OTE: Ohjelmointitekniikka
Programming Techniques

course homepage: `http://www.cs.uku.fi/~mnykanen/OTE/`

Week 51/2008

Exercise 1. In last week's Exercise 5, the idea was to develop a binary version for the addition algorithm $x + y$ we learned already at primary school. Let us now do the same for multiplication $x \cdot y$. Here x and y are again represented and manipulated as Boolean arrays.

(a) Modify last week's addition algorithm into one which adds $y \cdot 2^j$ into z. Here z is also represented and manipulated as a Boolean array, and j is an index.

(b) Express the postcondition of the multiplication algorithm in terms of x, the algorithm in part (a), and j.

(c) Manipulate this postcondition into the initialization, guard and invariant for the outer loop of the multiplication algorithm.

(d) Develop the complete multiplication algorithm from them.

Exercise 2. The familiar string searching problem is to find out where the short target string appears within the longer corpus text. For example, the target `cab` appears as the underlined part of the corpus `abrada`<u>`cab`</u>`ra`.

If the target t and corpus c are arrays, then this problem is informally "find the index position p within c such that the section $c[p \dots p + \ell - 1]$ consists of the same elements as t in the same order, where ℓ is the length of t".

(a) Give the expression for ℓ.

(b) Formalize this informal problem statement logically into a postcondition. For simplicity, assume that the corpus c does indeed contain the target t somewhere, so that such a p is guaranteed to exist.

We shall lift this simplifying assumption as Exercise 3 below.

(c) Manipulate this postcondition into the initialization, guard, and invariant of the outer search loop.

(d) Develop the complete algorithm from them.

Exercise 3. Let us now extend our solution to Exercise 2 above to the case where the target t *might not* appear anywhere within the corpus c, so that such a p might not exist.

(a) Extend the formal postcondition to allow also this.

(b) Manipulate this extended postcondition into the initialization, guard, and invariant of the extended outer search loop.

(c) Develop the complete extended algorithm from them.

(It is often useful to proceed like this in search problems: Develop first a solution which finds the answer, when it is guaranteed to exist. Then extend this solution to the case when it is not.)

Exercise 4. *Casting out nines* is a handy rule for determining whether a given big number $n \in \mathbb{N}$ is divisible by 9 or not: Simply add together the decimal digits of n, and see whether this much smaller sum s is divisible by 9 or not. For example, $n = 23571$ *is* divisible by 9, since $s = 2 + 3 + 5 + 7 + 1 = 18 = 2 \cdot 9$ is. But $n' = 23475$ *is not*, since $s' = 2 + 3 + 4 + 7 + 5 = 21 = 2 \cdot 9 + 3$ is not.

(a) Develop a loop for computing s from n.

(b) Develop an algorithm for casting out nines based on this loop.

(c) Explain why and how it even suffices to keep $0 \leq s < 9$ during this algorithm.

Exercise 5. Use the algorithmic idea from Exercise 4 to develop an algorithm for casting out *threes*.