The University of Melbourne
School of Computing and Information Systems

COMP30020
Declarative Programming
Second Semester, 2018

Assignment 2

Due Wednesday, 19 September 2018, 5:00PM

Worth 0% (optional assignment)

The objective of this project is to practice your Prolog programming skills. You will write a few fairly simple Prolog predicates.

Note well: This project is only assigned for students enrolled in COMP90048. Other students are encouraged to do the exercise for practice, and may submit their work if they like, but it is not part of their assessment, so they will not receive credit for it.

The Assignment

You will implement the following Prolog predicates.

1. `replace(E1, L1, E2, L2)`
   this holds when list `L1` is the same as `L2`, except that in one place where `L1` has the value `E1`, `L2` has `E2`. Note that only *one* occurrence of `E1` is replaced. This must work in any mode in which at least one of `L1` or `L2` is a proper list (that is, either `[]` or a list whose tail is a proper list). For example:

   `replace(2,[1,2,3,4],5,X)` should have only the solution `X = [1,5,3,4]`.

   `replace(2,[1,2,3,2,1],5,X)` should backtrack over the solutions `X = [1,5,3,2,1]` and `X = [1,2,3,5,1]`.

   `replace(2,X,5,[1,5,3,5,1])` should backtrack over the solutions `X = [1,2,3,5,1]` and `X = [1,5,3,2,1]`.

   `replace(X,[a,b,c,d],Y,[a,e,c,d])` should have only the solution `X = b, Y = e`.

   `replace(X,[1,2,3,2,1],Y,[1,5,3,5,1])` should have no solutions (it should fail).

2. `zip(As, Bs, ABs)`
   this holds when `As`, `Bs`, and `ABs` are lists of the same length, and each element of `ABs` is a term of the form A`-`B where A is the corresponding element of `As` and B is the corresponding element of `Bs`. This should work whenever at least one of the arguments is a proper list. Note that `-` is an infix operator here, but aside from syntax, `A-B` is an ordinary term whose functor is `-` and whose two arguments are `A` and `B`. For example:

   `zip([1,2,3,4],[a,b,c,d],L)` should have only the solution `L=[1-a,2-b,3-c,4-d]]`.

   `zip(X,Y,[1-a,2-b,3-c,4-d])` should have only the solution `X=[1,2,3,4], Y=[a,b,c,d]`.

   `zip([1,2,3,4],Y,[1-a,2-b,3-c,4-d])` should have only the solution `Y=[a,b,c,d]`.

`zip(X,[a,b,c,d],[1-P,2-Q,3-R,4-S])` should have only the solution `X=[1,2,3,4], P=a, Q=b, R=c, S=d`.

`zip([1,2,3],[a,b,c,d],L)` should fail.

3. `sublist(Xs, Ys)`
   this holds when `Xs` is a list containing some of the elements of `Ys`, in the same order they appear in the list `Ys`. This should work whenever `Ys` is a proper list. For example:

   `sublist([a,c,e],[a,b,c,d,e])` should succeed.

   `sublist([a,e,c],[a,b,c,d,e])` should fail.

   `sublist([a,X,d],[a,b,c,d,e])` should have the two solutions `X=b` and `X=c`.

   `sublist(X,[a,b,c])` should have the eight solutions `X=[]`; `X=[c]`; `X=[b]`; `X=[b,c]`; `X=[a]`; `X=[a,c]`; `X=[a,b]`; and `X=[a,b,c]`.

Note that for all of these predicates, the order in which solutions are found does not matter, but all listed answers must be found, and no extra solutions are permitted.

You must call your source file `assignment2.pl`.

Assessment

This assignment will not be assessed for credit; that is, it is worth 0% of your final mark. To make it useful as practice for later, assessed work, your code will be tested for correctness.

Your submission will only be checked for correctness. For this assignment, code quality will not be considered. However, for your own sanity, I do recommend commenting your code and programming it carefully, paying due attention to programming technique.

Timeouts will be imposed on all tests. Test cases will be rather small, so the timeouts should only affect you if you create an infinite recursion (infinite loop) or infinite backtracking loop.

Submission

You **must** submit your project from the unix server `nutmeg2.eng.unimelb.edu.au`. Make sure the version of your program source file you wish to submit is on this host, then `cd` to the directory holding your source code and issue the command:

```
submit COMP30020 assignment2 assignment2.pl
```

Important: you must wait a minute or two (or more if the servers are busy) after submitting, and then issue the command

```
verify COMP30020 assignment2 | less
```

This will show you the test results from your submission, as well as the file(s) you submitted. If the test results show any problems, correct them and submit again. You may submit as often as you like; only your final submission will be assessed.

If you wish to (re-)submit after the project deadline, you may do so by adding "`.late`" to the end of the project name (*i.e.,* `assignment2.late`) in the `submit` and `verify` commands.

But note that a penalty, described below, will apply to late submissions, so you should weigh the points you will lose for a late submission against the points you expect to gain by revising your program and submitting again.

It is your responsibility to verify your submission.

Your submission will be tested on one of the servers you are required to submit from. These servers run SWI Prolog version 7.6, which is probably older than the version you will develop on. You are advised to test your program on one of these servers before submitting; in the unlikely case that your program uses some Prolog features or libraries not supported by SWI 7.6, it will be much easier to discover this.

Note that these hosts are only available through the university's network. If you wish to use these machines from off campus, you will need to use the university's Virtual Private Network. The LMS Resources list gives instructions.

Windows users should see the LMS Resources list for instructions for downloading the (free) MobaXterm or Putty and Winscp programs to allow you to use and copy files to the department servers from windows computers. Mac OS X and Linux users can use the `ssh`, `scp`, and `sftp` programs that come with your operating system.