Discussion Problems: "Prolog Continued"
Declarative Methods (JHU 600.432/632)
Prof. Jason Eisner

1. Define a pure Prolog predicate `geq(A,B)` that is true if `A` and `B` are Peano integers with `A` $\geq$ `B`, and false otherwise.

   (Note that for full credit, `geq(three,two)` must be false, since the arguments aren't both Peano integers.)

2. (a) [8 points] Define a Prolog predicate `nth`/3 that finds the n^{th} element of a list (counting from zero). For example, the zeroth, first, and second elements of `[a,b,c,d,e,f]` are `a`, `b`, and `c`, so the query `nth(s(s(z)),[a,b,c,d,e,f],X)` should return `X=c`. Notice that the first argument to `nth`/3 is a Peano integer.

   (b) [2 points] In principle, what "should" be the behavior of the query `nth(N,[a,b,x,d,x,f],x)`?

   (c) [2 points] Given your definition of `nth`, does the previous query in fact behave as it "should"?

   (d) [3 points]What is the behavior of the query `nth(s(z),List,b)`?

3. You are using constraint logic programming (rather than a cumbersome SAT solver) to construct a crossword puzzle.

   For example, variables `X1` through `X5` have domain `a..z` (or `1..26`) and represent the letters of a particular unknown 5-letter word. You therefore write down a constraint `isword(X1,X2,X3,X4,X5)`.

   But this particular crossword puzzle is going to have a theme: you will only allow words like `radar` and `abode` in which consonants and vowels strictly alternate. So you also write down the constraints

```
vowel(X1) #\= vowel(X2), vowel(X2) #\= vowel(X3),
vowel(X3) #\= vowel(X4), vowel(X4) #\= vowel(X5).
```

   (You have previously defined an ECL^{i}PS^e constraint `vowel`/1 as follows:

```
vowel(A) :- A::[a,e,i,o,u].
```

   Then the constraint `vowel(B) #= vowel(C)` will be true iff `B` and `C` have the same "vowel status" (either both are vowels, or else neither are).)

   (a) [8 points] The four new constraints on `X1` through `X5` obviously imply that if `X5` is a vowel, so is `X1`. So if `X5` gets assigned to a vowel during backtracking search (for example, `a`), we could (in principle) immediately remove all consonants from the domain of `X1`. This would make progress and maybe even forces a contradiction quickly.

   What is the constraint solving technique that would allow us to do this in practice? That is, it would let us modify `X1`'s domain as soon as `X5` was assigned (without having to try any possibilities for `X2`, `X3`, and `X4`). Name the technique and explain briefly how it applies here.

   (b) [12 points] Using the recursive features of Prolog, define an ECL^{i}PS^e constraint `alternating`/1 that takes a list of 0 or more letter variables, such that you can just write the constraint `alternating([X1,X2,X3,X4,X5])` and it will be equivalent to the set of 4 simpler constraints that we used above.

   Your definition will look like a Prolog program, but it will have some ECL^{i}PS^e inequality constraints on the right-hand side. You can still use the `vowel` constraints there, as before. The point of this question is that `alternating` will make it easy to automatically define the appropriate set of constraints over any word of any length.

4. [4 points] Consider this program that uses a cut:

```
foo(List,A) :- member(A,List), !.
bar(X) :- member(List,[[a,b,c],[d,e,f],[g,h,i]]),
              foo(List,X), member(X,[a,a,b,b,c,c,d,d,e,e,f,f]).
```

What are all the bindings to `X` (if any) returned by the query `bar(X)`?

5. (a) [4 points] You load a strange Prolog program consisting of this single fact:

`mystery(f(As,B),B,As).`

Does the following query succeed, or return `No`? If it succeeds, what terms are `B` and `Answer` bound to?

`mystery(f([1,5,B,3],B),8,Answer)`

(b) [6 points] Now here's a slightly harder but more interesting version. You load a strange Prolog program consisting of this single fact:

`enigma(g(As,Bs), g(Bs,Cs), g(As,Cs)).`

Does the following query succeed, or return `No`? If it succeeds, what terms are `Bs`, `Cs`, and `Answer` bound to?

`enigma(g([9,2,4|Bs],Bs), g([1,8,0,9|Cs],Cs), Answer)`

(c) [4 **extra credit** points] What common programming problem is solved by calling `enigma/3` with arguments like those in the query above?