

# Homework 3

**Due:** The deadline (for both the paper and the electronic submission) is Thursday, July 12, 6pm for the day section, and Friday, July 13, 6pm for the evening section, in 324 drop box (SF, 2nd floor, near bridge to LP).

**Lateness reminder:** penalty 30% if two days late; not accepted later.

**Note:** The names of your predicates should be the ones required in this assignment. The name of the submitted file should **h3.pl**. Otherwise you will lose marks. You may not use any of the following: ; (“or”), -> (“if-then”), or any other special operators in Prolog (which subvert the pure logic programming paradigm).

## 1 Question 1

Consider the following predicates:

```
female(mary).
female(beth).
female(anna).
```

```
male(john).
male(jim).
male(ed).
male(al).
```

```
parent(john,beth).
parent(mary,beth).
parent(john,emma).
parent(mary,emma).
parent(beth,ed).
parent(jim,ed).
parent(emma,anna).
parent(anna,al).
```

**Note:** We say that a predicate holds if it returns “yes” or values to instantiate variables. We say that a predicate fails if it returns “no”.

The predicate `parent(X,Y)` holds if X is the parent of Y. This starter code is given to you on the webpage of the course. Using **ONLY** the predicates `parent`, `female`, and `male`, implement the following predicates:

- a. A predicate `grandchild(X,Y)` which holds if X is the grandchild of Y. Examples:

```
?- grandchild(ed,john).
Yes
```

```
?- grandchild(ed,Y).
```

```
Y = john ;
Y = mary ;
No
```

```
?- grandchild(X,john).
X = ed ;
X = anna ;
No
```

```
?- grandchild(X,Y).
X = ed
Y = john ;
```

```
X = ed
Y = mary ;
```

```
X = anna
Y = john ;
```

```
X = anna
Y = mary ;
```

```
X = al
Y = emma ;
```

```
No
```

- b. A predicate `grandson(X,Y)` which holds if X is the grandson of Y.
- c. A predicate `granddaughter(X,Y)` which holds if X is the granddaughter of Y.
- d. A predicate `descendent(X,Y)` which holds if X is the descendent of Y. Descendent means child, grandchild, grand-grandchild, and so on.

**Note:** for b and c you are allowed to use the predicate defined at a.

## 2 Question 2

a. Define a predicate `first_two(L)` which holds if the list L has the first two elements identical. It fails if the list L does not have the first two elements identical, or if it has fewer than two elements. The list L is always instantiated. Examples:

```
?- first_two([a,a,b]).
Yes
?- first_two([a,b,b]).
No
?- first_two([a]).
No
```

b. Define a predicate `nested_member(E,L)` which has two arguments (the second one is a possibly-nested list), and holds if the element `E` is in the list `L`, on any level. Examples:

```
?- nested_member(2, [1,2,b,c]).
Yes
?- nested_member(b, [1,2,[a,[b]],c]).
Yes
?- nested_member(3, [1,2,[a,b]]).
No
```

c. Define a predicate `remove(E,L,R)` which holds if list `R` is the list `L` from which all occurrences of the element `E` were removed. In case `L` is a nested list, the predicate only removes the occurrences on the superficial level. The first two arguments, `E` and `L`, are always instantiated. Examples:

```
?- remove(a, [1, a, b, a, a, 3], R).
R = [1, b, 3]
yes
?- remove(a, [1, a, b, a, a, 3], R).
R = [1, b, 3] ;
No
?- remove(a, [1, b, 3], R).
R = [1, b, 3] ;
No
?- remove(a, [1, b, [a], 3],R).
R = [1, b, [a], 3] ;
No
```

### 3 What to hand in?

Put all definitions of the predicates above, in a file called **h3.pl**. Add comments to explain what each predicate is doing, what are the preconditions and postconditions. The comments should also indicate the question number the predicate is an answer to.

#### 3.1 On paper

The paper submission has to be put in 324 drop box (SF, 2nd floor, near bridge to LP).

Please use an unsealed envelope, having on top the cover page provided at the end of this file. Put inside:

- A printout of your code (the file **h3.pl**).
- A printout of one or more script files containing tests run for each predicate. For question 1, you don't have to test with a different database (set of predicates `parent`, `male`, and `female`), but you should make your predicates general so that they will work with any database of the given form.

The tests for each predicate you implemented should include several relevant runs, including degenerated cases for the input (such as empty list). Also, make sure you test for multiple solutions and for different legal combinations of constants and variables.

Example of generating a script file named file1:

```
> script file1
> pl
?- X is 1.
Yes
?- ^D
^D
```

You can save your test cases in a file, then cut and paste when you use script. You are allowed to try alternative methods Prolog may provide for producing script files.

### 3.2 Electronically

In addition to your paper submission, you must submit your code electronically.

- The file **h3.pl** containing all your code.

Submit the file **h3.pl** using the following command:

```
submit -N homework3 csc324h h3.pl
```

See `man submit` if you need more information. Note that if you already submitted a file and you wish to replace it with a different version, you can submit it again. But you must use the `-f` option to overwrite the old version.

## 4 How the homework will be marked?

It is worth emphasizing that your work will be **marked not only for correctness, but for logic programming style, comments, and your testing strategy as well.**

**Style:** the code should use logic programming style not imperative style. The code should be as efficient and elegant as possible. You should pay attention to Prolog warnings.

**Comments:** should clearly state what are the preconditions of each predicate (expected number of arguments and their type), and what are the postconditions (results and expected behaviour). Additional comments inside the predicates should be used to explain steps that are not obvious. The comments should consist of full English sentences, including punctuation.

**Testing:** all predicates should be tested. The test cases for each predicate should include relevant cases (no less and no more than the necessary cases), including degenerate cases such as empty lists. You should also test for multiple solutions (using `;`) and different legal combinations of variables and constants. Please explain in a few sentences what is your testing strategy for each question (or annotate your test cases).

## Cover Page for Homework 3

Family name: \_\_\_\_\_ Student #: \_\_\_\_\_

First name: \_\_\_\_\_ CDF id: \_\_\_\_\_

Section:        Day section        Evening section

I declare that this assignment is my own work, and it is in accordance with the University of Toronto Code of Behaviour on Academic Matters and the Code of Student Conduct.

I discussed this assignment with the following people:

Name: \_\_\_\_\_ Name: \_\_\_\_\_

Name: \_\_\_\_\_ Name: \_\_\_\_\_

Signature \_\_\_\_\_