CSCI3180 – Principles of Programming Languages – Spring 2017

Assignment 4 — Declarative Programming

Deadline: Apr 23, 2017 (Sunday) 23:59

1 Introduction

Declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow. You will gain experience of declarative programming with an emphasis on Prolog and ML in this assignment.

2 Logic Programming

Implement the required predicates or queries of the following two problems in a Prolog program file named "asg4.pl". You should clearly indicate using comments the corresponding question number of each sub-problem. Note that the answers which are queries should be written in comments as well.

1. A *binary tree* is either empty or composed of a root and two children, which are binary trees themselves. A binary tree consists of a set of *nodes* and lines connecting parent with children. The nodes are depicted by circles with labels written inside.

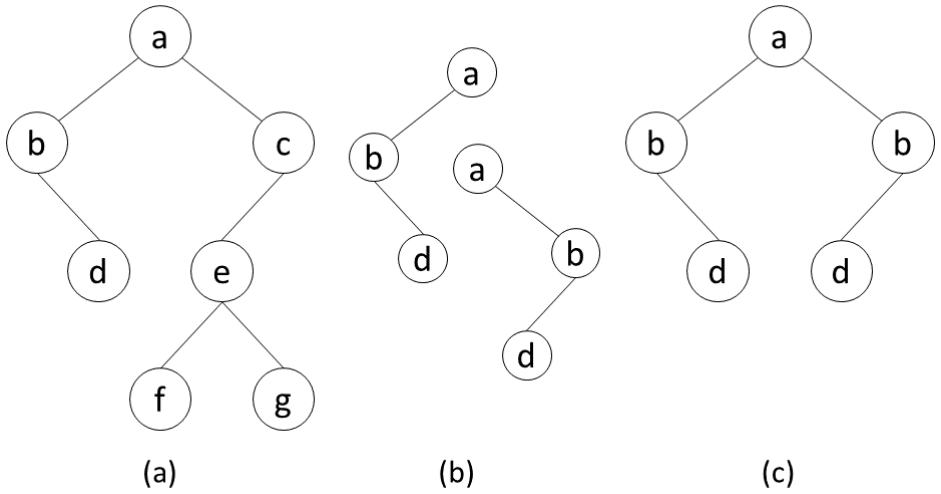

Figure 1: An example of binary tree

In Prolog, we represent an empty tree by the atom "`nil`" and a non-empty tree by the term "`bt(X,L,R)`", where `X` denotes the root node, `L` and `R` denote the left and right subtrees respectively.

(a) Represent the binary tree shown in Figure 1(a) as a Prolog term.

(b) Define the predicate `istree(Term)`, where "`Term`" can be any Prolog term. Predicate `istree(Term)` is true if "`Term`" represents a binary tree.

(c) Define the predicate `mirror(Tree1, Tree2)`, where "`Tree1`" and "`Tree2`" are both Prolog terms representing binary trees. Predicate `mirror(Tree1, Tree2)` is true if "`Tree1`" and "`Tree2`" are mirror images of each other. For example, `nil` is the mirror of `nil`. The two trees in Figure 1(b) are mirror images of each other.

(d) Give a query to generate the mirror image of the tree on the left-hand side in Figure 1(b).

(e) Based on `mirror/2`, define `symmetric(Tree)`, where "`Tree`" is a Prolog term representing a binary tree. Predicate `symmetric(Tree)` is true if "`Tree`" is symmetric. For

example, `nil` and a single-node tree are both symmetric. The tree in Figure 1(c) is a symmetric tree.

2. Recall the successor notation for representing natural numbers, and the `sum(X,Y,Z)` relation defined in the lecture which is true if Z is the sum of X and Y.

   (a) Define `uint_num(X)` which is true if `X` is a natural number.

   (b) Define `gt(X,Y)` which is true if `X` is greater than `Y`.

   (c) Give a query to list all natural numbers less than 3.

   (d) Based on `sum/3`, define `product(X,Y,Z)` which is true if `Z` is the product of `X` and `Y`.

   (e) Give a query to compute the product of 2 and 3.

   (f) Give a query to compute the result of 8 divided by 4.

   (g) Based on `sum/3`, define `intdiv(X,Y,Z)` which is true if `Z` is the integer quotient of `X` divided by `Y`. If `X` $\geq$ `Y`, `X` could always be divisible by `Y` and the quotient is `Z`. If `X` $<$ `Y`, the quotient is 0.

   (h) Give a query to compute the result of 2 divided by 3 using `intdiv`.

   (i) Give a query to compute product of 1 and 2 using `intdiv`.

3 Functional Programming

Implement the required functions of the following two problems in an ML program file named "asg4.ml". You should clearly indicate using comments the corresponding question number of each sub-problem.

1. A *binary tree* is a data structure in which each *node* has a label and at most two children which are referred to as the left child and right child. Recall the type definition of a binary tree:

   ```
   datatype 'a bTree = nil | bt of 'a bTree * 'a * 'a bTree;
   ```

   where "`nil`" stands for the empty tree and "`bt`" is the tag of a non-empty binary tree.

   (a) The *size* of a binary tree is the number of elements in the tree. For example, the size of an empty tree is 0 and the size of a single element tree is 1. The size of the binary tree in Figure 1(a) is 7.

   Implement an ML function `size` which takes a binary tree of any type as input and returns its size. Your function `size` should make use of pattern matching.

   (b) A *leaf node* of a binary tree is a node without any child nodes, which means the leaf node has two `nil` children. The *height* of a binary tree is the maximum number of edges from the root to leaf nodes. For example, the empty tree and a single element tree have a height of 0. The height of the binary tree in Figure 1(a) is 3.

   Implement an ML function `height` which takes a binary tree of any type as input and returns its height. Your function `height` should make use of pattern matching.

   (c) The number of leaves of an empty tree, namely a `nil` tree, is 0. A single element tree has 1 leaf node, which is just the root node. The binary tree in Figure 1(a) has 3 leaf nodes.

   Implement an ML function `nleaves` which takes a binary tree of any type as input and returns the number of leaf nodes. Your function `nleaves` should make use of pattern matching.

2. Recall the built-in function `hd` for obtaining the first element of a list. We consider lists of reals for the following problems. Suppose the input list is always non-empty and the head of a list is the first element.

   (a) Implement an ML function `last` to return the last element of the input list.

(b) Implement an ML function `tln`, which takes a list `L` and an integer n as inputs, to return a list of all but the first n elements of the input list. You can safely assume that the non-negative integer n is always less than or equal to the length of `L`.

(c) Implement an ML function `hdn`, which takes a list `L` and an integer n as inputs, to return a list of the first n element of the input list. You can safely assume that the non-negative integer n is always less than or equal to the length of `L`.

(d) Based on the function `tln`, implement an ML function `lnth`, which takes a list `L` and an integer n as inputs, to return the n-th element of the input list. You can safely assume that the integer n is always greater than 0 and less than or equal to the length of `L`.

(e) Implement an ML function `sum` to return the sum of all elements of the input list.

(f) The *variance* of a list of reals $[x_1, ..., x_n]$ can be the average of the squares minus the square of the average. One formula for computing variance is

$$var(x) = \frac{\sum_{i=1}^{n} x_i^2}{n} - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2 \quad (1)$$

Use function `square`, built-in functions `length` and `map` and function `sum` in (e) to implement an ML function `variance` to return the variance of the input list. Function `square` is defined as

```
fun square(x:real) = x*x;
```
val square = fn : real → real

4 Submission Guidelines

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of submission problem on your side, marks will still be deducted. So please start your work early!

1. In the following, **SUPPOSE**

   your name is *Chan Tai Man*,
   your student ID is *1155234567*,
   your username is *tmchan*, and
   your email address is *tmchan@cse.cuhk.edu.hk*.

2. In your source files, insert the following header. REMEMBER to insert the header according to the comment rule of Prolog and ML.

```
/*
 * CSCI3180 Principles of Programming Languages
 *
 * --- Declaration ---
 *
 * I declare that the assignment here submitted is original except for source
 * material explicitly acknowledged.  I also acknowledge that I am aware of
 * University policy and regulations on honesty in academic work, and of the
 * disciplinary guidelines and procedures applicable to breaches of such policy
 * and regulations, as contained in the website
 * http://www.cuhk.edu.hk/policy/academichonesty/
 *
 * Assignment 4
 * Name : Chan Tai Man
 * Student ID : 1155234567
 * Email Addr : tmchan@cse.cuhk.edu.hk
 */
```

   The sample file header is available at

   `http://course.cse.cuhk.edu.hk/~csci3180/resource/header.txt`

3. Late submission policy: less 20% for 1 day late and less 50% for 2 days late. We shall not accept submissions more than 2 days after the deadline.

4. Your solutions for Section 2 will be graded using SICStus Prolog 3.12.7 in the CSE Unix platform. Solutions for Section 3 will be graded using Standard ML 110.0.7 in the CSE Unix platform.

5. `Tar` your source files to `username.tar` by

   ```
   tar cvf tmchan.tar asg4.pl asg4.ml
   ```

6. `Gzip` the `tarred` file to `username.tar.gz` by

   ```
   gzip tmchan.tar
   ```

7. `Uuencode` the `gzipped` file and send it to the course account with the email title "HW4 *studentID yourName*" by

   ```
   uuencode tmchan.tar.gz tmchan.tar.gz \
   | mailx -s "HW4 1155234567 Chan Tai Man" csci3180@cse.cuhk.edu.hk
   ```

8. Please submit your assignment using your Unix accounts.

9. An acknowledgement email will be sent to you if your assignment is received. **DO NOT** delete or modify the acknowledgement email. You should contact your TAs for help if you do not receive the acknowledgement email within 5 minutes after your submission. **DO NOT** re-submit just because you do not receive the acknowledgement email.

10. You can check your submission status at

    `http://course.cse.cuhk.edu.hk/~csci3180/submit/hw4.html`.

11. You can re-submit your assignment, but we will only grade the latest submission.

12. Enjoy your work :>