Cairo University Faculty of Engineering Computer Engineering Department	CMP614 Theory of Computation and Programming Languages
	Logic Programming Sheet
	Due on 1/20/2012

1 Introduction

This homework has two objectives. First, you will exercise some basic logic programming skills. Second, you will be introduced to the joy of backtracking search.

To use the prolog system, go and download GNU prolog http://www.gprolog.org. You can find versions for Linux, Mac OSX and Windows. To run GNU prolog, simply type your prolog program in a text file (e.g., `mycode.pl` and load the prolog system (from a terminal)

```
gprolog
```

and type the command

```
consult('mycode.pl').
```

You can type any query at the prompt and you can reload the same program multiple times. I strongly encourage all of you to browse the GNU prolog documentation http://www.gprolog.org/manual/gprolog.html whenever you wish to find how to do a specific operation. For instance, you may check how to use `bagof/3`. All the standard prolog construct exist and are documented except the `not` predicate which you can easily implement from the notes in case you need it. You are free to use any standard Prolog feature. If you have doubts about what you can or cannot use for the assignment, please ask for clarifications.

2 Question 1

Write the following predicates

```
prefix1(X,Y):-
```

This predicate states that X is a prefix of Y where X and Y are lists. Make sure that the program

```
top :- prefix(X,[1,2,3,4,5]),print(X),print('\n'),fail.
```

finds all the solutions.

3 Question 2

Consider the following four prolog programs that represent the min relationship, i.e., `min(X,Y,Z)` holds if Z is the minimum of X and Y.

```
    min(X,Y,X):-X<Y.    min(X,Y,X):-X<Y,!.     min(X,Y,X):-X<Y,!.     min(X,Y,Z):-X<Y,!,X=Z.
    min(X,Y,Y):-X>=Y.   min(X,Y,Y):-X>=Y,!.    min(X,Y,Y).            min(X,Y,Y).
```

Do they have identical semantics ? Discuss their relative merits.

4 Question 3

As an introduction to the topic of Constraint Programming, here is a proto-typical problem requiring backtracking search. Consider the following puzzle:

Consider a 2D matrix of n by n tiles. Place all the numbers between 1 and n^2 into the tiles so that the following constraints are satisfied:

- Each tile contains exactly one number

- All the numbers are used
- Each number is used at most once
- The sum of all the tiles in each row is equal to the sum of all the tiles in each column and is also equal to the sum of all the tiles in the two diagonals. As a bonus, consider that this sum (denoted X) is always equal to $n(n^2+1)$. Clearly, if the tile matrix is denoted by T, then we have

$$\begin{aligned}
\sum_{j=1}^{n} T_{i,j} &= X \quad \forall i \in \{1,\ldots,n\} \\
\sum_{i=1}^{n} T_{i,j} &= X \quad \forall j \in \{1,\ldots,n\} \\
\sum_{i=1}^{n} T_{i,i} &= X \\
\sum_{i=1}^{n} T_{i,n-i+1} &= X
\end{aligned}$$

4.1 Prolog

First, you should produce a prolog implementation that contains two predicates

```
top(N,S) :- ....

topAll(N,C) :- ...
```

such that `top` produces and returns one solution for a magic square whose side is N whereas `topAll` searches for all the solutions for a magic square whose side is N and produces as a result the number of solutions C.

4.2 Other

You can use any programming language and any paradigm we have covered so far (procedural, functional or logical) to write a program that solves the same two problems (find one and find all solutions). You should attempt to get the fastest possible code for the largest possible value of N. You should document your efforts with the following:

- The working program
- The time it took you to write the program
- The performance of the program on the two types of sub-problems.
- How hard it was to produce a working program.
- If you did not succeed in writing a program that solve the problem, you should report your failed attempts. It is very important that you describe as precisely as possible what you tried, why it did not work and what caused these failures (algorithmic issue, implementation design, poor performance,....). Keep in mind that a *substantial* part of your grade will be function of how well you report your attempts. You should give yourself a reasonable time limit on this homework. If, after some numbers of hours, you did not produce a working implementation, you should wrap up and hand in a document describing everything you tried and why it failed. Simply reporting that nothing worked without justification will not earn you any grade. The deadline for this homework is strict. I will not accept delivery after the due date.

5 Acknowlegement

This sheet is taken from a class that was taught by Prof. Laurent Michel at the University of Connecticut in 2005.