

# CSE 505: Computing with Logic

Fall 2008

Homework 1

Due: Wed., Sep 24

This is a programming assignment. Write Prolog predicates to solve the following problems. You may define auxiliary (helper) predicates as needed to define the predicates required by the problems. Do *not* use library predicates (predicates defined in XSB's library or elsewhere). Place the predicates in a single Prolog program file and submit that file using Blackboard (in "Assignments" section).

1. Write a Prolog predicate `split(L, L1, L2)` to split a list `L` into two lists such that `L1` and `L2` such that `L1` contains all the elements of `L` occurring at *odd* positions in `L` and `L2` contains all the elements of `L` occurring at *even* positions in `L`. Assume that positions in a list are numbered starting with zero.

For instance,

- `split([a,b,c,d], X, Y)` should give `X=[a,c]` and `Y=[b,d]` as the answer.
- `split([a,b,c,d,e], X, Y)` should give `X=[a,c,e]` and `Y=[b,d]` as the answer.

2. Write a Prolog predicate `delete(X, L1, L2)` which nondeterministically deletes element `X` from list `L1` giving list `L2`.

For instance,

- `delete(a, [a,b,c], X)` succeeds with single answer `X=[b,c]`.
- `delete(a, [a,b,a,c], X)` succeeds with two answers:  
`X=[b,a,c]` and `X=[a,b,c]`.
- `delete(a, [b,c,d])` fails.

3. Write a Prolog predicate `sublist(L1, L2)` which determines, for two lists `L1` and `L2`, whether the elements of `L1` occur consecutively, and in the same order, in `L2`.

For instance, the following queries succeed:

- `sublist([a,b], [a,b,c,d])`
- `sublist([b,c], [a,b,c,d])`
- `sublist([], [a,b,c,d])`

and the following queries fail:

- `sublist([b,d], [a,b,c,d])`
- `sublist([a,b,c,d], [a,b])`
- `sublist([e], [a,b,c,d])`

For full credit, `sublist(X,L)` should enumerate (by backtracking) all sublists of a given list `L`. For example, `sublist(X, [a,b,c])` should give the succeed with the following answers:

`X=[]`, `X=[a]`, `X=[b]`, `X=[c]` `X=[a,b]`, `X=[b,c]`, `X=[a,b,c]`

(The same answer may be returned multiple times.)

4. Write a Prolog predicate `delete_sublist(L1,L2,L3)` which, given two lists L1 and L2 non-deterministically deletes L1 from L2 to give L3.

For instance,

- `delete_sublist([b,c],[a,b,c,d],X)` succeeds with a single answer `X=[a,d]`.
- `delete_sublist([b,c],[a,b,c,b,d,c,a,b,c],X)` succeeds with two answers: `X=[a,b,d,c,a,b,c]` and `X=[a,b,c,b,d,c,a]`.
- `delete_sublist([b,c],[a,b,a,c,b,d,c,a,b],X)` fails.

5. **Change Computation:** Consider a country which has coins of various denominations. When giving change totaling a particular value, we need to find out how many coins of each denomination to use. This question asks you to write a Prolog predicate for this problem. For this homework, assume that we have an unlimited supply of coins of each denomination.

Write a predicate `change` that, given the denominations available and the change amount, computes the ways in which we can give change. Let D be a list of distinct integers denoting the denominations of coins available. The predicate `change` is such that, given D and an integer S, `change(D,S,C)` returns a list of integers C that gives the number of coins of each denomination in D that add up to S.

For instance, `change([25,11,3], 26, C)` returns with `C=[0,1,5]`: i.e. *zero* 25-unit coins, *one* 11-unit coin, and *five* 3-unit coins can be used to make up 26 units. There is no answer to `change([25,12,3], 26, C)`: i.e. there is no way to make change for 26 units with the given denominations of coins.

For full credit, `change` must be able to compute *all* ways of making change; for partial credit, it must compute at least one way of making change.