Major Assignment 4

Date of Submission - **13th April, Thursday**

1. Ashwin, Ankush, Umang, Sanket, and Krishna live in a five-storey hostel. Ashwin doesn't live on the fifth floor and Ankush doesn't live on the first. Umang doesn't live on the top or the bottom floor, and he is not on a floor adjacent to Krishna or Ankush. Sanket lives on some floor above Ankush. Finally, they live on different floors. Write a predicate in Prolog called occupancy which will show which person lives on which floor.

   Example:

   ```
   ?- occupancy(B).
   B = [[sanket, 5], [ankush, 2], [ashwin, 3], [umang, 4], [krishna, 1]]
   ```

2. A bag has 3 green caps, 2 yellow caps and 2 red caps. Three persons A, B and C are called, blindfolded, and a cap is put on each persons head from the bag. When the blindfolds are opened, everyone can see the caps on the other's heads but not the one on his own. In addition everyone knows the contents of the bag. Each person is asked to make a statement of the following form:

   (a) I know that the color of my cap is X.

   (b) I know that the color of my cap is not X.

   (c) I can't say anything definite.

   A is asked first. He says that he cannot say anything definite. B is asked next and he says the same thing. Finally when C is asked, he says that he knows that the color of his cap is X. Write a prolog program that encodes exactly the information contained in A's and B's answers, and determine the color of C's cap. In particular, you have to write a predicate called goal(Color, Combination) which will return the unique color of C's cap and also the possible combination of caps that A, B, and C could be wearing i.e. the combinations that are not eliminated because of A's and B,s statements.

   ```
   ?- goal(Color, Combination).
   Color = g,
   Combination = [r, r, g] ;
   Color = g,
   Combination = [r, y, g] ;
   Color = g,
   Combination = [r, g, g] ;
   Color = g,
   Combination = [y, r, g] ;
   ```

```
Color = g,
Combination = [y, y, g] ;
Color = g,
Combination = [y, g, g] ;
Color = g,
Combination = [g, r, g] ;
Color = g,
Combination = [g, y, g] ;
Color = g,
Combination = [g, g, g].
```

3. Given a $n \times n$ chess-board, a knights tour is a sequence of moves such that starting from the initial position `pair(1,1)`, every position in the chessboard is visited exacly once. Write a program such that given the query

   `tour(Res).`

   `Res` is bound to the sequence of moves constituting the knight's tour. Fix n to be 7; anything higher takes a very long time.

   ```
   ?- tour(Res).
   [pair(1,1),pair(3,2),pair(5,3),pair(7,4),pair(5,5),pair(7,6),pair(5,7),
   pair(6,5),pair(7,7),pair(5,6),pair(3,7),pair(4,5),pair(6,6),pair(4,7),
   pair(2,6),pair(3,4),pair(4,6),pair(6,7),pair(7,5),pair(5,4),pair(7,3),
   pair(6,1),pair(4,2),pair(6,3),pair(4,4),pair(2,3),pair(3,1),pair(1,2),
   pair(2,4),pair(1,6),pair(3,5),pair(2,7),pair(1,5),pair(3,6),pair(1,7),
   pair(2,5),pair(1,3),pair(2,1),pair(3,3),pair(1,4),pair(2,2),pair(4,1),
   pair(6,2),pair(4,3),pair(5,1),pair(7,2),pair(6,4),pair(5,2),pair(7,1)]
   ```

4. On the left bank of a river, there are 3 cannibals and 3 missionaries. There is also a boat on the left bank which can carry at most 2 persons across the river. If any bank has more cannibals than missionaries, then it is a dangerous situation, since the cannibals may eat the missionaries. Needless to say, missionaries do not eat cannibals; they prefer biryani. The problem is to determine how they can all cross the river without anyone getting eaten up. You must have a predicate `safe(Ans)` such that at the end of the execution, `Ans` must be bound to a list which looks like `[[3, 3, left], [2, 2, right], [3, 2, left], [3, 0, right], ...]`. Here an element `[M, C, B]` of the list represents the number of missionaries `(M)` and cannibals `(C)` on the left bank and the boat position `(B)`. As you can see, in the first crossing, a missionary and a cannibal crossed over from the left bank to the right bank. You can use a built-in predicate called `reverse(X,Y)` which is true if `Y` is the reverse of `X`. Note that prolog sometime does not print a list completely but indicates the elements towards the end by dots `(...)`. To force full printing of `L`, use the predicate `write(L)`.

Get more Prolog problems to solve from

`https://sites.google.com/site/prologsite/prolog-problems`