Homework 6 (Prolog)

CS 381, Spring 2019

June 16, 2019

Prepared by

Adam Stewart *(stewaada)*
Anish Asrani *(asrania)*
Lucas Frey *(freyl)*
Mazen Alotaibi *(alotaima)*
Sergei Poliakov *(poliakos)*

CONTENTS

1 Exercise 1. Database Application

1.1 Define a predicate schedule/3 that gives for a student the classrooms and times of his or her taken classes, that is, if you evaluate the goal schedule(mary,P,T), Prolog should give the following result.

```
schedule(A,B,C) :- enroll(A, Z), where(Z, B), when(Z, C).
```

1.2 Define a predicate usage/2 that gives for a classroom all the times it is used. For example, the goal usage(cov216,T) should yield the following result.

```
usage(A,B) :- where(Z, A), when(Z, B).
```

1.3 Define a predicate conflict/2 that can compute conflicts in the assignment of classes to classrooms. A conflict exists if two different classes are assigned to one classroom for the same time. The arguments of the conflict predicate are two class names. You can use the goal conflict(275,X) (or conflict(X,275)) to find out any classes that are in conflict with the class 275.

```
conflict(A,B) :- when(A, Z), when(B,Z), A\==B.
```

1.4 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 one classroom. The last condition means that a student Jim can meet any student who has a class that is in the same classroom and immediately follows Jims class. (Note that your definition of meet doesnt have to be symmetric, that is, if students A and B can meet, then your implementation has to return Yes for meet(A,B) or meet(B,A), but not necessarily for both calls. You can ignore the case when students are enrolled in conflicting classes.)

```
meet(A,B) :- enroll(A,X), enroll(B,X), A\==B.
meet(A,B) :- enroll(A,X), enroll(B,Y), when(X,Z), ZA is Z+1, when(Y, ZA), where(X, W),
    where(Y, W), A\==B.
meet(A,B) :- enroll(A,X), enroll(B,Y), when(Y,Z), ZA is Z+1, when(X, ZA), where(X, W),
    where(Y, W), A\==B.
```

2 Exercise 2. List Predicates and Arithmetic

2.1 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.

```
rdup([X],[M]) :- X = M.
rdup([X,Y|Z],[A|B]) :- X \== Y,  X = A, rdup([Y|Z],B).
rdup([X,Y|Z],M) :- X == Y, rdup([Y|Z],M).
```

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

```
flat([X],[L]) :- not(is_list(X)), L = X.
flat([X],[L]) :- is_list(X), flat(X,[L]).
flat([X|Y],[A|B]) :- not(is_list(X)), A = X, flat(Y,B).
flat([X|Y],L) :- is_list(X), append(Y, X, Z), flat(Z,L).
```

2.3 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 project([2,4,5],[a,b,c,d],L) should produce the answer L=[b,d]. You can assume that the numbers in the first list are strictly inreasing, that is, your implementation does not have to care about situations like project([1,1,2],...) or project([2,4,3],...).

```
project(1,[X|_],L):- L = X.
project(P,[_|Y],L) :- integer(P), P > 1, NEXTP is P-1, Y \= [], project(NEXTP,Y,L).
project([P|_],B, []) :- integer(P), length(B, Q), Q < P.
project([P|[]],B, [L]) :- project(P,B,L).
project([P|[Pn|_]], B, [N]) :- integer(P), integer(Pn), length(B, Q), Q < Pn, project(P,
    B, N).
project([P|[Pn|X]], B, [N|L]) :- integer(P), length(B, Q), Q >= P, integer(Pn),  Q >= Pn
   ,  project(P, B, N), project([Pn|X], B, L).
```