National Technical University of Athens
School of Electrical & Computer Engineering
Department of Computer Science
`http://courses.softlab.ntua.gr/pl2/`

Programming Languages II

Solutions to the exercises are to be handed in to the instructors in electronic form. Deadlines are firm. You may hand in at most one late exercise.

Exercise 3 Pythagorean Triples

Due date: 2/12/2010

In this exercise we will concentrate on the problem of finding Pythagorean triples. As stated in the relevant page of Wikipedia,[1] a *Pythagorean triple* consists of three positive integers a, b, and c, such that $a^2 + b^2 = c^2$. The task is to write a program which takes as input an integer $n \geq 3$ and finds all Pythagorean triples where all numbers are up to n (i.e., $c \leq n$). By "finds" we mean it puts them in some data structure such as a list.

In a functional language such as Haskell, there is a straightforward, one-line solution using list comprehensions. This program though, although elegant and easy to understand, it's not the most efficient that one can write. One can come up with better solutions and little thought and the Wikipedia page probably can give you ideas on how to write better such programs.

But this assignment is not a competition for the fastest program! Its goal is different. You are asked to write *many* programs that solve this problem, in C and at least three functional language implementations (Haskell, SML/NJ, MLton, OCaml, Erlang, Clean, Lisp, Scheme, ...), and see what you can learn from the performance characteristics and differences of these programs regarding the decisions that these language implementations have made and the compiler optimizations that they perform (or clearly do not perform!). It is precisely this knowledge that we are after here. Also, the idea is to look at these questions from the perspective of a reasonably well-educated programmer (i.e., find this information *without* looking at the assembly code or the source of the compiler).

You will clearly see there are various parameters that one can experiment with:

- Use a list comprehension vs. writing explicit recursive functions, tail recursive and not
- Use 32-bit integers (`Int` in Haskell) vs. arbitrary precision integers (`Integer` in Haskell, `IntInf` in SML, ...)
- Write loop invariant code in loop iterations or in recursive functions and seeing whether the compiler is clever enough to factor it out.
- ... (The list stops here because an important part of the assignment is to encourage you to use your imagination.)

What you should submit: a report with your programs and findings about the implementation of functional languages compared with each other and with C. The report can be in English.

One more thing: it's OK (in fact, it's encouraged!) to work in pairs on this assignment.

[1] `http://en.wikipedia.org/wiki/Pythagorean_triple`