CS4212 – Compiler Design
School of Computing
National University of Singapore

September 1, 2012

Problem Set 1

Semester 1, 2012/13

Due: ***September 16, 23:59*** **Marks: 5**

Submission: In IVLE, in the cs4212 workbin, you will find a folder called "Homework submissions". In that folder, there are currently ***5 subfolders:*** **PS1P01,…,PS1P05**. 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 **PS1P01**, and so on (that is, you need to submit 5 separate solutions to 5 questions). A solution should consist of a ***single text file*** that can be loaded into the SWI-Prolog environment and executed. You should provide as much supplementary information about your solution as you can, ***in the form of Prolog comments***.

You are allowed to work in teams of 2 on each problem set. Each team should make a single submission for each problem. Please indicate clearly who the authors of the submission are, in a comment at the beginning of the file. Though not expressly required, it is recommended that you change teammates for subsequent problem sets, so you get to make more friends and interact with more people.

Problem 1 [1 mark, submit to PS1P01]

Write a Prolog program that counts the number of assignment statements of a program written in the toy language defined in Lecture 2 and Recitation 1.

Sample interaction:

```
?-  P = (
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
          if ( b > a )
              then { b = b - a }
              else { a = a - b } ;
          i = i + 1 ;
      } ), count_assig(P,N).
N = 6
P = ... % value of P will be printed as well, but we save space here
```

Problem 2 [2 mark, submit to PS1P02]

Write a Prolog program that returns a list containing all the identifier names that occur in a program written in the toy language defined in Lecture 2 and Recitation 1. Each identifier must appear exactly once in the list; however, the order of the list elements is not important.

Sample interaction:

```
?-  P = (
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
          if ( b > a )
              then { b = b - a }
              else { a = a - b } ;
          i = i + 1 ;
      } ), var_list(P,Vars).

Vars = [a,b,i] % may appear in different order
P = ... % value of P will be printed as well, but we save space here
```

Problem 3 [3 mark, submit to PS1P03]

Suppose we have the following restriction to our toy language: **`else`** branches are no longer allowed. In other words, **`if`** statements are restricted to having only a **`then`** branch. Write a Prolog program that converts a program of the toy language into a program of the restricted toy language, as defined above. The converted program must perform exactly the same computation and terminate with the same values for all the original variables. However, you may introduce new variables in your converted program; the values of these variables upon program termination are allowed to be arbitrary. (Hint: test program equivalence with the interpreter built in Recitation 1).

Problem 4 [1 mark, submit to PS1P04]

Write a Prolog program that renames a variable of a toy program. The variable to be renamed, and its new name, must be given as arguments to the main predicate of the program. The new name must be completely fresh; you must check that it is not the name of a variable already occurring in the program (Hint: use your solution to Problem 2 to test for that).

Sample interaction:

```
?-  P = (
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
          if ( b > a )
```

```
            then { b = b - a }
            else { a = a - b } ;
        i = i + 1 ;
    } ), change_var_name(P,NewP,a,x). % change a into x

NewP = ( % unfortunately, Prolog will not pretty print output
    x = 240 ; b = 144 ; i = 0 ;
    while ( x \= b ) do {
        if ( b > x )
            then { b = b - x }
            else { x = x - b } ;
        i = i + 1 ;
    } )

P = ... % value of P will be printed as well, but we save space here
```

Problem 5 [3 mark, submit to PS1P05]

A ***constant boolean condition*** is which no variables occur. For instance, **`(2<3)`** is a constant boolean condition. Such conditions evaluate to the same truth value, irrespective of the values of the variables in the program. An **`if`** statement whose boolean condition is constant, can be replaced completely by either its **`then`** branch (if the boolean condition evaluates to true), or by its **`else`** branch (if the boolean condition evaluates to false). Write a Prolog program that simplifies toy programs in the manner described above.

Sample interaction:

```
?-  P = (
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
          if ( 2+3+(2>1) > 1*1/1 )
              then { b = b - a }
              else { a = a - b } ;
          i = i + 1 ;
      } ), simplify_if(P,NewP).

NewP = ( % unfortunately, Prolog won't pretty-print output
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
              { b = b - a }
          i = i + 1 ;
      } )

P = ... % value of P will be printed as well, but we save space here
```

```
?-  P = (
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
          if (2+3+(2>1) < 1*1/1 )
              then { b = b - a }
              else { a = a - b } ;
          i = i + 1 ;
      } ), simplify_if(P,NewP).

NewP = ( % unfortunately, Prolog won't pretty-print output
      a = 240 ; b = 144 ; i = 0 ;
      while ( a \= b ) do {
              { a = a - b }
          i = i + 1 ;
      } )

P = ... % value of P will be printed as well, but we save space here
```