German University in Cairo
Department of Computer Science
Dr. Yomna M.I. Hassan

Concepts of Programming Languages, Spring Term 2025
Project 1: Class Scheduling Using Prolog
Due: 11th *April 2025*

1. Project Description. In this project, you are going to implement a Prolog program to assist the students in scheduling timings of the courses they are studying. The program will assist in the creation of the schedule, making sure there are no clashes with other courses taken by the student, allowing each student 2 days off. It will also help in finding a slot on which the course is taught.

It is noted that a knowledge base is provided with the project document for you to work on. The knowledge base is included in a predicate `studies(Student, Course)` which says that `student` studies `Course`. There might be multiple facts for the same student for the multiple courses they study. The other predicate `day_schedule(Day, Schedule)` shows the schedule of every week day as a list of 5 lists representing the courses taught in each slot.

inside a pl file called `studentKB`. To consult a knowledge based from a different file, all you need to do is to place the KB and your project in the same directory and add this line to the top of your project file:

```
:- consult('studentKB').
```

Below are the main constraints that need to be satisfied in any schedule:

- Each student should not have clashes in the schedule between any two courses that he/she is studying.
- Each student should not have slots allocated on more than a predefined number of days per week (to enforce the constraint of giving each student 2 days off).

2. Required Predicates. Your implementation must contain the five below predicates. Read the description of all of them before you start your implementation.

Note that:

1. You can add any other helper predicates you need.
2. You can use any predefined predicates **except for** `assert` and `retract`.
3. It is easier to implement the predicates in the reverse order of how they are listed below.
4 Some of the predicates required might use other required predicates.

a) `university_schedule(S).`
such that:

- The predicate gets schedules for all students and binds `S` to a list containing the schedules of all students registered in the university.
  An element of the list is a schedule of one student and each schedule is a structure of the format `sched(student_id, Slots)`, where `student_id` is the id of a student and `Slots` is a list containing all the slots that appear in the schedule of the student associated with `student_id`. An element of the list `Slots` is a structure, representing one slot, of the format `slot(day,slot_number,course_code)`.
  Example Query and Response:

```
?- university_schedule(S).
        S = [sched(student_0, [slot(saturday, 2, csen202),
        slot(sunday, 2, mech202), slot(monday, 2, physics201)]),
        sched(student_1, [slot(saturday, 2, physics201)]),
        sched(student_2, [slot(saturday, 2, csen202),
        slot(saturday, 3, signls402)]),
        sched(student_3, [slot(wednesday, 3, elct402),
        slot(saturday, 3, netw201), slot(tuesday, 1, signals402)])]
```

b) `student_schedule(Student_id, Slots).`
such that:

- The predicate binds `Slots` to the list of slots scheduled for the student whose id is bound to `Student_id`
  It should be noted that an element of the list `Slots` is a structure, representing one slot, of the format `slot(day,slot_number,course_code)`, which is the same format as the structure `slot` used in the predicate, `university_schedule(S)`, explained in part a).
  Example Query and Response:

```
?- student_schedule(student_0, X).
    X = [slot(saturday, 2, csen202), slot(sunday, 2, mech202),
    slot(monday, 2, physics201)] ;
    X = [slot(saturday, 2, csen202), slot(sunday, 2, mech202),
    slot(tuesday, 2, physics201)] ;
    X = [slot(saturday, 2, csen202), slot(sunday, 2, mech202),
    slot(wednesday, 3, physics201)] ;
...
```

c) `no_clashes(Slots).`
such that:

- The predicate succeeds when `Slots` is a list of slot structures, where no two slots are clashing with each other.

  It is noted that the schedule for each student is same format as the structure `slot` used in above mentioned predicates! The format of the structure is `slot(day,slot_number,course_code)`.
  Example Query and Response:

```
?- no_clashes([slot(saturday, 1, physics201),
        slot(sunday, 1, mech202), slot(tuesday, 1, csen202)]).
```

```
 true

 ?- no_clashes([slot(saturday, 1, physics201),
slot(sunday, 1, mech202), slot(saturday, 1, csen202)]).
 false.
```

d) `study_days(Slots, DayCount).`
such that:

- The predicate checks that in the list of slot structures, `Slots`, there are no slots on more days than `DayCount` days of the week.

  Example Query and Response:

```
 ?- study_days([slot(saturday, 1, physics201), slot(sunday, 1, mech202),
slot(tuesday, 1, csen202)], 2).
 false.

 ?- study_days([slot(saturday, 1, physics201), slot(sunday, 1, mech202),
slot(tuesday, 1, csen202)], 4).
 true.
```

e) `assembly_hours(Schedules, AH).`
such that:

- The predicate retrieves a list of all slots where all students are free and on their study days and binds this list to `AH`. Its arguments are `Schedules`, which is a list of student schedules as structures, described in part a), `AH` which is a list of common free slots for all students that are also not on their days off. A slot which is a member of the list `AH` is defined using the structure `slot(Day, SlotNumber)`.

  Example Query and Response (Given previous Schedule S):

```
 ?- assembly_hours([sched(student_0, [slot(saturday, 1, csen202),
        slot(sunday, 1, mech202), slot(monday, 1, physics201)]),
        sched(student_1, [slot(saturday, 1, physics201)]),
        sched(student_2, [slot(saturday, 1, csen202),
        slot(saturday, 2, signls402)]),
        sched(student_3, [slot(wednesday, 2, elct402),
        slot(saturday, 2, netw201), slot(tuesday, 0, signals402)])], AH).
AH = [slot(saturday, 3), slot(saturday, 4), slot(saturday, 5)]
```

3. Teams. You are allowed to work in teams of four members. You must stick to the teams submitted in the team submission form. IDs for the submitted teams will be posted on the CMS.

4. Deliverables. You should submit a single `.pl` file named with your team ID. The submission link will be posted on the CMS prior to the submission.