Mallitehtäviä tenttiin valmistautumista varten

Huomaathan, että elektronisten apuvälineiden käyttö tentissä ei ole sallittua.

Muistathan ilmoittautua tenttiin ajoissa Oodissa!

1. Mikä on etumerkittömän binääriluvun 10010111 (eniten merkitsevä bitti ensin -järjestyksessä) desimaaliesitys?

2. Tarkastellaan kokonaislukua `x`, jonka tyyppi on `Int`. Anna Scala-lauseke, joka palauttaa kokonaisluvun tyyppiä `Int`, jonka bittipaikka 7 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 JA kahden sanan välillä voidaan laskea operaattorilla `&`, 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 kolme sisääntuloa `a`, `b`, `c`, ja yksi ulostulo `r`. Sisääntulot voi toisistaan riippumattomasti asettaa joko arvoon `false` (epätosi) tai `true` (tosi). Piirin ulostulon `r` arvo on sen sisääntulojen enemmistön arvo. 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, 10      # compare $1 and 10
  beq >done       # branch to 'done' if most recent comparison is ==
  add $0, $0, $1  # add $1 to $0
  add $1, $1, 1   # add 1 to $1
  jmp >loop       # jump to 'loop'
@done:
  hlt             # halt
```

5. Anna alla olevaa, imperatiivisella tyylillä kirjoitettua ohjelmaa vastaava funktionaalisella tyylillä kirjoitettu ohjelma (paluuarvon ei tällöin luonnollisestikaan tarvitse olla tyyppiä `ArrayBuffer[Int]`, mikä tahansa `Seq[Int]`-piirteen toteuttava tyyppi käy):

```
def f[T](s: Seq[T], p: T => Boolean): Seq[Int] = {
  var i = 0
  val r = scala.collection.mutable.ArrayBuffer[Int]()
  for(v <- s) {
    if(p(s(i)))
      r += i
    i += 1
  }
  r
}
```

   Saat käyttää (tai olla käyttämättä) `scala.collections.immutable`-kirjaston luokkia ja metodeja. Vihje: `zipWithIndex` saattaa olla avuksi (esimerkiksi `List('a','b','c').zipWithIndex` palauttaa `List((a,0), (b,1), (c,2))`).

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

   Vihje: muista ottaa huomioon erityisesti `while`-lauseen ehdossa ja anonyymeissa funktioissa jokaisella suorituskerralla suoritettavat komennot.

   Ovatko kaikki alla olevat funktiot samoja eli laskevatko ne samalla argumentilla saman arvon?

   Funktio 1:

```
def myFunc1(l: List[Int]): Int = l.count(_ == l.max)
```

   Funktio 2:

```
def myFunc2(l: List[Int]): Int = {
  var m: Option[Int] = None
  var r = 0
  var i = 0
  while(i < l.length) {
    if(m.isEmpty || l(i) > m.get) {
      m = Some(l(i))
      r = 1
    } else if(l(i) == m.get) {
      r += 1
    }
    i += 1
  }
  r
}
```

   Funktio 3:

```
def myFunc3(l: List[Int]): Int = {
  var m: Option[Int] = None
  var r = 0
  l.foreach(v => {
    if(m.isEmpty || v > m.get) {
      m = Some(v)
      r = 1
    } else if(v == m.get) {
      r += 1
    }
  })
  r
}
```

   Funktio 4:

```
def myFunc4(l: List[Int]): Int = {
  val m = l.max
  l.count(_ == m)
}
```

7. Tarkastellaan oppimateriaalissa esitettyjä linkitettyjen listojen luokkia. Onko alla oleva metodi `length` 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 length: Int = {
    this match {
      case Nil() => 0
      case Cons(_, t) => 1 + t.length
    }
  }
}
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
}
```

8. 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`). Ylläolevalle merkkijonolle haluttu tulos olisi siis

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

Muistiapu: Merkkijonon metodi `drop(k)` palauttaa merkkijonon, josta on pudotettu `k` merkkiä alusta pois.

9. Tarkastellaan seuraavaa monisäikeistä ohjelmaa sen pysähtymiseen asti:

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

```
object mystery {
  def main(args: Array[String]) {
    for(i <- 0 until 2) {
      future { print("1") }
      future { print("2") }
      future { print("3") }
      future { print("4") }
    }
  }
}
```

Mitkä seuraavista ovat ohjelman mahdollisia tulosteita?

(a) 11 (b) 12341234 (c) 43214321 (d) 11223344 (e) 22142134 (f) 33333333 (g) 12134243

10. 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)
```

Määritä millä järjestysnumerolla taulukossa `data` on vektori, joka on Euklidisen etäisyyden mielessä lähimpänä vektoria `query`. (Jos tällaisia vektoreita on useita, minkä tahansa tällaisen vektorin järjestysnumero $0, 1, ...,$ `data.length` $- 1$ kelpaa.)

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

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