CSL100: Introduction to Computers and Computer Science
I semester 2013-14
Instructors: Subhashis Banerjee and S. Arun-Kumar

Week 4: 19-23 Aug 2013

1. **Constant time algorithms.** An algorithm is constant time if it can be expressed as a simple expression or containing only names of quantities each of which can be evaluated by a *constant time algorithm*. Typical examples of constant time algorithms are those problems for which closed form solutions are known – solutions to linear or quadratic equations in one variable, sum of the first n terms of an arithmetic or geometric progression etc.

   (a) Let's begin with some basic integer operations in SML that many students find confusing.

      i. What is SML's value for the following expressions?

      A. `7 div ~3 =`_____ `7 mod ~3 =`_____

      B. `Int.quot (7, ~3) =`_____ `Int.rem (7, ~3) =`_____

      C. `~7 div 3 =`_____ `~7 mod 3 =`_____

      D. `Int.quot (~7, 3) =`_____ `Int.rem (~7, 3) =`_____

      E. `~7 div ~3 =`_____ `~7 mod ~3 =`_____

      F. `Int.quot (~7, ~3) =`_____ `Int.rem (~7, ~3) =`_____

      ii. For any two integers a, b with $b \neq 0$, let
      - $d = a$ `div` b and $m = a$ `mod` b
      - $q =$ `Int.quot`(a, b) and $r =$ `Int.rem`(a, b)

      A. Give an identity relating a, b, d and m. _____

      B. Give an identity relating a, b, q and r. _____

      C. Give conditions relating a, b, d, q, m and r which show that d and q are **not always** equal. Similarly m and r are **not always** equal.

      _____

      _____

   (b) Dates may be represented as positive integers in the form *ymmdd*, where y is any positive integer, *mm* denotes the two digit month and *dd* denotes the two digit date within the month. For example, the date *29 Jan 2000* would then be the integer 20000129. The first date in this representation is the integer 10101 (*1 Jan 1*). Also there are many integers which do not represent a valid date *Examples: 1, 10, -256, 451301, 291131, 19990229 etc.*

   Write *technically complete* SML functions for the following, where in each case the argument is an *integer* possibly representing a date.

i. `leapyear (k)`: which determines whether the year in the date is a leap year. In particular, it is not a leap year if it ends with '00' and is not divisible by 400.

ii. `feb29 (k)`: determines whether the date is *29 February* of some year (need not be a leap year)

iii. `maxdays (m)`: which determines the maximum number of days in a month.

iv. `valid_date (k)`: determines whether `k` is the integer representation of a valid date.

(c) Define a function *nextdate* : $\mathbb{N} \to \mathbb{N}$ which computes the integer representation of the next day. Assuming that SML can represent any positive integer define a function `nextdate` to compute the integer representation of the next day. Be careful to raise the exception `invalid_date` whenever necessary.

(d) It is rather irksome to be limited to only dates as positive integers. It is possible to extend it also to negative integers which represent dates preceding the start of the Christian era.

i. Rename the function `valid_date (k)` that you have previously defined to `valid_pos_date (k)`. What is a suitable algorithmic definition for `valid_neg_date (k)` which takes negative integers as dates before the Christian era? Now write a new function `valid_date (k)` which uses both `valid_pos_date (k)` and `valid_neg_date (k)` and determines the validity of a given integer as representing a date.

ii. Alexander the Great died on 10 June 323 BCE. What is the date of his death as a negative integer?

(e) Define a function *prevdate* : $\mathbb{N} \to \mathbb{N}$ which calculates the date of the previous day to a given valid date.

(f) In 1882 Christian Zeller devised a constant time algorithm for calculating the day of week for any date. Check out the Wikipedia article at `http://en.wikipedia.org/wiki/Zeller's_congruence` which describes the congruence. Also note that the following statement made under the subheading "Implementation in software"

> The formulas rely on the mathematician's definition of modulo division, which means that $-2 \ mod \ 7$ is equal to positive 5. Unfortunately, the way most computer languages implement the remainder function, $-2 \ mod \ 7$ returns a result of -2. So, to implement Zeller's congruence on a computer, the formulas should be altered slightly to ensure a positive numerator.

is false for Standard ML.

i. Implement this constant time algorithm as an SML function `zeller`, assuming "Sunday" to be 0 and "Saturday" is 6.

ii. Does your program yield the correct result for today's date?

iii. What day of the week was 10101?

iv. On what day of the week did Alexander die?

2. **Linear time algorithms.** An algorithm is a linear time one if its running time is a linear function of the length of the input.

(a) Given an arbitrary positive integer, using only integer arithmetic operations, define a function *reverse* : $\mathbb{N} \to \mathbb{N}$ which yields the number obtained be reversing the digits of the given number. Write an SML function `reverse` which implements your function *reverse*.

(b) Can you use the function *reverse* to determine whether two numbers are palindromes of each other i.e. reversing the digits of one yields the other? Write a function *palindrome* : $\mathbb{N}^2 \to \mathbb{N}$ which determines whether a pair of numbers are palindromes. Implement it in SML.

(c) Use the functions *nextdate* and *prevdate* to define linear time algorithms for computing the integer representation of the day m days *after a given date* and m days *before a given date*

(d) Without using the functions *nextdate* or *prevdate* can you define *constant time* algorithms (perhaps inspired by Zeller!) to find the integer representation of the day m days *after* a given date. Your algorithm should ensure that if m is negative then it computes the day m days *before* the given date.