

Problem Set 4—Prolog

Due: Monday, 25 November at noon
Weight: 4% of your course grade.

This assignment asks you to write predicates in Prolog. You may use helper relations as needed in defining these predicates.

You may *not* use any of the following:

- ; (“or”)
- -> (“if-then”)
- ! (cut)

or any other special operators in Prolog (which generally subvert the pure logic programming paradigm). If you are in doubt about what’s allowed, ask. But generally, if you haven’t seen a certain notation in class or tutorial, it’s probably not a good idea to use it without checking with me.

In running Prolog to try out these relations and queries, be sure to load `basics.P` (*i.e.*, type `[basics].` when you start up), as that contains some standard predicates like `append`.

For this assignment, **you are *not* allowed to use any built-in relations** other than `append`. And, except as noted in question 1, you cannot use relations we defined in class unless you give their definition as part of your solution.

1. In this problem, you will extend our family predicates. We just can’t keep away from marriage, after all—we will add new facts to the database with the predicate `married`, such as:

```
married(ken, jo).  
married(lucia, tim).
```

That is, `married(X,Y)` means that `X` and `Y` are spouses. Using this new predicate, define the following new relations:

- (a) `mother-in-law(X,Y)` holds iff `X` is the mother in law of `Y`, *i.e.*, `X` is the mother of `Y`’s spouse.
 - (b) `sister-in-law(X,Y)` holds iff `X` is the sister in law of `Y`. Note that `X` can be `Y`’s sister in law in one of the following ways: by being married to `Y`’s sibling, or by being the sister of `Y`’s spouse.
2. Write relations for the following:
 - (a) `first(X,Y)` holds iff `X` is the first element of list `Y`.
 - (b) `rest(X,Y)` holds iff `X` is all but the first element of list `Y`.
 - (c) `third(X,Y)` holds iff `X` is the third element of list `Y`.
 - (d) `secondLast(X,Y)` holds iff `X` is the second-last element of list `Y`.

For this problem, you must also submit a test suite for each relation. Select your test cases carefully to test the interesting cases. You don't need many cases to thoroughly test these relations.

3. Using only the append relation, formulate queries to determine the following:

- (a) X is the last element of the list L .
- (b) The second and second-last elements of list L are the same.
- (c) The list Y is the list X with one element removed.
- (d) The lists X and Y share a common substring of length 3.

Note that you must write **queries** here, and not new relations.

For example, the answer for “ X is the third element of list L ”, which you may not use for question 2 (c), would be:

```
?- append([_,_],[X|_],L).
```

4. Write the following recursive list predicates. You will need to use the $\backslash+$ (not) operator, to ensure that two variables do not have the same value, such as $\backslash+(X=Y)$.

- (a) `delete(X,L1,L2)` holds iff $L2$ is the same as $L1$ with all instances of X removed.
For example, `delete(a,[a,b,a,a,a,c,c],[b,c,c])` holds.
For this question, you may assume that $L1$ is fully instantiated.
- (b) `remDups(L1,L2)` holds iff $L2$ is the same as $L1$ with the second and succeeding immediately adjacent duplicates removed.
For example, `remDups([a,b,a,a,a,c,c],[a,b,a,c])` holds.
For this question, you may assume that $L1$ is fully instantiated.

For this problem, you must also submit a test suite for each relation. Select your test cases carefully to test the interesting cases. You don't need many cases to thoroughly test these relations.

5. Write relations for the following:

- (a) `even(L)` holds iff the list L has an even number of elements.
- (b) `palindrome(L)` holds iff the list L is a palindrome, *i.e.*, the list of elements of L is the same read from left to right or right to left.
- (c) `selectionsort(L1,L2)` holds iff $L1$ is a list of positive integers, and $L2$ contains the same elements as $L1$ but in non-decreasing order. The name of the predicate indicates the general approach you must take in solving this problem.
Selection sort works as follows: find the smallest element of the list, and move it to the front. Then find the smallest element of the rest, and move it to the next position, and so forth until you've sorted the whole list.
For this question, you may assume that $L1$ is a fully instantiated list of numbers.

For this problem, you must also submit a test suite for each relation. Select your test cases carefully to test the interesting cases. You don't need many cases to thoroughly test these relations.

Silent Policy

A silent policy will take effect 24 hours before this assignment is due. This means that no question asked after noon on Sunday, 24 November will be answered, whether it is asked on the newsgroup, by e-mail or in person.

Handing It In

You must submit this entire problem set electronically.

Submit your code in files `ps4q1.P`, `ps4q2.P`, `ps4q3.P`, `ps4q4.P`, and `ps4q5.P`.

Submit your test suites in files `ps4q2.in`, `ps4q4.in` and `ps4q5.in`.

Run each test suite using the command

```
xsb < filename.in > filename.out
```

and submit the resulting `.out` files.

Submit all your files using the following command:

```
submit -c csc324h -a PS4 filename(s)
```