

Kirjoita jokaiseen paperiin oma nimi, oppilasnumero, tutkinto-ohjelma, kurssikoodi ja kurssin nimi, päivämäärä, sali, palauttamiesi paperien lukumäärä sekä *allekirjoituksesi*. Numeroi palauttamasi paperit juoksevalla numeroinnilla. Tentissä ei saa käyttää mitään ylimääräisiä apuvälineitä.

1) Kymmenen kysymystä (10 p)

Tämä tehtävä on tentin pakollinen osa, josta on saatava vähintään 5/10 pistettä, jotta loput tentistä tarkistetaan. Tämä tehtävä ei kuitenkaan yksistään riitä tentin läpäisyyn. Käytä aikaa perustelujen miettimiseen ja esittämiseen. Viittaa perusteluissa ohjelmakoodin rivinumeroihin, jos mahdollista.

Alla on annettu kaksi algoritmia (`insertionSort` ja `selectionSort`), jotka molemmat järjestävät parametrina annetun taulukon `a` alkiot suuruusjärjestykseen pienimmästä suurimpaan. Algoritmien toimintaa testattiin lisäämällä niihin `print`-lauseita, jotka tulostavat silmukkamuuttujien `i` ja `j` arvoja algoritmin suorituksen aikana. Lue ensin kaikki kysymyskohdat vastaamatta niihin ja sen jälkeen vertaile annettuja koodinpätkiä erittäin huolella. Vastaa vasta tämän jälkeen kysymyksiin.

```
1 def insertionSort(a):
2     n = len(a)
3     for i in range(1, n):
4         print("i = ", i)
5         temp = a[i]
6         j = i
7         while j > 0 and a[j - 1] > temp:
8             print("j = ", j)
9             a[j] = a[j - 1]
10            j = j - 1
11            a[j] = temp

1 def selectionSort(a):
2     n = len(a)
3     for i in range(0, n-1):
4         print("i = ", i)
5         min = i
6         for j in range(i+1, n):
7             print("j = ", j)
8             if a[j] < a[min]:
9                 min = j
10            temp = a[i]
11            a[i] = a[min]
12            a[min] = temp
```

- Mitä arvoja silmukkamuuttujat `i` ja `j` saavat, kun `insertionSort`-algoritmia kutsutaan taulukolla `a = [10, 11, 12, 13]`?
- Mitä arvoja silmukkamuuttujat `i` ja `j` saavat, kun `selectionSort`-algoritmia kutsutaan taulukolla `a = [10, 11, 12, 13]`?
- Mitä arvoja silmukkamuuttujat `i` ja `j` saavat, kun `insertionSort`-algoritmia kutsutaan taulukolla `a = [14, 14, 14, 14]`?
- Mitä arvoja silmukkamuuttujat `i` ja `j` saavat, kun `selectionSort`-algoritmia kutsutaan taulukolla `a = [14, 14, 14, 14]`?
- Mitä arvoja silmukkamuuttujat `i` ja `j` saavat, kun `insertionSort`-algoritmia kutsutaan taulukolla `a = [18, 17, 16, 15]`?
- Mitä arvoja silmukkamuuttujat `i` ja `j` saavat, kun `selectionSort`-algoritmia kutsutaan taulukolla `a = [18, 17, 16, 15]`?
- Mikä on `insertionSort`-algoritmin parhaan tapauksen aikavaativuus? Perustele.
- Mikä on `selectionSort`-algoritmin parhaan tapauksen aikavaativuus? Perustele.
- Mikä on `insertionSort`-algoritmin pahimman tapauksen aikavaativuus? Perustele.
- Mikä on `selectionSort`-algoritmin pahimman tapauksen aikavaativuus? Perustele.

2) Terminologiaa (2p + 2p + 2p + 2p)

Määrittele seuraavat käsitteet (4 x 1p). Huom! Anna jokaisesta myös *esimerkki* (4 x 1p).

- a) Abstrakti tietotyyppi (Abstract Data Type)
- b) Binääripuu (Binary Tree)
- c) Rekursioyhtälö (Recurrence Relation)
- d) Pino (Stack)

3) Hajautus (2 p + 4 p + 2 p)

a) Selitä hajautuksen (*hashing*) peruseriaatteet.

b) Vertaile hakurakenteisiin liittyvien operaatioiden aikakompleksisuuksia. Mitkä operaatiot ovat tehokkaampia hajautusmenetelmissä kuin tasapainotetuissa hakupuissa? Entä mitkä operaatiot ovat tehottomampia hajautusmenetelmissä kuin tasapainotetuissa hakupuissa? Perustele ja pohdi sekä keskimääräistä että pahinta tapausta.

c) Arvioi hajautusmenetelmien hyviä ja huonoja puolia. Minkälaisiin sovelluksiin kurssilla esitetyt perushajautusmenetelmät soveltuvat tai eivät sovellu hyvin?

4) Prioriteettijonot (2 p + 3 p + 3 p)

a) Määrittele käsite *prioriteettijono* (*priority queue*).

b) Esitä välivaiheittain, miten alkio 7, 2, 25, 1, 10, 23, 14, 20, 3, 5, 6, 15 ja 13 voidaan lisätä yksi kerrallaan alun perin tyhjäan binääriheikkoon (*binary heap*). Kekohto on ”isä pienempi kuin lapsensa”.

c) Esitä välivaiheittain, miten *linearisessa ajassa* toimiva *BuildHeap*-algoritmi (joka tunnetaan myös nimillä *FixHeap* ja *Bottom-Up Heap Construction*) toimii, jos se saa syötteenä taulukon, jossa on alkio 7, 2, 25, 1, 10, 23, 14, 20, 3, 5, 6, 15 ja 13. Kekohto on ”isä pienempi kuin lapsensa”.

5) Verkkoalgoritmit (4 p + 4 p)

a) Selitä joko *Primin* tai *Dijkstran* algoritmin toiminta sanallisesti. Käytä apuna sopivaa esimerkkiä.

b) Toimivatko em. algoritmit, jos painotetussa verkossa on negatiivisia kaaria? Perustele näkemyksesi molempien algoritmien osalta erikseen. Jos jompikumpi algoritmi toimii mielestäsi virheellisesti, anna siitä esimerkki.