CS2104 – Programming Languages August 30, 2010
School of Computing
National University of Singapore

Problem Set 2

Semester 1, 2011/12

Due: ***September 15, 23:59*** **7 marks**

Submission: In IVLE, in the cs2104 workbin, you will find a folder called `“Homework submissions”.` In that folder, there are currently ***3 subfolders:*** **`PS2P01,…,PS2P03`**. The last two digits of the folder name indicate the solution that is to be submitted into that folder: the solution to *Question 1* into **`PS2P01`**, and so on (that is, you need to submit 3 separate solutions to 3 questions). A solution should consist of a ***single text file*** that can be compiled, or loaded into the interpreter of interest and executed. You should provide as much supplementary information about your solution as you can, ***in the form of program comments***.

Problem 1 **[2 mark, submit to `PS2P01`]**

Write a Prolog predicate definition that counts the number of occurrences of an operator inside an expression. For instance, the query:

```
?- count(a+b*c-(2+3*4)/(5*(2+a)+(b+c)^f((d-e)*(x-y))), *, C).
```

would count the number of occurrences of operator `*` in the expression given as the first argument. The return value should be bound to the variable passed as the third argument. For the query given above, the answer should be `C=4`.

Problem 2 **[2 marks, submit to PS2P02]**

Consider the language of arithmetic expressions with the operators `+, -, *,` and `/`, where the operands are either numeric constants, or Prolog atoms denoting mathematical variables. Thus, the Prolog term:

```
(x+2)*(a-x)
```

stands for the mathematical function *f(x)=(x+2)(a-x),* where *a* would be symbolic constant.

Write a Prolog program that computes the derivative of such an expression. For instance, the query:

```
?- derive((x+2)*(a-x),x,D).
```

should derive the expression given as the first argument with respect to the variable given as the second argument. The result should be placed in the variable given as the third argument. For the query above, the answer should be

```
C = (1+0)*(a-x)+(x+2)*(0-1)
```

Do not perform any arithmetic simplification in your solution to this problem. Simplification is the topic of the next problem.

Problem 3 **[3 marks, submit to PS2P03]**

Write a Prolog program that performs arithmetic expression simplification. Your program should at least eliminate multiplications by 0 and 1, and additions with 0. Ideally, it should also convert any subexpression containing only constants into the value of that expression, and convert multiplications with -1 into the negative of the multiplicand. For instance, the expression

```
(1+0)*(a-x)+(x+2)*(0-1)
```

could be simplified into:

```
a-x-(x+2)
```

Further simplifications may include distribution of high-precedence operators, and grouping of terms that contain the same variable. The expression above could be further simplified into:

```
a-2*x-2
```

Further Practice Problems

These problems are for your own individual practice. Solutions are not to be submitted, and will not be marked. You are, however, allowed to post your solutions in the forum for comparison and discussion. Good posts will earn marks.

Further Practice Problem 1

Considering arithmetic expressions over numeric constants and variables (represented by Prolog atoms), write a Prolog predicate that renames a given variable in an expression. For instance, the call:

```
?- rename(2*x+y-a/3^x, x, y, E).
```

should return the answer:

```
E = 2*y+y-a/3^y
```

Further Practice Problem 2

Write a Prolog program that returns the list of the first N prime numbers.

Further Practice Problem 3

Write a Prolog program that computes the prefix sums of a list of integers. In the list of prefix sums, the `N`th element is the sum of the first `N` elements of the original list.

Further Practice Problem 4

Write a Prolog procedure that takes in a list of integers, and succeeds if the numbers in the list are consecutive, and fails otherwise.