ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

27. toukokuuta 2026

Yleistä

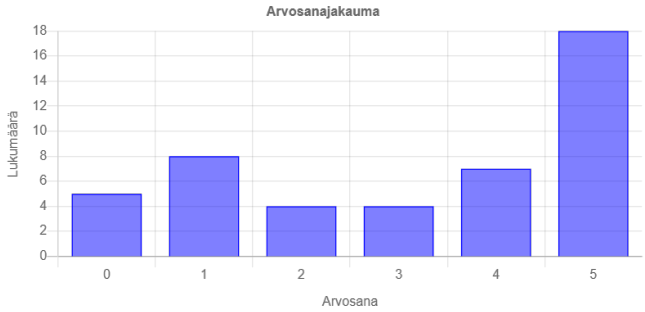
Tentti oli pistekeskiarvon 14,9 (keskihajonta 6,4) perusteella vaikeudeltaan keskitasoa. Demohyvitysten kanssa keskiarvo oli 19,5. Huomaa, että demopisteet on laskettu tentistä saatujen pisteiden päälle, ja arvosana lasketaan vasta sen jälkeen. Opiskelijan omat tehtävät ovat nähtävissä TIMissä alkuperäisellä tenttisivulla. Uusintojen ajankohdat löydät kurssin kotisivulta.

Tehtävä	Teki	Keskiarvo	Min	Max	SD
T1	46	4,7	0	6	1,6
T2	45	4,6	0	6	1,9
T3	42	3,2	0	6	2,7
T4	38	3,4	0	6	2,3
Yht (ennen demopisteitä)	46	**14,9**	0	24	6,4

JYVÄSKYLÄN YLIOPISTO

Arvosteluasteikko ja arvosanajakauma

Pistemäärä (inklusiivinen alaraja)	Arvosana
11	1
14,5	2
17	3
19,5	4
22	5



Tentin arvosanajakauma.

Tehtävä 1 (6 p.)

Oikeat vastaukset on lihavoitu.

Alla on monivalintakysymyksiä. Kuhunkin kysymykseen on täsmälleen yksi oikea vastaus. Arviointi: Oikea vastaus, 1 p. Väärä vastaus, -1 p. Ei vastattu, 0 p. Tehtävän minimipistemäärä on 0 p.

- Kysymys 1.1: Alla on kaksi esimerkkiä sijoituslauseista. Oletetaan, että `Funktio`-aliohjelma sekä muuttuja `y` ovat olemassa ja koodissa ei ole virheitä.

  ```
  int x = 3;
  x = 5 + Funktio(4) + y;
  ```

  Mikä seuraavista väitteistä pitää paikkansa?

  - Sijoituslause vertaa kahta arvoa ja palauttaa tulokseksi totuusarvon.
  - **Sijoituslauseessa oikean puolen lausekkeen arvo tallennetaan vasemmalla olevaan muuttujaan.**
  - Sijoituslause muuttaa aina muuttujan tyypin vastaamaan sijoitettavan arvon tyyppiä.
  - Sijoituslauseessa vasemman ja oikean puolen järjestyksellä ei ole merkitystä.

- Kysymys 1.2: Mikä seuraavista väitteistä pitää paikkansa ehtolauseesta?

  - `if`-lauseen ehto saa olla mikä tahansa kokonaisluku.
  - `else`-lohko suoritetaan aina `if`-lohkon jälkeen.
  - `else`**-lohko suoritetaan vain silloin, kun siihen liittyvän** `if`**-lauseen ehto on epätosi.**
  - `else if` -lohkoja voi käyttää vain silmukan sisällä.

- Kysymys 1.3: Mikä seuraavista väitteistä pitää paikkansa seuraavasta silmukasta?

  ```
  int summa = 0;
  for (int i = 1; i <= 3; i++)
  {
      summa += i;
  }
  ```

  - Silmukan runko suoritetaan kaksi kertaa.
  - **Silmukan päätyttyä muuttujan** `summa` **arvo on 6.**
  - Silmukan runkoa ei suoriteta kertaakaan, koska `i` saa aluksi arvon 1.
  - Silmukka ei pääty koskaan, koska muuttujan `summa` arvo kasvaa.

- Kysymys 1.4: Mikä seuraavista väitteistä pitää paikkansa taulukosta?

  - Taulukkoon voidaan lisätä uusia alkioita niin, että sen `Length` kasvaa automaattisesti.
  - **Taulukon jokaisella alkiolla on indeksi, jonka avulla kyseiseen alkioon voidaan viitata.**
  - Taulukon alkioiden tyyppi voi vaihtua alkioittain saman taulukon sisällä.
  - Taulukon viimeiseen alkioon viitataan indeksillä, joka on sama kuin taulukon `Length`.

- Kysymys 1.5: Mikä seuraavista väitteistä pitää paikkansa seuraavan koodin suorituksen jälkeen?

```
int a = 7;
int b = 2;
int tulos = a / b;
```

  - Muuttujan `tulos` arvo on 3.5.
  - **Muuttujan `tulos` arvo on 3.**
  - Koodi ei käänny, koska kokonaislukumuuttujia ei voi jakaa keskenään.
  - Muuttujan `tulos` arvo on 4, koska tulos pyöristetään lähimpään kokonaislukuun.

- Kysymys 1.6: Mikä seuraavista väitteistä pitää paikkansa seuraavan koodin suorituksesta?

```
public static int Tuplaa(int luku)
{
    luku = 2 * luku;
    return luku;
}

int arvo = 4;
int tulos = Tuplaa(arvo);
```

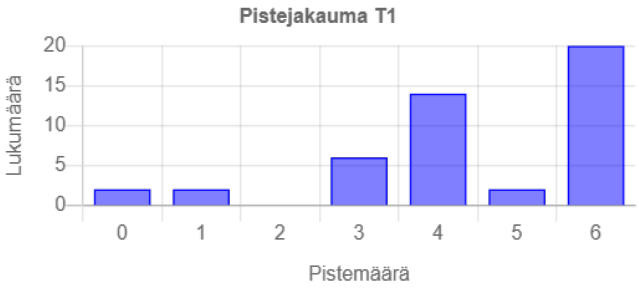
  - Muuttujan `arvo` arvoksi tulee 8 ja muuttujan `tulos` arvoksi tulee 8.
  - Muuttujan `arvo` arvoksi tulee 4 ja muuttujan `tulos` arvoksi tulee 4.
  - **Muuttujan `arvo` arvoksi tulee 4 ja muuttujan `tulos` arvoksi tulee 8.**
  - Koodi ei käänny, koska parametrin `luku` arvoa muutetaan aliohjelman sisällä.

Tehtävä 2 (6 p.)

Oikeat vastaukset on lihavoitu.

Kuhunkin kysymykseen on täsmälleen yksi oikea vastaus. Arviointi: Oikea vastaus, 2 p. Väärä vastaus, -1 p. Ei vastattu, -1 p. Tehtävän minimipistemäärä on 0 p.

Alla on aliohjelma `X` sekä kolme siihen liittyvää kysymystä. Funktion ja muuttujien nimet on tarkoituksella kirjoitettu lyhyiksi.



Tehtävän 1 pistejakauma.

```
public static int X(string s)
{
    int a = 0;
    int b = 0;

    for (int i = 0; i < s.Length; i++)
    {
        if (s[i] == '(')
        {
            b++;
            if (b > a) a = b;
        }
        else if (s[i] == ')')
        {
            if (b > 0) b--;
        }
    }

    return a;
}
```

Seuraavat syöte-tulos-testit kuvaavat `X`-aliohjelman toimintaa:

Syöte	Tulos
`""`	0
`"abc"`	0
`"()"`	1
`"(())"`	2
`"((()))"`	3
`"()()"`	1
`"(()())"`	2
`"a(b(c)d)e"`	2
`")("`	1
`"))abc"`	0
`"((("`	3
`"))))"`	0
`"())(()"`	2
`"(()"`	2
`")(()"`	2
`"abc((x)y(z))"`	2
`"((a)(b(c)))"`	3

- Kysymys 2.1: Mikä seuraavista kuvauksista sopii parhaiten `X`-aliohjelmalle? `X`...
  - palauttaa merkkijonon pituuden.
  - palauttaa merkkijonossa olevien sulkujen kokonaismäärän.
  - **palauttaa suurimman vasempien sulkumerkkien määrän, joka on sillä hetkellä ilman vastaparia, eli oikeaa sulkumerkkiä.**
  - palauttaa ensimmäisen oikean sulkumerkin indeksin.
- Kysymys 2.2: Taulukossa ei ole seuraavaa testitapausta. Mikä olisi sen tulos?

Syöte	Tulos
`"a((b(c(d)e)))"`	?

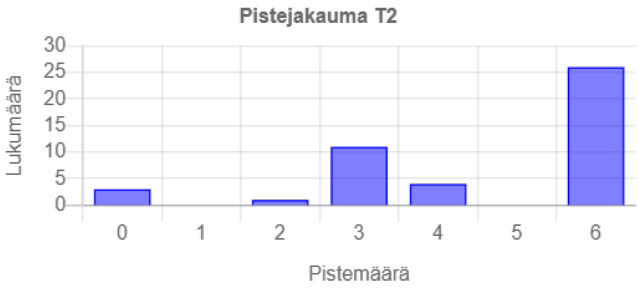
  - 0
  - 1
  - 3
  - **4**
- Kysymys 2.3: Mikä seuraavista väitteistä pitää paikkansa?

- Aliohjelma palauttaa viimeisen vasemman sulkumerkin indeksin.
- Aliohjelma palauttaa aina `0`, jos merkkijonossa on yksikin oikea sulkumerkki.
- Aliohjelma tarkistaa, löytyykö kullekin vasemmalle sulkumerkille sopiva vastapari.
- **Aliohjelma huomioi vain merkit `'('` ja `')'`; muilla merkeillä ei ole lopputuloksen kannalta merkitystä.**

Malliratkaisu

Sama toiminnallisuus voidaan kirjoittaa selkeämmin esimerkiksi näin:

```
/// <summary>
/// Palauttaa merkkijonossa yhtä aikaa avoinna olevien vasempien
/// sulkumerkkien suurimman määrän.
/// Vain merkit '(' ja ')' vaikuttavat laskentaan.
/// Ylimääräiset oikeat sulkumerkit ohitetaan.
/// </summary>
public static int SuurinAvoinnaOlevienSulkujenMaara(string teksti)
{
    int suurinMaara = 0;
    int nykyinenMaara = 0;

    for (int i = 0; i < teksti.Length; i++)
    {
        if (teksti[i] == '(')
        {
            nykyinenMaara++;
            if (nykyinenMaara > suurinMaara)
            {
                suurinMaara = nykyinenMaara;
            }
        }
        else if (teksti[i] == ')')
        {
            if (nykyinenMaara > 0)
            {
                nykyinenMaara--;
            }
        }
    }

    return suurinMaara;
}
```



Tehtävän 2 pistejakauma.

Tehtävä 3 (6 p.)

Metsäretken reitti on mitattu peräkkäisissä tarkastuspisteissä. Jokaisesta pisteestä tiedetään korkeus metreinä. Haluamme selvittää, kuinka pitkä reitin pisin yhtäjaksoinen nousuosuus on.

Tee funktio `PisinNousuosuus`, joka ottaa parametrina `int`-taulukon `korkeudet` ja palauttaa pisimmän yhtenäisen aidosti nousevan osuuden pituuden.

Esimerkiksi taulukko `korkeudet = [3, 1, 2, 3, 0, 4, 5, 6]` tarkoittaa, että ensimmäisen tarkastuspisteen korkeus on 3 metriä, toisen 1 metri, kolmannen 2 metriä ja niin edelleen. Reitti siis ensin laskee korkeudesta 3 korkeuteen 1, nousee sitten korkeuksiin 2 ja 3, laskee korkeuteen 0 ja nousee lopuksi korkeuksiin 4, 5 ja 6.

Yhtenäinen osuus tarkoittaa peräkkäisiä tarkastuspisteitä tältä samalta reitiltä. Osuus on aidosti nouseva, jos jokainen seuraava korkeus on edellistä suurempi. Jos kaksi peräkkäistä korkeutta ovat samat tai seuraava korkeus on pienempi kuin edellinen, nousuosuus katkeaa.

Paluuarvo tarkoittaa tarkastuspisteiden lukumäärää.

Esimerkki 1: Syöte: `korkeudet = [3, 1, 2, 3, 0, 4, 5, 6]`, paluuarvo: `4`. **Selitys:** Pisimmät nousuosuudet ovat `[1, 2, 3]` ja `[0, 4, 5, 6]`. Näistä jälkimmäisen pituus on 4.

Esimerkki 2: Syöte: `korkeudet = [1, 2, 2, 3]`, paluuarvo: `2`. **Selitys:** Tasainen kohta `2 -> 2` katkaisee aidon nousun, joten pisimmän nousuosuuden pituus on 2. Osat `[1, 2]` ja `[2, 3]` ovat molemmat aidosti nousevia, ja näistä kummankin pituus on 2.

Jos taulukko on tyhjä, funktion tulee palauttaa 0. Jos taulukossa on vähintään yksi alkio, pisimmän nousuosuuden pituus on vähintään 1.

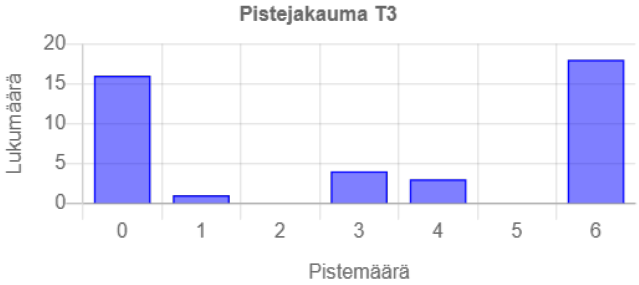
Malliratkaisu

```
public static int PisinNousuosuus(int[] korkeudet)
{
    if (korkeudet.Length == 0) return 0;

    int pisin = 1;
    int nykyinen = 1;

    for (int i = 1; i < korkeudet.Length; i++)
    {
        if (korkeudet[i] > korkeudet[i - 1])
```

```
        {
            nykyinen++;
        }
        else
        {
            nykyinen = 1;
        }

        if (nykyinen > pisin)
        {
            pisin = nykyinen;
        }
    }

    return pisin;
}
```

Ratkaisussa pidetään kirjaa nykyisen nousuosuuden pituudesta ja tähän mennessä löydetystä pisimmästä nousuosuudesta. Kun seuraava korkeus on edellistä suurempi, nykyinen osuus pitenee. Muuten uusi osuus alkaa nykyisestä tarkastuspisteestä.

Arvostelu pohjautui automaattiseen arvioon Liitteessä A esitettyjen arvostelukohteiden/testitapausten kautta. Pisteet pyöristettiin alaspäin lähimpään kokonaislukuun.



Tehtävän 3 pistejakauma.

Tehtävä 4 (6 p.)

Lintuharrastaja kirjaa kevään aikana muistiin, montako muuttolintua hän näkee kunakin havaintopäivänä. Haluamme selvittää, monenako päivänä havaintomäärä on uusi ennätys.

Tee funktio `EnnatyshavaintojenMaara`. Funktio ottaa parametrina `int`-taulukon `havainnot`, jossa jokainen alkio kertoo yhden päivän lintuhavaintojen määrän.

Esimerkiksi taulukko `havainnot = [3, 5, 4, 8, 8, 10]` tarkoittaa, että ensimmäisenä päivänä nähtiin 3 lintua, toisena 5 lintua, kolmantena 4 lintua ja niin edelleen.

Päivä on uusi ennätyspäivä, jos kyseisen päivän havaintomäärä on suurempi kuin kaikkina sitä edeltävinä päivinä. Ensimmäinen havaintopäivä on aina uusi ennätys, jos taulukossa on vähintään yksi alkio. Jos havaintomäärä on täsmälleen sama kuin aiempi ennätys, se ei ole uusi ennätys.

Funktion tulee palauttaa uusien ennätyspäivien lukumäärä.

Esimerkki 1: Syöte: `havainnot = [3, 5, 4, 8, 8, 10]`, paluuarvo: `4`. **Selitys:** Uudet ennätykset syntyvät havaintomäärillä 3, 5, 8 ja 10. Toinen arvo 8 ei ole uusi ennätys, koska ennätys oli jo 8.

Esimerkki 2: Syöte: `havainnot = [7, 6, 5, 4]`, paluuarvo: `1`. **Selitys:** Ensimmäinen päivä on ennätys. Sen jälkeen havaintomäärät vain pienenevät, joten uusia ennätyksiä ei tule.

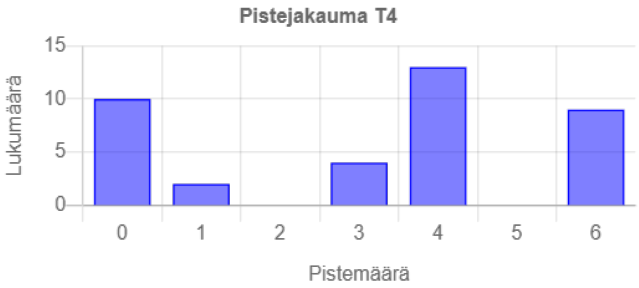
Jos taulukko on tyhjä, funktion tulee palauttaa 0. Voit olettaa, että kaikki taulukon arvot ovat vähintään 0.

Malliratkaisu

```
public static int EnnatyshavaintojenMaara(int[] havainnot)
{
    if (havainnot.Length == 0) return 0;

    int ennatys = havainnot[0];
    int maara = 1;

    for (int i = 1; i < havainnot.Length; i++)
    {
        if (havainnot[i] > ennatys)
        {
            ennatys = havainnot[i];
            maara++;
        }
    }

    return maara;
}
```

Ratkaisussa pidetään kirjaa tähän mennessä suurimmasta havaintomäärästä. Kun vastaan tulee sitä suurempi arvo, syntyy uusi ennätys ja laskuria kasvatetaan yhdellä.

Arvostelu pohjautui automaattiseen arvioon Liitteessä A esitettyjen arvostelukohteiden/testitapausten kautta. Pisteet pyöristettiin alaspäin lähimpään kokonaislukuun.



Tehtävän 4 pistejakauma.

A Arviointikohteet

A.1 Tehtävä 3

#	Syöte	Tulos	Pisteet	Virheilmoitus
1	`[]`	`0`	0.5	Testaa tyhjän reitin. Tyhjässä taulukossa ei ole yhtään tarkastuspistettä.
2	`[7]`	`1`	0.5	Testaa yhden tarkastuspisteen reitin. Yksittäinen piste muodostaa pituuden 1 osuuden.
3	`[1, 2]`	`2`	0.5	Testaa kahden pisteen nousun.
4	`[2, 1]`	`1`	0.5	Testaa kahden pisteen laskun. Pisimmän nousuosuuden pituus on vain 1.
5	`[1, 2, 3, 4]`	`4`	0.5	Testaa koko ajan nousevan reitin. Koko taulukko on yksi nousuosuus.
6	`[4, 3, 2, 1]`	`1`	0.5	Testaa koko ajan laskevan reitin. Mikään kahden pisteen peräkkäinen osuus ei ole nouseva.
7	`[1, 2, 2, 3]`	`2`	0.5	Testaa, että tasainen kohta katkaisee nousuosuuden.
8	`[3, 1, 2, 3, 0, 4, 5, 6]`	`4`	0.5	Testaa esimerkin, jossa pisin nousuosuus löytyy reitin lopusta.
9	`[1, 3, 2, 4, 6, 1]`	`3`	0.5	Testaa tapausta, jossa pisin nousuosuus löytyy reitin keskeltä.
10	`[2, 2, 2]`	`1`	0.5	Testaa täysin tasaisen reitin.
11	`[-3, -2, -1, -1, 0, 1]`	`3`	0.5	Testaa korkeuksia, joissa mukana on myös merenpinnan alapuolisia arvoja ja tasainen kohta.
12	`[5, 1, 2, 3, 2, 3, 4, 5]`	`4`	0.5	Testaa useita erillisiä nousuosuuksia. Funktion tulee palauttaa pisimmän osuuden pituus.

A.2 Tehtävä 4

#	Syöte	Tulos	Pisteet	Virheilmoitus
1	`[]`	`0`	0.5	Testaa tyhjän havaintolistan. Ilman havaintopäiviä ei voi syntyä ennätyksiä.

#	Syöte	Tulos	Pisteet	Virheilmoitus
2	`[4]`	`1`	0.5	Testaa yhden havaintopäivän. Ensimmäinen päivä on aina ennätys.
3	`[1, 2, 3, 4]`	`4`	0.5	Testaa koko ajan kasvavan havaintomäärän. Jokainen päivä on uusi ennätys.
4	`[7, 6, 5, 4]`	`1`	0.5	Testaa koko ajan pienenevän havaintomäärän. Vain ensimmäinen päivä on ennätys.
5	`[3, 5, 4, 8, 8, 10]`	`4`	0.5	Testaa tehtävän ensimmäisen esimerkin. Sama ennätysarvo ei riitä uudeksi ennätykseksi.
6	`[2, 2, 3]`	`2`	0.5	Testaa, että sama arvo heti alussa ei muodosta uutta ennätystä.
7	`[5, 1, 2, 3, 6]`	`2`	0.5	Testaa taulukon, jossa ennätys paranee vasta lopussa.
8	`[1, 5, 2, 6, 3, 7]`	`4`	0.5	Testaa taulukon, jossa havaintomäärä vaihtelee mutta ennätys paranee useita kertoja.
9	`[3, 3, 3, 3]`	`1`	0.5	Testaa taulukon, jossa kaikki havaintomäärät ovat samoja.
10	`[0, 0, 1, 0, 2]`	`3`	0.5	Testaa nollia sisältävän taulukon. Ensimmäinen nolla on ennätys ja myöhemmin suuremmat arvot parantavat sitä.
11	`[10, 12, 11, 15, 14, 20, 19]`	`4`	0.5	Testaa suurempia havaintomääriä ja useita ei-ennätyspäiviä ennätysten välissä.
12	`[6, 1, 6, 7, 2, 8]`	`3`	0.5	Testaa, että aiempaa ennätystä pienemmät arvot eivät vaikuta laskuriin.