Assignment Set-II

Logic Programming using Prolog

- **Assignment will be evaluated by the TAs.**
- **You should submit complete source codes and executable files, and a README file containing the logic and predicates you have used in your program.**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F for the whole course, irrespective of your performance in the other parts of the lab.**
- **Submission Deadline: October 23, 2016**
- **Total weight = 25%**
- **Marks distribution: 40, 30, 30**

Assignment-1: **[40 Marks]**

Suppose you are using the unary number system. For example 5 will be written as '11111'. Your task will be to implement some basic arithmetical operations for working with unary numbers. We will represent unary numbers as lists of x's of the appropriate length. Thus, five would be [x, x, x, x, x], twelve would be [x, x, x, x, x, x, x, x, x, x, x, x], and zero would be []. In the sequel, all numbers are understood to be such non-negative integers given in unary notation. You don't need to use any "normal" numbers in your program and you should not use any arithmetic operations provided by Prolog.

[a] The successor of a number is the number we obtain if we add one to it. Thus, for example, the successor of four is five. Write a predicate called ***successor*** that will return, in the second argument position, the successor of the number provided in the first argument position.

Examples:

?- successor([x, x, x], result).
result = [x, x, x, x]
Yes

?- successor([], result).
result = [x]
Yes

[b] Implement a predicate ***plus*** to compute the sum of two given numbers.

Example:

?- plus([x, x], [x, x, x, x], result).
result = [x, x, x, x, x, x]
Yes

[c] Implement a predicate ***minus*** to compute the subtraction of second argument from the first argument and the result will be stored in the third argument.

Examples:

?- minus([x, x, x, x, x, x, x], [x, x, x, x], result).
result = [x, x, x]
Yes

?- minus([x, x, x], [x, x, x, x], result).
result = []
Yes

[**Note**: As we are assuming that all the numbers are non-negative so the concept of negative result won't arise, so it will return zero.]

[d] Implement a predicate ***multiply*** to multiply two given numbers.

Example:

?- multiply([x, x], [x, x, x, x], result).
result = [x, x, x, x, x, x, x, x]
Yes

[e] Implement a predicate ***divide*** to divide two given numbers and the result will be either yes or no depending on, whether the second argument is divisible by the first argument or not. Assume that the first argument is always greater than the second argument.

Examples:

?- divide([x, x, x, x], [x, x],).
Yes

?- divide([x, x, x,], [x, x],).
No

Assignment - 2: **[30 Marks]**

Suppose you are designing a robot movement system, based on the following commands: turn ***right*** (which rotate the robot by 90 degree towards right), turn ***left*** (which rotate the robot by 90 degree towards left) and ***move*** (which makes the robot to move forward by 1 metre). Initially your robot is at a grid position (0,0), facing east. Your task is to write a Prolog predicate ***status*** that will return the robot's position and orientation after having executed a given list of commands. For example, if your robot first moves forward twice, then turns left, and then moves three more times, then it will be at position (2,3), facing north.

Examples:

?- status([move, move, left, move, move, move], Position, Orientation).

Position = (2,3)
Orientation = north
Yes

?- status([], Position, Orientation).

Position = (0,0)
Orientation = east
Yes

?-status([left, left, move], Position, Orientation).

Position = (-1,0)
Orientation = west
Yes

Now you are required to design a strategy for finding the route from the current position to a final position. For the above purpose you have to write a predicate ***movement*** which will take the current position, the current orientation and final position, and returns the movement strategy by which the robot can move from current position to the final position. Note that positions are pairs of the form (X,Y), with X and Y representing integers, while the orientation has to be one of the four atoms north, south, west, east.

Examples:

?- movement((0,0), east , (2,3)).

[move, move, left, move, move, move]
Yes

?- movement((0,1), north , (2,-2)).

[right, move, right, move, move]
Yes

Assignment - 3: **[30 Marks]**

In Techniche-2016 there are food stalls along a line where a great variety of delicious food is sold. Due to the heavy rain, the Pizza-Hut chalkboard was wiped out where lists of various types of pizza with their ingredients and prices were written. Now your job is to rewrite the board with the previous (wiped out) information using the following information: There are five types of pizza with completely different toppings and prices. Each type of pizza has two toppings: one from the list of chicken, mutton, prawn, salami and tuna; the other from the list of onion, corn, olive, tomato and pineapple. Five prices are Rs 55, Rs 65, Rs 70, Rs 85 and Rs 100. Hawaiian pizza has mutton and cost over Rs 65. Marco-Polo had tomato, but not chicken, which is on the pizza that costs Rs 85. Pepperini costs Rs 70 and Super-Supreme had no pineapple. The price of the pizza has tuna and corn is not Rs 65 and the pizza that cost Rs 55 has olive but not salami. The price of the pizza that has pineapple is not Rs 100. The last type of pizza is called Ninja-pizza. Use the Prolog logic programming to determine which pizzas have what toppings and prices.