Exercises for "Prolog for beginners"
set by Edward Tsang, University of Essex, 2005

Exercise 1. (10%)
(an easy start)

Assume that facts are stored in the database in the form of:

```
parent( P, C )    /* meaning: P is the parent of C */
female( X )       /* meaning: S is female */
```

Given any atom Y, if female(Y) fails, then Y is assumed to be male. (note that this is different from the stipulation in (Bratko 1990)).
Define:

```
uncle( X, Y )
```

which succeeds when X is the uncle of Y, and fails otherwise.

For example, given the database:

```
parent( tom, bob ).
parent( tom, jim ).
parent( bob, ann ).
parent( bob, pat ).
female( pat ).
female( ann ).
```

When called by:

```
?-  uncle(jim, Y).
```

your program should instantiate Y to ann and pat. Test the robustness of your program carefully. For example, does it work with both X and Y instantiated or uninstantiated?

Exercise 2. (20%)
(Exercise on recursion)

Assume that the database contains facts of the form:

```
parent(Parent, Child)
```

Define common_ancestor/3 such that when called by:

```
?- common_ancestor(X, Y, CA)
```

with X and Y instantiated, instantiate CA to the common ancestor of X and Y. If X is the ancestor of Y or vice versa, then CA should be instantiated to the ancestor of the two. common_ancestor(X, Y, CA) should fail if X and Y have no common ancestors.

For example, given:

```
parent( pam, bob ).
parent( tom, bob ).
parent( tom, liz ).
parent( bob, ann ).
parent( bob, pat ).
parent( pat, jim ).
```

in the database, if the following call is made:

```
?- common_ancestor( ann, jim, CA ).
```

your program should instantiate CA to bob.
If called by:

```
?- common_ancestor( liz, tom, X ).
```

your program should instantiate X to tom.

Exercise 3. (30%)
(Exercise on list manipulation)

Assume that the database contains facts of the form:

parent(Parent, Child)

Define ancestor_link/3 such that when called by:

?- ancestor_link(X, Y, Link)

with X and Y instantiated, instantiate Link to a list which contains the ancestors of both X and Y which forms the link between X and Y. Link should start with X, followed by the ancestors of X up to the common ancestor of X and Y, followed by the children of this common ancestor down to and including Y.

For example, given:

```
parent( pam, bob ).
parent( tom, bob ).
parent( tom, liz ).
parent( bob, ann ).
parent( bob, pat ).
parent( pat, jim ).
```

in the database, if the following call is made:

?- ancestor_link(ann, jim, Link).

your program should instantiate Link to [ann, bob, pat, jim].
Note that your program needs not return the shortest possible link.

Exercise 4. (40%)
(The Dutch National Flag Problem)

Define the predicate:

dutch_national_flag(List, SortedList)

such that given a list of unspecified number and unspecified order of atoms 'w', 'r' and 'b', return a list with all the 'r' placed before 'w', which are all before 'b'.
For example, when called by:

?- dutch_national_flag([w, b, b, b, r, w], SortedList)

your program should instantiate SortedList to: [r, w, w, b, b, b].
Note that not necessarily all r, w and b are present in the input list. For example, when called by:

?- dutch_national_flag([w, b, w], SortedList)

your program should instantiate SortedList to [w, w, b].

Challenge:
Challenge yourself by implementating dutch_national_flag/2 using 'difference lists' (Refer to (Bratko 1990) for the use of difference lists.). If you can do that, you may consider yourself "quite good in Prolog".