ICS-A1120 Programming 2

Exam May 29th, 2015

Petteri Kaski (050-430 0948) and Tommi Junttila (050-430 0861)

The use of electronic devices is not allowed in the exam.

Solve as many as possible of the following twelve problems. Each problem is worth the same amount of points.

1. What is the decimal (base 10) representation of the unsigned binary integer 10101110 (given in the most-significant-bit-first order)?

   What is the 8-bit unsigned binary representation (in the "most significant bit first" order) of the base 10 integer 115?

2. Consider an integer `x` whose type is `Int`. Give a Scala-expression that returns an integer with type `Int` such that the bit position 23 has value 1, and all other bit positions agree with the corresponding bit positions in `x`. (In an integer of type `Int`, the least significant bit is at position 0, the most significant bit is at position 31.)

   Reminder: The bitwise NOT of a word can be obtained with the operator `~`, the bitwise AND of two words can be obtained with the operator `&`, and the bitwise OR in turn with the operator `|`. The bit positions in a word can be shifted to the left (towards more significant bits) with the operator <<, and to the right (towards less significant bits) with the operators >> and >>>.

3. Construct a Boolean circuit according to the following specification, using the gates NOT (`!`), OR (`||`), and/or AND (`&&`). The circuit has two input elements `a`, `b`, and one output `r`. The input elements may be independently assigned either to the value `false` or to the value `true`. The value of the output `r` must be `true` exactly when the inputs `a` and `b` disagree (exactly one input has the value `true` and exactly one input has the value `false`). Present the circuit either as a Scala-expression or as a drawing, where the input elements, the output, and the types of the gates (NOT, OR, AND) have been clearly indicated.

4. Let us consider the following simple assembly-language program. What is the value stored in register $0 when the program stops (halt)?

```
 mov $0, 0       # move 0 to $0
 mov $1, 1       # move 1 to $1
@loop:
 cmp $1, 4       # compare $1 and 4
 add $0, $0, $1  # add $1 to $0
 add $1, $1, 1   # add 1 to $1
 bbw >loop       # branch to 'loop' if < holds in most recent comparison
 hlt             # halt
```

5. Describe in one or two sentences what the function below (written in the imperative programming style) computes. Give a corresponding function written in the functional programming style.

```
def f(s: IndexedSeq[Int], p: Int => Boolean): Seq[Int] = {
 var i = 0
 val r = scala.collection.mutable.ArrayBuffer[Int]()
 while(i < s.length) {
  val v = s(i)
  if(p(v)) {
   r += v*v // Recall: r += x appends x to r
  }
  i = i + 1
```

```
  }
  r
}
```

You may use the classes and methods in the `scala.collections.immutable` package.

Hint: the methods `filter` and `map` may be useful.

6. For the both functions given below, use the big $\mathcal{O}$ notation to describe the growth rate of the run-time with respect to the length n of the argument sequence `s`. The growth rate functions you provide should have as slow growth as possible.

   Recall that indexed access and evaluating the `length` method are constant-time operations in the `Array` type.

   Function 1:

```
def myFunc1(s: Array[Int]): Int = {
  require(s.length >= 2)

  def helper(): Int = {
    var r = s(0)
    var i = 1
    while(i < s.length) {
      if(s(i) < r)
        r = s(i)
      i = i + 1
    }
    r
  }

  var i = 1
  var result = s(0)
  while(i < s.length) {
    if(s(i) > helper() && s(i) < result)
      result = s(i)
    i = i + 1
  }
  result
}
```

   Function 2:

```
def myFunc2(s: Array[Int]): Int = {
  require(s.length >= 2)

  var i = 1
  var result = s(0)
  var smallest = s(0)
  while(i < s.length) {
    if(s(i) < smallest) {
      result = smallest
      smallest = s(i)
    } else if(s(i) < result) {
      result = s(i)
    }
    i = i + 1
  }
```

```
  result
}
```

7. Describe in your own words (that is: without giving any program code) how binary search can be applied to find a zero point of a continuous function f when we know two values l and u such that $l \leq u$, $f(l) \leq 0$ and $f(u) \geq 0$.

   For which values x does your binary search compute the value of f when $f = x^2 - 4x - 32$, $l = 0$ and $u = 32$?

8. Consider the function `f` given below. Describe in one sentence (with words, no code allowed) what the function computes. Illustrate the call stack when `f` is called with the parameter `l` set to the value `List(1,4,6)` and `c` to 3.

```
def f(l: List[Int], c: Int): Int = {
  def inner(r: List[Int], s: Int): Int = {
    if(r.isEmpty) s
    else inner(r.tail, if(r.head >= c) s+r.head else s)
  }
  inner(l, 0)
}
```

9. Let us recall the linked lists data structure presented in the lecture notes. Is the method `sum` below in tail-recursive form? If not, explain why this is the case and also give a corresponding tail-recursive version in functional programming style.

```
abstract class LinkedList[A] {
  def isEmpty: Boolean
  def head: A
  def tail: LinkedList[A]
  def sum(p: A => Int): Int = {
    this match {
      case Nil() => 0
      case Cons(h, t) => p(h) + t.sum(p)
    }
  }
}
case class Nil[A]() extends LinkedList[A] {
  def isEmpty = true
  def head = throw new java.util.NoSuchElementException("head of empty list")
  def tail = throw new java.util.NoSuchElementException("tail of empty list")
}
case class Cons[A](val head: A, val tail: LinkedList[A]) extends
    LinkedList[A] {
  def isEmpty = false
}
```

10. Let `t` be an arbitrary string (`String`), for example

```
val t = "soossi"
```

    Present a function (or a one-line expression) that takes as parameter a string `t` and returns an array (`Array`) of strings consisting of, in increasing lexicographic order, all the non-empty suffixes of `t`.

    For the string given above the desired return value would thus be

```
val result = Array("i", "oossi", "ossi", "si", "soossi", "ssi")
```

Reminder: Each string has an accompanying method `drop(k)` that returns a copy of the string with the `k` first characters dropped (deleted from the string). The method `sorted` for a sequence-type collection (type `Seq` or any of its subtypes) returns a version of the collection where the elements have been sorted to increasing order. For strings, the default order is lexicographic order.

11. Let us consider the following multithreaded program from the start of its execution until its termination:

```
import scala.concurrent._
import ExecutionContext.Implicits.global

object mystery {
  def main(args: Array[String]) {
    val fa = Future { Thread.sleep(1000); 123 }
    val fb = Future { Thread.sleep(2000); 345 }
    val fc = for { a <- fa; b <- fb } yield { Thread.sleep(1000); a + b }
    Await.ready(fc, duration.Duration.Inf)
    Await.ready(fa, duration.Duration.Inf)
    Await.ready(fb, duration.Duration.Inf)
  }
}
```

Measured from the start, how long does it take for the program to terminate? Let us assume that we have available an unbounded amount of parallel computing resources and there is no substantial delay to start, join, or otherwise synchronize threads. Justify your answer.

Reminder: The method call `Thread.sleep(m)` stops the calling thread for `m` milliseconds (`m/1000.0` seconds). The method call `Await.ready(f, duration.Duration.Inf)` waits for the completion of the value of the future `f` and returns when the value is complete.

12. Let us consider an array of real-valued `k`-dimensional vectors:

```
val data : Array[Array[Double]]
val k : Int
require(data.forall(_.length == k))
```

Suppose we are also given one real-valued `k`-dimensional vector:

```
val query : Array[Double]
require(query.length == k)
```

Write a program that determines a position in the array `data` that contains a vector whose Euclidean distance to the vector `query` is as small as possible. (If there are multiple such vectors, any position $0, 1, \ldots,$ `data.length` $- 1$ of such a vector is a valid output from the program.)

```
val nearestNeighborPosition : Int = ???
```

Hint: It suffices to consider the square of the Euclidean distance, which is obtained by taking the sum of squares of the coordinate-wise differences of the two vectors.