Homework: Logic Programming

Learning Objectives:

1. Problem solving using logic programming paradigm
2. Prolog programming

Instructions:

- Total points 36 pt
- Early deadline: Apr 21 (Wed) at 11:59 PM; Regular deadline: Apr 23 (Fri) at 11:59 PM (or till TAs start grading the homework)
- Download and install Swi-prolog `http://www.swi-prolog.org/`
- Please zip **.pl files and output files (e.g., screenshot) for all the solutions** and submit it to Canvas.

Questions:

1. (3 pt) Understand the following Prolog program:

   Given:
   $mystery([\,],[\,]).$
   $mystery([H|Tail],[H,H|DTail]) :-$
   $mystery(Tail,Dtail).$

   What would Z be in $mystery([1,4,6],Z)$.

2. (10 pt) Prolog programming:

   - (4 pt) [Prolog for numbers] Write a Prolog program to compute GCD of two numbers.
     Example:

```
?- gcd(10,6,X).
2.
?- gcd(10,5,X).
5.
```

   - (6 pt) [Prolog for list] Write a Prolog program to duplicate the elements of a list a given number of times.
     Example:

```
?- duplicate([a,b,c],2,X).
X = [a,a,b,b,c,c].
?- duplicate([a,b,c],3,X).
X = [a,a,a,b,b,b,c,c,c].
```

3. (5 pt) [Prolog for integer constraints] Write a Prolog program to generate the integer values of x, y and z that can satisfy the constraints in the following C program. If no such values can be found, return `false`.

```
 if (2x == y) {
        if (y == x+ 10)
                z = x+y;
        else
                z = x-y;
}
```

4. (6 pt) [Prolog for logic puzzle] Write a Prolog program (including the query) to solve the following logic puzzle:

   Five people were eating apples, A finished before B, but behind C. D finished before E, but behind B. What was the finishing order?

5. (12 pt) [Prolog for parsing] Write a Prolog program for parsing:

   (a) (7 pt) Consider the grammar we worked in HW1 below. Write a Prolog program that parses strings using this grammar. Your program can be used to check if a given sentence can be generated by the grammar. An example interpreter session is provided below.

   Grammar:

   - terminals: $x, y, z, >, <, 0, 1, +, -, =$, if, then, else
   - non-terminals: S, F, B, T, E, N
   - start symbol: S
   - production rules:
     $S \rightarrow F|T\ N\ T$
     $F \rightarrow$if B then begin S end |if B then begin S else S end
     $B \rightarrow (T\ E\ T)$
     $T \rightarrow x|y|z|1|0$
     $E \rightarrow > \mid <$
     $N \rightarrow + \mid - \mid =$

   Example:

```

| ?- sentence([if, x, > , 0, then, begin,'(', [x, =, 1],')', end]).
| true.
| ?-sentence([if, x, > , 0, then, begin, '(',[x, =, 1] ,')', end, else, begin, '(
    ',[x, =, 0] ,')', end]).
| true.
| ?sentence([if, x, > , 0, then, begin, [x, =, 1], end]).
| false.
| ?- sentence([if, x, > , 0, then, begin, '(',[x, =, 1] ,')', end, else, (x, =,
   0)]).
| false.
```

(b) (3 pt) Write the query to generate all possible sentences that can be derived from the grammar. Show the screenshot of 3 sentences.

(c) (2 pt) Does the order of the sub-goals in your rules make a difference?