Institut for Matematik og Datalogi — August 28, 2014
Syddansk Universitet — JFB

Introduction to Computer Science E14 – Discussion Sections 1 – Week 36

1. Trace the execution of the following pseudocode on inputs $M = 5$ and $N = 14$. What values do the variables M, N, and Z get during the execution. More generally, what does the pseudocode calculate?

```
**CALC**($M, N$):
{ Input: two positive integers $M, N$ }
{ Output: CALC($M, N$) }

    $Z \leftarrow 0$
    **while** $M \neq 0$ **do**
        $Z \leftarrow Z + N$
        $M \leftarrow M - 1$

    return($Z$)
```

2. Recall that the greatest common divisor algorithm can be written in pseudocode as follows:

```
**GCD**($M, N$):
{ Input: two positive integers $M, N$ }
{ Output: gcd($M, N$) }

    $A \leftarrow \max(M, N)$
    $B \leftarrow \min(M, N)$

    $Q \leftarrow A$ div $B$
    $R \leftarrow A - (Q \cdot B)$
    **while** $R \neq 0$ **do**
        $A \leftarrow B$
        $B \leftarrow R$
        $Q \leftarrow A$ div $B$
        $R \leftarrow A - (Q \cdot B)$
    return($B$)
```

- What is the largest number of times you can make the algorithm execute when $M = 34$ and $N \leq 34$? As a challenge, try to generalize this to determine which pairs of integers are "bad" for the greatest common divisor algorithm, where a pair is bad if the algorithms must perform a lot of steps relative to how large the numbers are. (One expects more steps for larger numbers.)
- Using pseudocode as in section 5.2 of the textbook, remove the max and min functions and use an **if-then-else** structure instead, along with the symbol $>$ to test if one value is larger than another or not. Your new pseudocode should produce the same result in all cases.
- Using pseudocode as in section 5.4 of the textbook, remove the **while** structure and use a **repeat-until** structure instead. Your new pseudocode should produce the same result in all cases.

3. Write up an algorithm for calculating $N! = N \cdot (N-1) \cdots 2 \cdot 1$.

4. Consider the Boolean formula $(a \wedge b \wedge \bar{c}) \vee (\bar{a} \wedge b \wedge c)$. Write a truth table for it Draw a circuit for it. Try to write a smaller formula and circuit for it. (Note that $\bar{x}$ is the same as $\neg x$.)

   (Note that this formula is in what is called Disjunctive Normal Form (DNF). The parts inside parentheses are called clauses. Each clause consists of some variables and/or complements of variables (literals), combined with ANDs, and the clauses are combined with ORs.)

5. Design a circuit, using only AND, OR and NOT gates which takes three bits as input and outputs a 1 if the input has at least two zeros, and a 0 otherwise.

6. Design and draw a circuit containing only AND and XOR gates (with at most two inputs) which takes three bits as input and outputs a 1 if the input has at least two ones, and a 0 otherwise.

(In the student resources for the course textbook, under the Activities for Chapter 1, there is a simulator for logic circuits which you could use to check your circuit. It is time consuming to use, though.) As an extra challenge, try to do it so that there is only one AND gate, though more XOR gates. (Minimizing the number of AND gates can be useful in some cryptographic applications.)