CS 381 – Homework 7.

This is an individual assignment. Put all the definitions into one file named hw7.pl. This file should also contain the database facts and comments including your name. Do not include any extraneous entries for the predicates.

Problem 1. Database Application

Consider a database about courses, sections, times, classrooms and student enrollment, given in the form of Prolog facts below. The entire database in in hw7.pl.

- course(course_number, course_name, credits)
- section(CRN, course_number)
- place(CRN, building, time)
- enroll(sid, CRN)
- student(sid, student_name, major)

Define the following derived Prolog predicates by one or more rules. Note that the shown goals are just examples. You should define the predicates so that it is possible to formulate goals with variables or constants at any argument position.

a) Define a predicate schedule/4 that gives for a student (by sid) the course name, building and time of the classes the student is taking. For example if you evaluate the schedule(122,C,B,T), Prolog should give the following results.

```
?- schedule(122,C,B,T).
C = python,
B = owen102,
T = 10 ;
C = calculusI,
B = kec1001,
T = 13.
```

b) Define a predicate schedule/3 that gives for a student (by sid), the students name and the names of the courses the student is taking. For example if you evaluate the schedule(122,N,C), Prolog should give the following results.

```
?- schedule(122,N,C).
N = mary,
C = python ;
N = mary,
C = calculusI.
```

c) Define a predicate offer/4 that gives, the course number, course name, CRNs for sections and times the sections of the course are offered. For example, the goal offer(mth210,N,C,T) should yield the following result.

```
?- offer(mth210,N,C,T).
N = calculusI,
C = 2322,
T = 11 ;
N = calculusI,
C = 8522,
T = 13 ;
N = calculusI,
C = 7822,
T = 14.
```

d) Define a predicate conflict/3 that can compute conflicts in a student's schedule. A conflict exists if a student is enrolled in two classes that are scheduled to meet at the same time. The arguments of conflict are a student's sid and two CRNs for sections of classes. If the students is not enrolled in the section then a false is returned.

```
?- conflict(410,X,Y).
X = 1621,
Y = 7822 ;
X = 7822,
Y = 1621 ;
false.

?- conflict(122,X,Y).
false.

?- conflict(122, 2421,Y).
false.
```

e) Define a predicate meet/2 that can determine pairs of students that can meet in a classroom by either attending the same class or by having classes that are back to back in the same classroom. Meet will take a two student sids as arguments. For example,

```
?- meet(122,SID).
SID = 150 ;
SID = 300 ;
SID = 310 ;
SID = 212 ;
SID = 175 ;
SID = 410 ;
false.
```

f) Define a predicate roster/3 that produces a list of all students talking a section of a course. The arguments to roster are roster(CRN, Student_name).

```
?- roster(3522,Sname).
Sname = pat ;
Sname = amy ;
Sname = zoe.
```

g) Define a predicate highCredits/1 that produces a list of courses that are 4 or more credits. The argument to highCredits is a course_name. For example,

```
?- highCredits(Cname).
Cname = calculusI ;
Cname = data_structures ;
Cname = algorithms ;
false.
```

Problem 2: List Predicates and Arthmetic

Note: Do not use the predefined flatten and nth. You are allowed to use predefined predicates append and member.

a) Define a Prolog predicate rdup(L,M) to remove duplicates from an ordered list L. The resulting list should be bound to M. Note that M must contain each element of L exactly once and in the same order as in L. You can assume that L is an ordered list. Some examples are given below:

```
?- rdup([3,4,4,5],M).
M = [3, 4, 5].

?- rdup([3,3,4],[1,4]).
false.

?- rdup([],M).
M = [].

?- rdup([1,1,2,2,3,4,7,7],M).
M = [1, 2, 3, 4, 7].

?- rdup([1,1,2,2],[1,2]).
true.
```

b) Define a Prolog predicate flat(L,F) that binds to F the flat list of all elemets in L (where L can be a possibly nexted list). For example flat([a,b, [c,d],[],[[e]],f], L) yields L = [a,b,c,d,e,f]. For example:

```
?- flat([[],[]],L).
L = [].

?- flat([[a,b,[c,d]],f],L).
L = [a, b, c, d, f].

?- flat([[1,2,3],1,2,[[1,2],3]],L).
L = [1, 2, 3, 1, 2, 1, 2, 3].

?- flat([[1,2],3],[1,2,3]).
true.
```

c) Define a Prolog predicate project/3 that selects elements from a list by their position and collects them in a result list. For example, the goal of project([2,4],[a,b,c,d],L) should produce the answer L=[b,d]. You can assume that the numbers in the first list are strictly increasing, that is, your implementation does not have to care about situations like project([1,1,2],..) or project([2,5,3],..). For example:

```
?- project([3,4,7],[1,2,3,4,5,6,7,8],[3,5]).
false.

?- project([3,4,7],[1,2,3,4,5,6,7,8],L).
L = [3, 4, 7].

?- project([1,3],[a,b,c,d,e],L).
L = [a, c].

?- project([1,3],[a,b,c,d,e],[a,c]).
true.
```