to PANDA:

1. Doublecheck that you have commented all your solution queries.
2. Thereafter (!), doublecheck that your program still compiles correctly.
3. Doublecheck, that you have solved all queries as required by typing

```
> tester1  a1  -1          ( on Windows )
> ./tester1  a1  -1        ( on MacOS )
> ./tester1suse  a1  -1    ( on Linux in the University Pool )
```

and you get messages *your solution is most likely correct* for queries 1 to 24
and you get *only* messages of the following kind for subtasks 25 to 27:

"oK: Your database state to subtask 27 shows that a solution to query …
returns a result that is not returned by a solution to query …".