Computational Logic: CS 6374
CS 6374: Homework

All exercise numbers refer to the Sterling and Shapiro text book.

1. Exercises 8.3.1 (i), (iii), (vi), (vii)

2. Exercises 11.3 (i) and (ii)

3. Solve the Belgian Snake Problem (described below)

4. Program the N-Queen problem from the book, as discussed in class

5. Write a Prolog program to solve cryptarithmetic addition problems such as

```
   S E N D
+  M O R E
-----------
M O N E Y
```

The solution is D = 7, E = 5, M = 1, N = 6, 0 = 0, R = 8, S = 9, Y = 2. Each letter should stand for a unique digit. If there is a solution, Prolog should return the list of letters and corresponding digits. If there is no solution, Prolog should report 'no'.

6. Solve the stable marriage problem in Exercise 14.1 (ii) pg. 261

7. Program the block worlds problem described in the book (program the intelligent behavior using `choose_action` on page 269). Assume that there are 3 locations `p, q` and `r`, and five blocks `a, b, c, d, e`. Generate a plan to go from initial configuration shown below to final configuration shown below.

Initial configuration:

```
b          e
a    c     d
-------------
p    q     r
```

Final configuration:

```
e          a
d    b     c
-------------
p    q     r
```

8. Consider the missionary-cannibal problem. Three missionaries and three cannibals come to a river that they want to cross and find a boat that holds two. If the cannibals ever outnumber the missionaries on either bank, the missionaries will be eaten. Think of this problem as a planning problem and program it in Prolog. You should print a sequence of moves (one per line) that lists the people crossing the river at each step. You can

represent the 3 cannibals as c1, c2 and c3, and the three missionaries as m1, m2 and m3. For example, the move:

c1 m1

means that cannibal c1 and missionary m1 went from the bank where the boat is to the other bank. Keep in mind that the boat is always needed to go from one side to another.

1. Belgian Snake Problem

The body of the Belgian snake shows a repeating pattern. However, the pattern is not necessarily repeated an integral number of times. A pattern consists of a sequence of rings, and a ring has an identifier which is an atom of length 1. When the Belgian snake takes a nap, it likes to lie coiled up in a particular way: it always lies in a rectangle, its head in the upper left corner and filling the rectangle row by row (see the query below). Write a predicate `snake/3`, which displays such a coiled up Belgian snake. This predicate will be called with the following three arguments: a list of atoms representing a pattern, a list whose length is equal to the number of rings in one column and a list whose length is equal to the number of rings in one row. Your `snake/3` should draw the coiled up snake as output on the screen. For example:

```
?- snake([a,b,c,d],[_,_,_,_,_],[_,_,_]).

abcda
badcb
cdabc
```

This snake would look like `abcdabcdabcdabc` when stretched out.

There is a catch: Belgian snakes dislike arithmetic computations very much. Therefore, we urge you to avoid any arithmetic.

Hints *The snake consists of an ever repeating pattern, and one way of representing this is by a cyclic list: it contain the pattern and bites its tail. This cyclic list is used as a infinite supply of the pattern, from which we need to take pieces with the same length as the list in the second argument. This piece must be reversed for even rows.*
