ICS-A1120 Ohjelmointi 2

Tentti 29.5.2015

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

Elektronisten apuvälineiden käyttö tentissä ei ole sallittua.

Ratkaise mahdollisimman monta seuraavista 12 keskenään samanarvoisesta tehtävästä.

1. Mikä on etumerkittömän binääriluvun 10101110 (eniten merkitsevä bitti ensin -järjestyksessä) 10-kantainen esitys?

   Mikä on 10-kantaisen luvun 115 8-bittinen etumerkitön binääriesitys (eniten merkitsevä bitti ensin -järjestyksessä)?

2. Tarkastellaan kokonaislukua `x`, jonka tyyppi on `Int`. Anna Scala-lauseke, joka palauttaa kokonaisluvun tyyppiä `Int`, jonka bittipaikka 23 on arvossa 1, ja muiden bittipaikkojen arvot ovat kuten kokonaisluvussa `x`. (Kokonaisluvussa tyyppiä `Int` vähiten merkitsevä bitti on paikassa 0, eniten merkitsevä bitti paikassa 31.)

   Muistiapu: Biteittäinen EI yhdestä sanasta voidaan laskea operaattorilla `~`, biteittäinen JA kahden sanan välillä voidaan laskea operaattorilla `&` ja biteittäinen TAI puolestaan operaattorilla `|`. Sanan bittipaikkoja voi siirtää vasemmalle (kohti enemmän merkitseviä bittejä) operaattorilla `<<`, ja oikealle (kohti vähemmän merkitseviä bittejä) operaattoreilla `>>` ja `>>>`.

3. Rakenna seuraavaa määritelmää vastaava Boolen piiri käyttäen portteja EI (NOT, `!`), TAI (OR, `||`) ja/tai JA (AND, `&&`). Piirillä on kaksi sisääntuloa `a`, `b`, ja yksi ulostulo `r`. Sisääntulot voi toisistaan riippumattomasti asettaa joko arvoon `false` (epätosi) tai `true` (tosi). Piirin ulostulon `r` arvon on oltava tosi täsmälleen silloin jos sen sisääntulot `a`,`b` ovat päinvastaisissa arvoissa (täsmälleen yksi sisääntulo on arvossa tosi ja täsmälleen yksi sisääntulo on arvossa epätosi). Esitä piiri joko Scala-lausekkeena tai piirroksena, jossa sisääntulot, ulostulo ja porttien tyypit (EI, TAI, JA) on selkeästi merkitty.

4. Tarkastellaan seuraavaa yksinkertaista konekieliohjelmaa. Mikä on rekisterin $0 arvo ohjelman pysähtyessä (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. Kuvaile yhdellä tai kahdella lauseella, millaisen arvon alla oleva, imperatiivisella tyylillä kirjoitettu funktio laskee. Anna sitä vastaava funktionaalisella tyylillä kirjoitettu ohjelma.

```
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
```

```
}
```

Saat käyttää `scala.collections.immutable`-kirjaston luokkia ja metodeja.

Vihje: ainakin `filter` ja `map` saattavat olla käteviä metodeja.

6. Anna kummallekin alla olevalle funktiolle ajoaikaraja suhteessa argumenttisekvenssin `s` pituuteen (merkitse sitä muuttujalla n) käyttäen $\mathcal{O}$-notaatiota. Antamiesi ajoaikarajafunktioiden tulee olla mahdollisimman hitaasti kasvavia.

   Muistathan, että `Array`-tyypillä indeksoitu elementin hakeminen ja `length`-metodi ovat vakioaikaisia operaatioita.

   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. Kerro lyhyesti (antamatta ohjelmakoodia), kuinka puolitushakua voidaan käyttää löytämään jatkuvan funktion f nollakohta kun tiedetään arvot l ja u siten, että $l \leq u$, $f(l) \leq 0$ ja $f(u) \geq 0$.

   Missä pisteissä x puolitushakusi laskee funktion f arvon kun $f = x^2 - 4x - 32$, $l = 0$ ja $u = 32$?

8. Tarkastellaan alla olevaa funktiota `f`. Kerro sanallisesti, millaisen arvon `f` laskee. Anna suorituksen kutsupino kun funktiota `f` kutsutaan argumentin `l` ollessa `List(1,4,6)` ja argumentin `c` ollessa 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. Tarkastellaan oppimateriaalissa esitettyjä linkitettyjen listojen luokkia. Onko alla oleva metodi `sum` häntärekursiivisessa muodossa? Jos ei ole, kerro miksei ole ja anna funktionaalisella tyylillä toteutettu vastaava häntärekursiivinen versio.

```
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. Olkoon `t` mielivaltainen merkkijono (`String`), esimerkiksi

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

    Esitä funktio (tai yhden rivin lauseke), joka palauttaa parametrina annetun merkkijonon `t` kaikista mahdollisista ei-tyhjistä loppuosista koostuvan taulukon (`Array`), jonka sisältö on järjestetty kasvavaan sanakirjajärjestykseen (lexicographic order). Ylläolevalle merkkijonolle haluttu tulos olisi siis

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

    Muistiapu: Merkkijonon metodi `drop(k)` palauttaa merkkijonon, josta on pudotettu `k` merkkiä alusta pois. Sekvenssityyppisen kokoelman (tyyppi `Seq` tai mikä tahansa sen alityyppi) metodi `sorted` palauttaa kokoelmasta kasvavaan järjestykseen järjestetyn version. Merkkijonoille oletusjärjestys on sanakirjajärjestys.

11. Tarkastellaan seuraavaa monisäikeistä ohjelmaa sen käynnistymisestä pysähtymiseen asti:

```
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)
  }
}
```

Kauanko kestää ohjelman käynnistymisestä, että ohjelma pysähtyy? Oletetaan, että käytössä on rajattomasti rinnakkaislaskentaresursseja, ja että säikeiden käynnistämisestä (start), säikeisiin liittymisestä (join), tai muusta synkronoinnista säikeiden välillä ei aiheudu merkittäviä viiveitä. Perustele vastauksesi.

Muistiapu: Metodikutsu `Thread.sleep(m)` odotuttaa kutsuvaa säiettä `m` millisekuntia (`m/1000.0` sekuntia). Metodikutsu `Await.ready(f, duration.Duration.Inf)` odottaa futuurin `f` arvon valmistumista ja palaa kun arvo on valmistunut.

12. Tarkastellaan taulukkoa reaaliarvoisia `k`-ulotteisia vektoreita:

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

Olkoon annettuna lisäksi yksi reaaliarvoinen `k`-ulotteinen vektori:

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

Kirjoita ohjelma, joka määrittää missä paikassa taulukossa `data` on vektori, joka on Euklidisen etäisyyden mielessä lähimpänä vektoria `query`. (Jos tällaisia vektoreita on useita, minkä tahansa tällaisen vektorin paikka $0, 1, \ldots,$ `data.length` $- 1$ kelpaa.)

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

Vihje: Riittää tarkastella Euklidisen etäisyyden neliötä, joka saadaan laskemalla yhteen vektorien koordinaattien erotuksien neliöt.