Aalto University, School of Science
Department of Computer Science

CS-A1120 Programming 2 Spring 2026
Exam May 21, 2026, at 17:00–20:00

Use of material, calculators or other electronic devices is not allowed in the exam.
You may answer in Finnish, Swedish, or English.

Part I:
Provide *only answers* to questions 1–10. No motivation. All answers on one sheet.
1 point per correct answer. Enumerate answers clearly. Unclear answer or hedging ⇒ 0 points.

1. Give the decimal value 55 in binary notation. Use most significant digit first order.

2. Let `val k: Int = 3`. What is the result of the Scala expression `(0x0FA3CCC0 | (0x3 << k))`? Give your answer as a hexadecimal number.

3. The circuit in Figure 1 has four binary valued inputs (`a`, `b`, `c`, `d`) and one binary valued output (`x`). Out of the total 16 possible assignments of 0 or 1 each to the four inputs, how many of these result in `x` evaluating to 1?

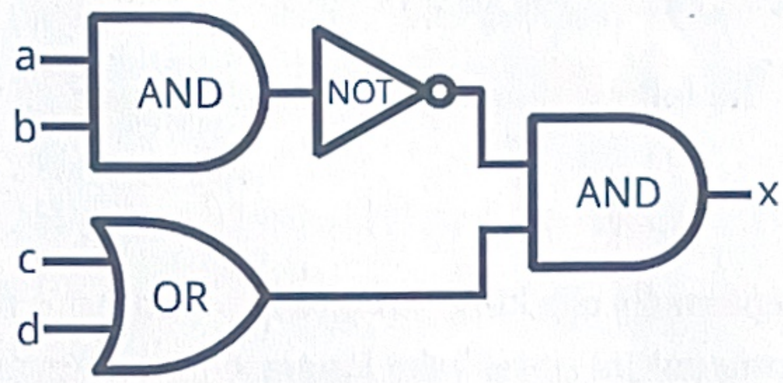


Figure 1: Circuit

4. Consider the Armlet program in Listing 1. Assuming it is executed from the first line: at which of the two `hlt`'s does the program stop? (Remember: `cmp` is compare, and `bne` is branch if not equal.)

5. For the same program in Listing 1: What is the value in register `$7` when it reaches `hlt`?

6. Suppose that $f(n) = 7n^2+50n$ and $g(n) = 3n\log n+5n^2$. Which of the following hold?

a) $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$
b) $f(n) = \mathcal{O}(g(n))$ and $f(n) \neq \Omega(g(n))$
c) $f(n) \neq \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$
d) $f(n) \neq \mathcal{O}(g(n))$ and $f(n) \neq \Omega(g(n))$

7. Consider the Scala function in Listing 2. What value does the call `myFun(6)` return?

8. Which operation constitutes a *tail call* of `myFun` in Listing 2?
a) `if` b) `else` c) `+` d) `myFun(n - 1)` e) `n - 1` f) `-`

9. What value does the Scala expression in Listing 3 produce?

10. Given the Scala code in Listing 4, what is the result of running `p("C") + p("Z")`?

```
1 mov $7,3137
2 and $7,$7,1
3 cmp $7,0
4 bne >h2
5 hlt
6 @h2:
7 hlt
```

Listing 1: Armlet

```
1 def myFun(n: Int): Int =
2   if n <= 0 then 0
3   else n + myFun(n - 1)
```

Listing 2: myFun

```
1 List("yay", "soon",
2   "summer", "break")
3   .foldLeft(0)((acc, x) =>
4     acc + x.length)
```

Listing 3: Expression

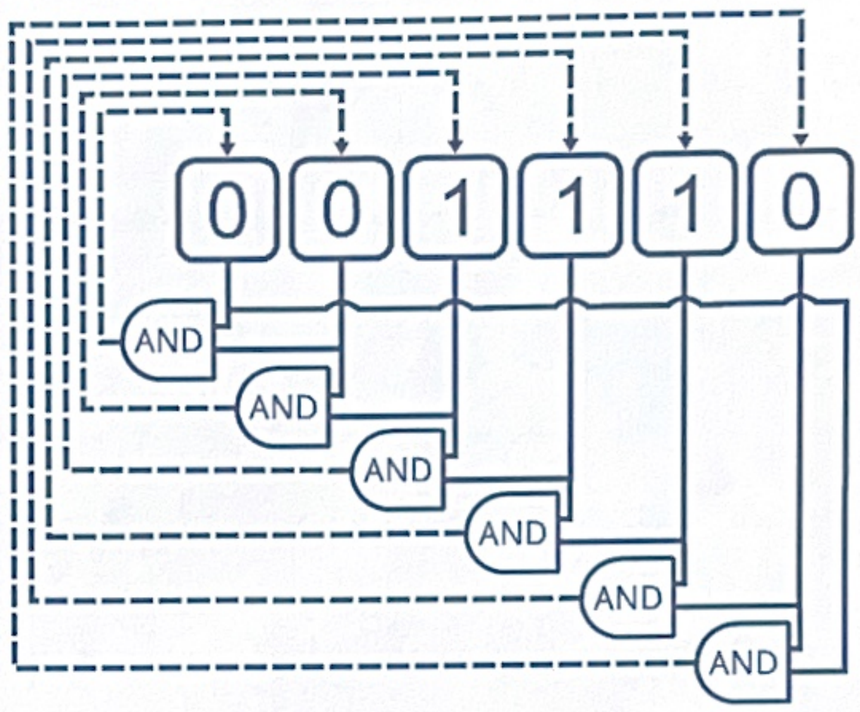
```
1 def cmb(x: String)(y: String):
2     String = x + y
3 val p = cmb("A")
```

Listing 4: Some code

Exam continues on next page

Part II:
Motivate all answers in this section (questions 11-13) clearly and unambiguously.
Your answers should be clear, well-structured and concise.
Answers in this section will only be marked if you achieve at least 5 points on Part I.

11. Consider the sequential logic circuit to the right.
It depicts a 6-bit feedback register with initial state 0 0 1 1 1 0.
Each gate takes the current bit and its right neighbour as inputs, with the rightmost gate using the leftmost bit as its second input. Feedback (dashed lines) is always to the current bit. That is the first gate feeds back to the first bit, the second gate to the second bit, and so on.



Answer the following, motivating each answer.

a) Repeatedly clocking this circuit will drive it towards a steady state. Which steady state, and why? (2p.)
b) Some initial states take longer than others to reach this steady state with this circuit. Which initial state(s) require the most clock cycles? How many ticks, and why? (3p.)
c) By replacing some AND gates with OR gates, the updated circuit can be made to converge to a different steady state. Specify the gate type (AND or OR) for each of the six gate positions so that the circuit settles at 1 1 0 0 1 0 starting from the initial state in the figure. (3p.)
d) Is your solution to c unique? (2p.)

$a = a \wedge b$
$b = b \wedge c$
$c = c \wedge d$
$d = d \wedge e$
$e = e \wedge f$
$f = f \wedge a$

Exam continues on next page

12. Consider the following Scala code and description of its *intended* functionality:

```
1  import scala.collection.mutable.ArrayBuffer
2  def mm(s: ArrayBuffer[Double]):
3    (Double,Double) =
4      var a = s(0)
5      var b = s(0)
6      for x <- s do
7        if x<a then a = x
8        if x>b then b = x
9      (a,b)
10 def nls(s: ArrayBuffer[Double]):
11   ArrayBuffer[Double] =
12     var i = 0
13     while i < s.length do
14       val (a,b) = mm(s)
15       val d = b - a
16       s(i) = if math.abs(d) < 1e-7 then 0.0
17              else (s(i) - a) / d
18       i = i + 1
19     s
```

The intended functionality is for the function `nls` to map the values in the parameter sequence `s` (implemented as an `ArrayBuffer[Double]`) to the range $[0,1]$ such that

- the smallest value(s) map to 0.0,
- the largest value(s) map to 1.0,
- intermediate values are scaled proportionally.

The relative order of elements in the sequence does not change. If the difference between the largest and smallest values is less than 10^{-7} in absolute value, all values map to 0.0.
For example,
`nls(ArrayBuffer(1.0, 2.0, 3.0))`
should result in
`ArrayBuffer(0.0, 0.5, 1.0)`, and
`nls(ArrayBuffer(2.0, -1.0, 5.0))`
should result in
`ArrayBuffer(0.5, 0.0, 1.0)`.

This implementation is faulty: it contains a bug causing erroneous results for some inputs.

a) Identify and explain the bug, including an example of valid input for `nls` that demonstrates the incorrect behaviour. Here, "valid input" means input that the function is intended to accept without throwing an exception (you may assume `s` is always non-empty). (3p.)

b) Show how the implementation above can be fixed by explaining the necessary changes to the code required for it to work as intended. (2p.)

c) Does your fix change the expected runtime (in terms of big-$\mathcal{O}$ complexity) of `nls`? Why/why not? (Remember to motivate your answer.) (2p.)

d) Explain the concept of *side-effects* in the context of functions. Are the original implementation of `mm` and `nls` given above side-effect-free? Why/why not? (3p.)

13. Imagine a generalised deck of cards made up of k suits, with each suit valued from 1 to n.
For example, in a standard western deck of 52 playing cards $k = 4$ (clubs, diamonds, hearts, spades), and $n = 13$. Each card is represented as a pair of integers (value, suit). In a complete deck, there are $n \times k$ cards, one of each value-suit combination. *However, in your deck, exactly one card is missing.*
However, all cards in your deck have been sorted from lowest to highest by value but not by suit. The suits can be in any order for each value. That is, first come all cards valued 1 (in some order of suits), then all cards valued 2 (in some, possibly different, order of suits), and so on up to n.
Provide an algorithm with run-time efficiency strictly less than $\mathcal{O}(nk)$ for finding the missing card in the deck, and derive its big-$\mathcal{O}$ time-efficiency in terms of n and k.
You may assume that accessing a card in the deck and checking/comparing value and suit are all $\mathcal{O}(1)$ operations.
Your algorithm should be structured and unambiguous. You may describe it using clear verbal language or as pseudocode. Give the tightest (slowest-growing) run-time efficiency you can justify for your algorithm. (8p.)