## **CSE 240 Homework 13, Fall 2015 (50 points)**

Due Saturday, December 5, 2015 at 11:59PM,

No grace period, and no late submission period.

The submission link will expire at 11:59pm on December 5, 2015

### Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures. By the end of the assignment, you should have exercised

* Prolog list operations and manipulations
* Prolog recursive rules with multiple nested calls

**Reading**: Chapter 5 and lecture slides.

You are expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their office hours.

You are encouraged to ask and answer questions on the course discussion board. (However, **do not share your answers** in the course discussion board.)

### Practice Exercises (no submission required)

1. Answer the multiple choice questions in text section 5.9 that have been covered in the lectures.

2 Complete the questions 2 through 5 in text section 5.9.

10 Define recursive rules to compute mathematical functions Y = XN.

11 Define a rule to return the common members of two lists. Trace the execution manually and by the trace routine. Compare the two traces. The format of the rule is

common(X, List1, List2)

12 Using BNF notation (see text chapter 1), a Prolog list can be recursively defined as follows:

<list> ::= [], where [] is an empty list

<list> ::= [<X> | <Y>], where X is a variable or value, and Y is a list.

In the definition, [<X> | <Y>] can be considered to be a pair defined in text chapter four.

12.2 Describe the steps and the simplification rules that you use in each step to simplify the list   
[1 | [2 | [3 | []]]] into [1 2 3].

12.3 Is this structure [[1 | 2] | [3 | [4 | 5]]] a valid Prolog list? Explain your answer.

7 Compare different programming paradigms.

7.1 What are the differences between the variables in imperative, functional and logic programming languages?

7.2 What are the differences between the parameters (arguments) in imperative, functional and logic programming languages?

7.3 There are several data passing mechanisms between the calling function (caller) and the called function (callee): call-by-values, call-by-reference, return value, and global variable (global name). What data passing mechanisms are supported by imperative C, functional Scheme, and logic Prolog? Draw a table to summarize the supported mechanisms by these three languages.

8 Execution model of Prolog

8.1 How is a Prolog program executed? What is the definition of a match?

8.2 What is a backtracking point? When will a backtracking point recorded by the Prolog runtime?

8.3 What is a cut (!)? Why do we need a cut? When should we use a cut?

8.4 When do we need to use the repeat? How do we exit a repeat?

8.5 Does the order of the clauses in a rule matter? Does the order of the facts and rules in a fact/rule base matter?

### Programming Assignment (50 points)

**1** Follow the Fantastic Four-Step Abstract approach to define **recursive rules** to compute mathematical functions. **You must indicate the steps in the comments for all parts**. Note, a Prolog rule does not return a value. You need to use a parameter to hold the return value. **You may NOT use the exponential operator \*\* to compute the expressions.**  [30 points]

1.1 The Ackermann function is defined recursively for two positive integers s and t as follows:

Ack(S, T) =

1.1 Follow the fantastic-four design steps to write the recursive rule ackermann(S,T,A) where A is the parameter to hold the return value (see test cases below). [10 points]

Test cases:

ackermann(0,0,A) 🡪 1

ackermann(1,1,A) 🡪 3

ackermann(2,3,A) 🡪 9

ackermann(3,4,A) 🡪 125

1.2 Write recursive rules *exp*(*Y, X, N*) to compute mathematical functions *Y* = *XN*, where *Y* is used to hold the result, *X* and *N* are non-negative integers, and *X* and *N* cannot be 0 at the same time, as 00 is undefined. The program must print an error message if *X* = *N* = 0. [10]

1.3 Write recursive rules *factorial(Y, X, N)* to compute Y = (*XN*)! This function can be described as the factorial of *exp*. The rules must use the rule *exp* that you designed. [10]

**2** A toy company produces bags of marbles. Each bag of marbles weighs exactly 30 grams. There are 5 different color marbles, each with a corresponding weight. [20 points]

|  |  |
| --- | --- |
| Color | Weight (grams) |
| Red  Orange  Yellow  Green  Blue | 1  2  3  4  5 |

2.1 Define a rule weight(R, O, Y, G, B) to find out how many marbles of each color can be contained in each bag, where R, O, Y, G, and B are the number of red, orange, yellow, green, blue marbles, respectively, assuming the size of the bag is big enough to hold all the marbles with the total weight of 30 grams. [15]

2.2 Use your weight rule to find all possible outputs to the following question (goal). [3 point]

| ?- weight(5, O, Y, G, 3). Put all answers of the question in comment.

2.3 Use your weight rule to find all possible outputs to the following question (goal). [2 point]

| ?- weight(10, O, 1, G, 2). Put all answers of the question in comment.

### Grading of Programming Assignment

The TA will grade your program following these steps:

(1) The TA will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(2) Compile the code. If it does not compile, 20% of the points given in (1) will be deducted. For example, if you are given 20 points in step (1), your points will become 16 if the program fails to compile.

(3) If the code passes the compilation, the TA will execute and test the code. If, for any reason, the program gives an incorrect output or crashes for any input, 10% of the points given in (1) will be deducted.

### What to Submit?

You are required to submit your solutions in a compressed format (.zip). Make sure your compressed file is label correctly - lastname\_firstname13.zip.

The compressed file MUST contain the following:

recursive\_rules.pl.

weight.pl.

The files must contain the rules and the outputs of the goals. No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded. (Even if the assignment is submitted late.)

### Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

### Late submission deduction policy for the Final Homework

* No grace period;
* No late submission.
* The submission link will expire (disappear) after the official due date.