Ohjelmointi 1, tentti 11.4.2014. Tentaattori Antti-Jussi Lakanen.

Arviointiraportti

Yleistä

Tentti (http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2014-04-11-tentti1.pdf) oli pistekeskiarvon (16.2) perusteella vaikeudeltaan helpohko. Demopisteet lasketaan luonnollisesti vielä tuohon päälle arvosanaa laskettaessa. Tehtäväkohtaiset pistekeskiarvot löytyvät alta kunkin tehtävän kohdalta.

Omasta tehtäväpaperista saa kopion saa Antti-Jussilta, huone C414.2. Jos en ole paikalla, laita sähköpostia tai soita 0500 603111. Papereihin on merkitty virheet, jotka voin käydä kanssasi läpi, tarvittaessa tarkastajan kanssa.

Ensimmäinen uusinta on 16.5.2014, ja toinen uusinta 13.6.2014.

Tehtävä	Teki	Keskiarvo	Tarkastaja
T1	115	4.6	Atte Rautio
T2	118	4.5	Ari Tuhkala
T3	104	3.6	Tuomas Soikkeli
T4	104	4.3	Jouni Potila
T5	37	3.8	Simo Rinne
Yht		**16.2**	

Tehtävä 1

- Teki: 115 opiskelijaa
- Keskiarvo: 4.6 pistettä
- Palautteet: mittaa 4.3, vaikea 2.4, arvio 5.6, aika 34 min
- Tarkastaja: Atte Rautio

1a: Monet eivät ottaneet huomioon, että dokumentaatiot ovat myös kommentteja. Toisaalta, jos oli mainittu että kommentit helpottavat dokumentaation ohella koodin lukemista, en siitä sakottanut.

Osa taas piti `///`-merkintää ainoana tapana kommentointiin eivätkä edes maininneet `//`- tai `/* ... */`-kommentteja.

Pisteytys: Eri kommentointitavat tiedetty: 0,5 pistettä, jos puuttuu niin ei voinut saada täysiä pisteitä.

Dokumentaatiokommenteista 0,5 pistettä, jos oli annettu laajempi esimerkki eri tagien kanssa.

Syitä kommentointiin: 0,5 pistettä per syy. Alla esimerkkejä, joita hyvässä vastauksessa esiintyi (kaikkia ei tarvinnut mainita).

- Kommentointi on koodin sekaan kirjoitettua tekstiä, jota kääntäjä ei huomioi
- Kommentit kertovat ohjelmasta kaiken oleellisen tiedon.
- Koodia on helpompi ymmärtää myös niiden, ketkä eivät ole ohjelmaa itse kirjoittaneet
- Ajan myötä toiminta unohtuu, joten kommentointi helpottaa työskentelyyn palaamista myöhemmin
- Koodia on helpompi päivittää ja käyttää uudelleen
- Myös aliohjelmien ja olioiden, muuttujien jne nimet voidaan lukea osaksi kommentointia!
- XML-kommenttien avulla voidaan tuottaa API-dokumentteja
- Erilaisia kommentteja ovat yksiriviset (`//` -kommentti) ja moniriviset (`/* monta riviä [rivinvaihto] kommentteja */`) sekä dokumentaatiokommentit (`/// <tägi> Kommentti... </tägi>`)

1b: Meni pääsääntöisesti mielestäni todella hyvin. Suurin osa ymmärsi mitä ohjelma tekee, virheet olivat yleensä lähinnä joidenkin tägien puuttumista.

Pieni osa ei kirjoittanut dokumentaatiota ollenkaan, vaan kirjoitti kommentteja suoraan koodin sekaan. Näistä jouduin antamaan 0 pistettä.

Osa kuvasi kaikki aliohjelman muuttujatkin param-tageilla, mutta niin ei kuitenkaan kuulu tehdä.

Pisteytys

- Selvästi ymmärtää mitä ohjelma tekee: 0,5 pistettä.
- parametri-tagi + parametri fiksusti kuvattu: 0,5 pistettä.
- returns-tagi 0,5 pistettä.
- summary-tag 0,5 pistettä.

Jos oli räikeä syntaksivirhe jossain kohtaa, annettiin -0,5 pistettä.

Jotkut käyttivät parametrin merkitsemiseen `<parameter>` -tägiä, vaikka pitäisi olla `<param>` . Tästä ei kuitenkaan sakotettu, koska selkeästi ymmärrettiin, että tägiä pitäisi käyttää.

1b-bonus: Arvostelu oli ankara, koska kyseessä oli kuitenkin bonuspiste. Todella moni sai tästä puoli pistettä.

Osa ei huomannut tehdä taulukoita aliohjelmalle parametriksi, vaan tekstirivit olivat muotoa: `Teht1.SummaaPituudet("kissa", "koira") === 10;`

Pisteytys

- syntaktisesti oikeat testit: 0,5 pistettä.
- tyhjä lista testitapausten joukossa: 0,5 pistettä.

Jos siis testaa vain listoilla, joissa on yksi tai useampi alkio, jää yksi erikoistapaus testaamatta.

1c: Tämäkin meni pääosin todella hyvin. Kaikki osasivat esittelemänsä operaattorit. Osa kuitenkin esitteli esimerkiksi vertailuoperaattoreita loogisten operaattoreiden sijaan.

Kysyttävät operaattorit olivat siis (ks. Moniste luku 13.6, tai MSDN dokumentaatio, luvut 7.10-- 7.12): `!`, `&`, `&&`, `|`, `||`, `^`.

1d: Tämä meni myös hyvin, pääosa virheistä oli pieniä laskuvirheitä.

Pisteytys

0,5 p. per kohta, piti olla täysin oikein.

Tehtävä 2

- Teki: 118 opiskelijaa
- Keskiarvo: 4.5 pistettä
- Palautteet: mittaa 4.7, vaikea 2.5, arvio 5.3, aika 26 min
- Tarkastaja: Ari Tuhkala

Tenttitehtävä meni varsin hienosti ja täyden (tai lähes täyden) pisteen vastauksia oli useita.

Valitettavasti moni oli tehnyt hienosti luokan ja pääohjelman, mutta sinne ainoastaan aliohjelmakutsun esimerkiksi LaskeJaolliset(3,5). Tästä valitettavasti automaattisesti 0 pistettä. Mikäli luokan ja pääohjelman dokumentaatio oli OK, siitä sai lohtuna 0.5 - 1 pistettä. Eikä auttanut, vaikka olisi kommentoinut "//lasketaan aliohjelmassa viidellä ja kolmella jaollisten lukujen summa", mikäli sitä aliohjelmaa ei oltu toteutettu. Tilanne on vähän sama kuin vastaisi esseetentissä kysymykseen ”Mitä Talcott Parsons tarkoittaa rakennefunktionalismilla” että ”Parsons tarkoittaa rakennefunktionalismilla sitä, mitä tenttikirjassa lukee”. Valitettavasti vaikutti siltä, että muutama oli ymmärtänyt tehtävänannon väärin.

Huomiot

- Mikäli dokumentaatio oikein ja tehtäväosiosta ei vähennyksiä.
- Sai käyttää `while` tai `for`-silmukoita
- Kelpasi sekä `if (ehto)` looginen tai (`||`) että `if-else` rakenteet

Pisteytys

- Dokumentaatio oikein, 1 p.
- ei dokumentaatiota, -1 p.
- dokumentaatio miten sattuu tai missä sattuu, -1 p.
- vain luokka tai pääohjelma dokumentoitu, -0.5 p.
- ei versiota ja tekijää, -0.5 p.
- dokumentaatiossa parametreja tms, joita ei ole -0.5 p.

Pääohjelma: pitää tutkia tehtävänannon mukaisesti kolmella tai viidellä jaolliset luvut välillä 0..1000 ja tulostaa niiden summa, 5 p.

- jos algoritmi oli päin seiniä tai koodissa ei mitään järkeä, niin 0 pistettä
- pääohjelman esittely väärin tai luokka puuttuu, -1 p.
- koodin rakenne väärin, -1 p.
- turhia muuttujia / listoja / taulukoita / olioita, -1 p.
- tehty turhaan aliohjelma, -1 p.
- vertailuoperaatio väärin, -1 p.
- vertailuoperaatiossa pieni virhe, -0.5 p.
- silmukka täysin väärin, -1 p.
- silmukassa pieni virhe, -0.5 p.
- summa-muuttujaan sijoittamisessa virhe, -1 p.
- summa-muuttuja alustettu silmukan sisällä, -1 p.
- härvätty breakkien ja returnien kanssa silmukoissa, -1 p.
- ei vähennystä, oliko Mainissa parametri `args` ja oliko sitä kommentoitu
- ei vähennystä, vaikka tutkii nollankin tai luvun tuhat
- ei vähennystä, vaikka `Console.ReadKey();` puuttuisi
- importeista en välittänyt mitään

Malliratkaisu

```
/// @author Ari Tuhkala
/// @version 11.4.2014
///
/// <summary>
/// Tehtävä 2, tentti 11.4.2014
/// </summary>
public class Tentti
{
    /// <summary>
    /// Tulostetaan pääohjelmassa kolmella ja viidellä jaollisten lukujen
    /// summa välillä 0...999.
    /// </summary>
    /// <param name="args">args ei käytössä</param>
    public static void Main(string[] args)
    {
        int summa = 0;

        for (int i = 0; i < 1000; i++)
        {
            if (i % 3 == 0 || i % 5 == 0)
            {
                summa += i;
            }
        }
```

```
        Console.WriteLine(summa);
        Console.ReadKey();
    }
}
```

Tehtävä 3

- Teki: 104 opiskelijaa
- Keskiarvo: 3.6 pistettä
- Palautteet: mittaa 4.1, vaikea 3.8, arvio 4.1, aika 44 min
- Tarkastaja: Tuomas Soikkeli

Malliratkaisu, 3a

```
/// <summary>
/// Toiston poisto.
/// </summary>
public static void Main()
{
    int[] luvut = { 2, 4, -6, 3, 1 };
    int summa = 0;
    for (int i = 0; i < luvut.Length; i++)
    {
        if (luvut[i] >= 0) summa += luvut[i];
        else summa += 10 + i;
    }
    Console.WriteLine(summa);
}
```

A-J: Valitettavasti 3b:n tehtävänanto oli hieman epäonnistunut. Tämä tuli ilmi paitsi antamastanne palautteesta, myös epäselvyyksistä ratkaisuissa. Täydet pisteet sai esimerkiksi seuraavilla ratkaisuilla.

Malliratkaisu 1 tehtävään 3b

```
/// <summary>
/// Aliohjelmakutsut.
/// </summary>
public static void Main()
{
    Random r = new Random();
    int arvottu = r.Next(1, 4);
    TulostaKentanTiedot(1, arvottu, "Edessäsi on ovi.");
    TulostaKentanTiedot(2, arvottu, "Edessäsi on järvi.");
    TulostaKentanTiedot(3, arvottu, "Olet metsässä. On pimeää.");
}
```

```
    /// <summary>
    /// Tulostaa kentän tiedot jos kenttä ja arvottu kenttä
    /// täsmäävät.
    /// </summary>
    /// <param name="kenttaNro">Tulostettava kenttänumero</param>
    /// <param name="arvottuKentta">Arvottu kenttä</param>
    /// <param name="teksti">Kentän esittelyteksti</param>
public static void TulostaKentanTiedot(int kenttaNro, int arvottuKentta, string te
{
  if (arvottuKentta == kenttaNro)
  {
    Console.WriteLine("Tervetuloa peliin, olet kentässä " + kenttaNro + ".");
    Console.WriteLine(teksti);
  }
}
```

Malliratkaisu 2 tehtävään 3b

```
/// <summary>
/// Aliohjelmakutsut.
/// </summary>
public static void Main()
{
  Random r = new Random();
  int arvottu = r.Next(1, 4);
  String[] tekstit = {
   "Edessäsi on ovi.",
   "Edessäsi on järvi.",
   "Olet metsässä. On pimeää."};
  TulostaKentanTiedot(arvottu, tekstit[arvottu - 1]);
}

/// <summary>
/// Tulostaa kentän tiedot jos kenttä ja arvottu kenttä
/// täsmäävät.
/// </summary>
/// <param name="kenttaNro">Tulostettava kenttänumero</param>
/// <param name="teksti">Kentän esittelyteksti</param>
public static void TulostaKentanTiedot(int kenttaNro, string teksti)
{
  Console.WriteLine("Tervetuloa peliin, olet kentässä " + kenttaNro + ".");
  Console.WriteLine(teksti);
}
```

Yleisimmät virheet järjestyksessä

3a

1. ei dokumentaatiota
2. elsessä lisättiin `i+10` sijaista vain `i`
3. elsessä lisättiin `i+10` sijasta `12`

3b

1. ei dokumentaatiota
2. koodissa turhaa toistoa
3. ei suhdetta tehtävänannossa kuvatun parametrien stringin ja kokonaisluvun välillä vaan aliohjelmassa hoidetta tämä logiikka

Pisteytys

Molemmille tehtäville samat arviointikriteerit:

Jos dokumentointi ihan oikein, mutta koodi ihan väärin yht. 0,5 p. Dokumentointi puuttuu/väärin (Mainista saa puuttua jos on muualla) -0.5 p. Syntaksivirhe, -0.5 p. Kuitenkaan epäoleellinen pieni virhe ei haittaa, lopettava aliohjelma sulku puuttuu. Isommat virheet esim sulut puuttuu, `for` väärin, `if` väärin tms. Toimii väärin, mutta suurimmilta osin oikein, -0,5 p. Toimii väärin, -1 p. Ollaan tehty täysin väärää asiaa 0 p.

Virheistä tarkemmin.

3 a) Indeksin plussaus väärin -0,5 ehto väärin -0,5

3 b)

- if ehto väärin -1
- `taulukko[i]` väärin -1
- Parametrit: aliohjelmakutsu tai esittelyrivi väärin -1
- ei kolmatta parametriä tai ei selvennetty miksi ei käytetty -0,5
- ei suhdetta esittelytekstin ja intin välillä -1
- aliohjelmassa turhaa toistoa -0,5 tai -1

Ehdottomasti eniten oli unohdettu kirjoittaa dokumentaatio kokonaan josta -0,5 p.

Tehtävä 3a meni pääsääntöisesti hyvin. Looppi osattiin kirjoittaa oikein, jotkut tarjoili vakiota `Length` :n sijaan. Ehto meni myös hyvin, elsessä sitten tuli ongelmia joillakin.

Tehtävä 3b meni huonommin ehkä myös tehtävänannon takia. Osa oli jättänyt kesken ja sutannut tentin kun eivät keksineet käyttöä kolmannelle parametrille.

Tehtävä 4

- Teki: 104 opiskelijaa
- Keskiarvo: 4.3 pistettä

- Palautteet: mittaa 4.5, vaikea 3.2, arvio 5.5, aika 33 min
- Tarkastaja: Jouni Potila

Yleisimmät virheet

- Parametrina tuodun taulukon *kelvollisuutta* ei tarkisteta mitenkään. Taulukko voi olla `null` tai alle 2 alkiota pitkä, jolloin ohjelma kaatuu. -1 p. `Null` - tarkistuksen puuttumisesta ei kuitenkaan vähennyksiä.
- Epäoleellisista kirjoitusvirheistä (esim `Lenght` vs. `Length`) ei vähennyksiä.
- `taulukko[i + 1]` voi mennä yli taulukon indeksin, ellei `for`-silmukassa ole ehtona (`i < taulukko.Length - 1`) tai vastaavasti `for`-silmukka pitää aloittaa paikasta numero 1 (eikä 0) ja vertailla `if`-lauseessa alkiot `taulukko[i - 1]` ja `taulukko[i]`. -1 p.
- Dokumentaatio puuttuu kokonaan (-1 p.) tai siitä puuttuu puolet (-0.5 p.) (`<param>` ja `<returns>` pitää olla myös). Yleisesti ottaen siis puoli pistettä on saanut `<summary>`:stä ja puoli pistettä siitä, että on ollut myös `<param>` ja `<returns>` oikein.
- Dokumentaatiokommenteissa on syntaksivirheitä tai epäselvyyksiä. Virheen laadusta riippuen -0.25 – 0.5 pistettä.
- `<returns>`-dokumentaatiokommentissa ei kerrota, mitä aliohjelman palauttama `true` tai `false` tarkoittavat, vaan kerrotaan, että "palauttaa `true tai false`". Paluuarvon tyyppi `bool` kuitenkin jo kertoo, että arvo on joko `true` tai `false`, joten tärkeämpää olisi kertoa, että tässä tapauksessa `true` tarkoittaa sitä, että samanlaisia alkioita löytyi 2 tai useampia ja `false` sitä, että ei löytynyt 2:a tai useampaa. -0.25 p.
- Kirjoitettu `if (i == i + 1)` tai `if ([i] == [i + 1])` kun on tarkoitettu `if (taulukko[i] == taulukko[i + 1])` eli taulukon nimi puuttuu indeksinumeron edestä kokonaan. -1 p.
- Parametrina tuotua taulukkoa yritetään luoda uudelleen aliohjelmassa tyyliin `int[] taulukko = new int[5];` tai vastaavaa. -1 p.
- Paluuarvot `true` ja `false` ovat väärinpäin, toista vaihtoehtoa ei palauteta koskaan tai `return false;` `for`-silmukan sisällä, kun pitäisi olla vasta sen jälkeen. –1 p.
- Virheitä aliohjelman esittelyrivillä. -0.25 – 0.5 p. Esim. paluuarvon tyypin oltava joko `bool` tai `Boolean`, ei esimerkiksi `boolean` (pienellä alkukirjaimella), `int` tai `int[]`. -0.25 p. per virhe.
- Luokkaa tai `Main`ia ei tarvinnut kirjoittaa, mutta jos on kirjoitettu, piti olla oikein.
- Luokan tai `Main`in dokumentaatiokommentti ei dokumentoikaan alapuolellaan olevaa asiaa, vaan tehtävänä toteutettavaa aliohjelmaa. -0.25 p.

Tehtävän 4 bonus-kohta

Puolitettuina samat miinuspisteet pätevät kuin varsinaisen tehtävän puolella, mutta samasta virheestä, esim. dokumentaation puutteesta, ei rokoteta bonustehtävässä toista kertaa. Maksimipistemäärä toteutuksesta on 1p ja aikavaativuuden selittämisestä 1p.

Yleisimmät virheet 4-tehtävän bonus-kohdassa

Sisemmässä silmukassa oleva tarkistus `if (taulukko[i] == taulukko[j])` ei toimi siinä tapauksessa, että `i` ja `j` sattuvat olemaan sama luku! Tällöin tulee vahingossa verrattua alkiota itseensä, kun piti verrata kaikkiin muihin. Jos sisempi “j-silmukka” lähtee alkiosta `i+1` liikkeelle, tätä `(i != j)` -tarkastusta ei tarvitse tehdä.

Aikavaativuusanalyysi väärin -1 p. Jos analyysissä kuitenkin on oikeaa ideaa, mutta loppupäätelmä väärin, voi saada 0.25 – 0.5 pistettä.

Malliratkaisu tehtävään 4

```
/// <summary>
/// Tarkistaa onko taulukossa kahdesti tai useammin toistuva alkio.
/// Huom: Järjestää taulukon suuruusjärjestykseen!
/// </summary>
/// <param name="taulukko">Taulukko, josta etsitään. Kutsumisen
/// jälkeen taulukko on järjestetty uudelleen.</param>
/// <returns>Toistuuko taulukossa sama arvo kahdesti tai useammin</returns>
public static bool OnkoToistuviaAlkioita(int[] taulukko)
{
    // Jos koko taulukko on null tai sisältää alle kaksi alkiota,
    // ei toistuvia alkioita voi olla, joten palautetaan
    // suoraan false.
    if (taulukko == null || taulukko.Length < 2) return false;

    // Jos haluaa, voi kloonata taulukon erilliseen samankokoiseen
    // taulukkoon, MUTTA on muistettava, että se syö muistia heti
    // 2-kertaisen määrän. Kopiointi on myös tehtävä alkio kerrallaan.
    // Pelkkä int[] kopio = taulukko; ei ole vielä kopio, vaan
    // vain toinen viite samaan jo olemassa olevaan taulukkoon!
    // Hyvänä puolena kloonauksessa on se, että parametrina
    // tuotua alkuperäistä taulukkoa ei sotketa.

    // Järjestetään taulukko suuruusjärjestykseen:
    Array.Sort(taulukko);

    // Käydään läpi kaikki alkiot alkaen paikasta 1
    // (on jo varmistuttu, että taulukossa on varmasti olemassa
    //  ainakin paikat 0 ja 1, joten ei ole vaaraa, että kaatuisi tähän):
    for (int i = 1; i < taulukko.Length; i++)
    {
        // Tarkistetaan, onko edellisellä ja nykyisellä sama arvo
        if (taulukko[i - 1] == taulukko[i])
        {
            return true; // Kaksi samanlaista peräkkäin --> true
        }
```

```
    }

    // Yhdetkään peräkkäiset alkiot eivät olleet samanlaisia,
    // joten palautetaan false
    return false;
}
```

// Tehtävän 4 BONUS-osion toteutus: (1p)

```
/// <summary>
/// Tarkistaa onko taulukossa kahdesti tai useammin toistuva alkio.
/// </summary>
/// <param name="taulukko">Taulukko, josta etsitään</param>
/// <returns>Toistuuko taulukossa sama arvo kahdesti tai useammin</returns>
public static bool OnkoToistuviaAlkioita2(int[] taulukko)
{
    // Jos koko taulukko on null tai sisältää alle kaksi alkiota,
    // ei toistuvia alkioita voi olla, joten palautetaan
    // suoraan false.
    if (taulukko == null || taulukko.Length < 2) return false;

    // Käydään läpi kaikki alkiot kahdella sisäkkäisellä silmukalla.
    // Jokaista alkiota verrataan kaikkiin muihin alkioihin.
    for (int i = 0; i < taulukko.Length; i++)
    {
        // Aloitetaan (i + 1):stä, jotta vältetään osan jo tarkistettujen
        // tarkastaminen uudelleen.
        // Toinen vaihtoehto olisi aloittaa 0:sta ja tarkastaa silmukan
        // sisällä olevassa if:ssä, että (i != j), jottei verrattaisi
        // yksittäistä alkiota itseensä. Ratkaisusta tulee kuitenkin hitaampi.
        for (int j = i + 1; j < taulukko.Length; j++)
        {
            // Verrataan alkiota toiseen alkioon
            if (taulukko[j] == taulukko[i])
            {
                return true; // Kaksi samanlaista löytyi, palautetaan true
            }
        }
    }
    // Yhdetkään alkiot eivät olleet samanlaisia, joten palautetaan false
    return false;
}
```

Aika-analyysi bonus-tehtävään: (1 p.)

Bonus-tehtävän mukainen ratkaisu on hitaampi kuin taulukon ensin järjestelevä versio. `Array.Sort()` on keskimäärin `O(n log n)` ja järjestelemätön versio parhaimmillaankin aritmeettisen sarjan summa `S(n) = n(1 + n)/2`, missä `n` on taulukon pituus. Esimerkiksi 1000 mittaisen taulukon tapauksessa aikavaativuudet `S(1000) = 500500` (järjestelemätön), kun `1000 * log 1000 ~ 6900` (järjestetty),

ja ero kasvaa aika nopeasti.

Tehtävä 5

- Teki: 37 opiskelijaa
- Keskiarvo: 3.8 pistettä
- Palautteet: mittaa 4.9, vaikea 3.8, arvio 5.4, aika 38 min
- Tarkastaja: Simo Rinne

Malliratkaisu

```
/// <summary>
/// Antaa suurimman perättäisten numeroiden osajonon summan.
/// </summary>
/// <param name="numbers">Tutkittava jono</param>
/// <param name="howManyConsecutive">Montako perättäistä numeroa
/// otetaan huomioon (tätä ei tentissä tarvinut laittaa.)</param>
/// <returns>Suurimman osajonon summa</returns>
/// <example>
/// <pre name="test">
///  Teht1.SuurinViidenOsajono("7316717653") === 27;
///  Teht1.SuurinViidenOsajono("012345") === 15;
///  Teht1.SuurinViidenOsajono("43210123") === 10;
///  Teht1.SuurinViidenOsajono("43210") === 10;
///  Teht1.SuurinViidenOsajono("4321") === 10; // BONUS
///  Teht1.SuurinViidenOsajono("") === 0; // BONUS
/// </pre>
/// </example>
public static int SuurinViidenOsajono(string numbers, int howManyConsecutive = 5)
{
    int greatestSum = 0;
    if (numbers.Length < howManyConsecutive)
        howManyConsecutive = numbers.Length;

    for (int i = 0; i < numbers.Length - howManyConsecutive + 1; i++)
    {
        String thisNumber = numbers.Substring(i, howManyConsecutive);
        int sum = 0;
        for (int j = 0; j < thisNumber.Length; j++)
        {
            sum += int.Parse("" + thisNumber[j]);
        }
        if (sum > greatestSum) greatestSum = sum;
    }
    return greatestSum;
}
```

Virheet ja pisteytys

Muutamat oli käyttäneet turhaan `List<int>`, tai jotkut jopa `List<List<int>>` tietorakenteita, mutta jos aliohjelma teki mitä sen pitikin, niin tästä ei annettu miinuspisteitä.

Luokkaa ja pääohjelmaa ei vaadittu, mutta jos ne oli laitettu, niin ne piti olla syntaksiltaan ja dokumentaatioltaan oikein.

Monet olivat lisänneet summaan suoraan `jono[i]`:n. Kääntäjä hyväksyy sen, mutta lopputulos ei ole se mitä halutaan. Jos merkkijonossa oleva merkki sattuu olemaan vaikka '5' niin summaan lisätään silloin 53, eikä 5. Tuon ongelman saa kumottua esim. vähentämällä aina '0' verran, eli 48.

Aika moni oli unohtanut muuntaa `int.Parse`:n parametrin merkkijonoksi laittamalla alkuun `""+` tai käyttämällä `ToString()`.

Alla on virheet järjestettynä siten, että vakavimmat ja useiten esiintyvät ovat ensin.

- ihan kaikkia merkkijonon osajonoja ei tarkasteltu, -0.5 p.
- on käytetty summa apumuuttujaa, jota ei nollata joka kierroksella, -0.5 p.
- taulukon indekseistä mentiin reilusti yli, -1.0 p.
- summaan lisättiin suoraan ilman muuttamisia taulukon alkio, eli siis merkin ASCII-koodi, -1.0p
- `int.Parse`:lle annettiin `char` eikä `string`, -0.25 p.
- dokumentaatio löytyy, mutta siinä on merkittäviä virheitä, yhden merkin virheitä ei lasketa, -0.5p
- dokumentaatio puuttuu kokonaan, -1.0 p.
- syntaksivirheistä -0.25:sta -0.5:een, riippuen niiden määrästä. Yleisimmät virheet: `for`-silmukasta puuttuu `int`, aliohjelman parametrin tyyppi puuttuu esittelyriviltä ja puuttuvia aaltosulkuja.

Tuo ensimmäinen virhe esiintyi melkein kolmasosalla vastaajista. Ensimmäinen silmukka oli tyypillisesti tehty näin:

```
for (int i=0; i < jono.Length-5; i++) { ... }
```

Tuossa ei tarkisteta nyt kaikkia osajonoja, tuon 5:n tilalla pitäisi olla 4. Muutamilla oli ehtona suoraan `i < jono.Length`, jolloin indekseistä mentiin reilusti yli sisemmässä silmukassa, joka kävi läpi osajonoa.

Jos aliohjelma ei tehnyt mitään järkevää, niin sai 0 pistettä. Hyvin tehdystä dokumentaatiosta saattoi kuitenkin saada (oikeellisuudesta riippuen) 0.5 - 1.0 pistettä.