MSc Computer Science – 1st Assessed Prolog Lab Exercise
Issued: 16th November
Due: 30th December

This Lab Exercise is concerned with the new MSc degree in Magic at the Hogwarts School of Witchcraft and Wizardry. You will be responsible for writing a program that helps to find out more information about the structure of this year's MSc degree. This involves for example the students' choices on magical courses, which courses are taught by which witches/wizards, and which students are sharing courses.
To help you with this exercise, a file **hogwarts.pl** is available on Gitlab. The file contains some facts about the students and teachers at Hogwarts, as well as course options for students doing the MSc degree in Magic:

student(SID, SN, House)	**SID** is the unique magical ID of a student with name **SN**, which is part of a **House** (*gryffindor*, *hufflepuff*, *ravenclaw*, or *slytherin*)
teacher(TID, TN)	**TID** is the unique magical ID of a teacher with name **TN**
compCourse(SCN, CN, TID)	**SCN** is the short course name of the compulsory course **CN** taught by the wizard/witch whose ID is **TID**
optCourse(SCN, CN, TID)	**SCN** is the short course name of the optional course **CN** taught by the wizard/witch whose ID is **TID**

Of course, the four houses gryffindor, hufflepuff, ravenclaw, and slytherin stay the same (names) every year, but the students changes evyer year. Furthermore, teachers and courses can change over the year. You should consider this in your program, i.e. it should still work in future years.
Extend the supplied program about the Hogwarts School of Witchcraft and Wizardry by adding the following facts to your copy of **hogwarts.pl**:

1.) Add yourself as a student of the Hogwarts School of Witchcraft and Wizardry using the predicate **student(SID, SN, House)**. Make sure that your name **SN** and your ID **SID** are different from the ones of the predefined students. It is completely up to you which house you would like to be in (no need to ask the sorting hat).

2.) Define facts **enrolled_opt(SID, SCN)** expressing which 3 optional courses each student takes (it's completely your choice which courses the students take), where **SID** is the student's magical ID and **SCN** the short name of a course.
Make sure that you define 6 instead of 3 optional courses for Hermione Granger!

Now further extend the program by adding rules defining the following predicates. Note that once you have defined a predicate, it can be used in the definition of the following predicates.
As in the Prolog manual, a "**?**" in front of a predicate's argument means that this argument might or might not be given when querying the predicate. A "**-**" in front of a predicate's argument means that this argument is always a variable when querying the predicate. Don't write these "?" and "-" in your code, they are just a way of telling you how your predicates can be queried.

3.) **enrolled(?SID, ?SCN)**
The student whose ID is **SID** is enrolled on the course with short name **SCN** (this can be a compulsory or an optional course).

4.) **teaches(?TN, ?SCN)**
The wizard/witch whose name is **TN** (not the ID!) teaches the magical course with short name **SCN**.

5.) **taughtBy(?SN, ?TN)**
The student **SN** is enrolled on a magical course that is taught by **TN**, where **TN** is the teacher's name and **SN** the student's name.
In the case that a student name x is taught by a teacher y on two courses and you query "taughtBy(SN,TN)", it is completely normal that you get "SN = x, TN = y" twice as a solution.

6.) **takesOption(?SN, ?CN)**
The student **SN** is enrolled on the optional course called **CN**.

7.) **takesAllOptions(?SN, ?OptCourses)**
OptCourses is the list of all optional magical courses the student called **SN** chose. The list **OptCourses** contains the names of these magical courses in alphabetical order.
If you query an invalid student name, Prolog's answer should be "no". You can assume that every valid student takes optional courses (so OptCourses will never be the empty list).

8.) **studentsInHouse(?House, ?Students)**
Students is the list of all student names (!) which are part of a particular **House**. The list is ordered alphabetically by the respective students' ID (!). This means that you have to order the list **Students** by ID, even though it contains only the respective names.
House is one of *gryffindor*, *hufflepuff*, *ravenclaw*, or *slytherin*. For an invalid **House**, Prolog's answer should be "no". If a valid **House** does not have any students, then **Students** should be the empty list.

9.) **studentsOnCourse(?SCN, ?CN, ?StudentsByHouse)**
For a magical course **CN** with short course name **SCN** (this can be a compulsary or an optional course), the list **StudentsByHouse** has 4 elements of the form **House-Students**, one for each house. **House** is one of *gryffindor*, *hufflepuff*, *ravenclaw*, or *slytherin*, and **Students** is a list consisting of all students (their names) in the respective **House** enrolled on the course **CN/SCN**:
studentsOnCourse(SCN, CN, [gryffindor-GryffindorList, hufflepuff-HufflepuffList, ravenclaw-RavenclawList, slytherin-SlytherinList])
The order of student names in the list **Students** does not matter. If no student of a certain house takes the course, then the respective list of **Students** is the empty list. This means, that if no students at all are taking a course with name *cn* and short name *scn* and you query *StudentsOnCourse(scn, cn, StudentsByHouse)*, the solution should be

StudentsByHouse = [gryffindor-[], hufflepuff-[], ravenclaw-[], slytherin-[]]. However, if you are querying a non-existing course, the answer should be "no".

10.) **sharedCourse(?SN1, ?SN2, ?CN)**
The optional magical course with name **CN** is taken by two different students with names **SN1** and **SN2**.
It is normal that your sharedCourse succeeds for/generates both sharedCourse(name1, name2, cn) and sharedCourse(name2, name1, cn) for any two people name1 and name2 who share the course named cn.

11.) **sameOptions(?SN1, ?SN2, ?Courses)**
Two differerent students **SN1** and **SN2** are enrolled on exactly the same three optional courses, which form the list **Courses** of course names (the order does not matter). Since Hermione is taking 6 instead of 3 optional courses, we will say that she has the same courses as another student if 3 out of her 6 magical courses match the other student's courses.
Again, it is normal that your predicate succeeds for/generates both sameOptions(name1, name2, courselist) and sameOptions(name2, name1, courselist).

Some further advice and information about the marking scheme:

- 85% of the marks are allocated for your solutions to questions 1.) - 11.). Most of them are allocated by an auto-test.
  This means it is very important that you **define predicates exactly as specified** (with respect to spelling and number of arguments).
  Of course, you can/should define your own **auxiliary predicates** in order to reduce the size of predicate definitions.
- 15% of the marks are allocated to programming style, including for example:
  - Clear and helpful **commenting**.
  - No long predicate definitions. In case a clause gets too long, try to split it into auxiliary predicates.
  - Readable code: indentation, sensible naming etc.

- You **won't need any Prolog libraries** for this exercise! However, you can of course use built-in predicates.
- Try to use predicates you have already defined as well as recursion, rather than solving everything with setof and findall (you can of course use setof and findall where appropriate, just try to not use it too often).
- **Test** all your predicates using appropriate queries.
- Your solutions will be tested using **Sicstus Prolog** under Linux, so make sure that you use Sicstus (not SWI or any other Prolog) and that your code runs on the Lab Machines.

Submit your extended program as a file named **hogwarts.pl** via the usual procedure (Gitlab, LabTS, CATE).