ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

12. huhtikuuta 2019

Yleistä
Tentti[1] oli pistekeskiarvon (13.3) perusteella vaikeudeltaan keskitasoa. Omasta tehtäväpaperista saa kopion Antti-Jussilta, huone Ag C414.2. Laita sähköpostia tai soita 040 805 3276 etukäteen. Uusintojen ajankohdat löydät kurssin Korppi-sivulta.

Tehtävä	Teki	Keskiarvo	Keskihajonta	Tarkastaja
T1	161	2.86	1.46	Mikko Röyskö
T2	161	4.06	1.33	Mikko Röyskö
T3	161	2.84	1.43	Mikko Röyskö
T4 (yht.)	156	3.68	2.14	Panu Lappalainen
A-kohta	136	3.78	2.05	Panu Lappalainen
B-kohta	20	3.00	2.68	Panu Lappalainen
Yht		**13.3**	5.04	

Arvosteluasteikko

Arvolause	Pistemäärä (alaraja)
5	24
4	21
3	18
2	15
1	12

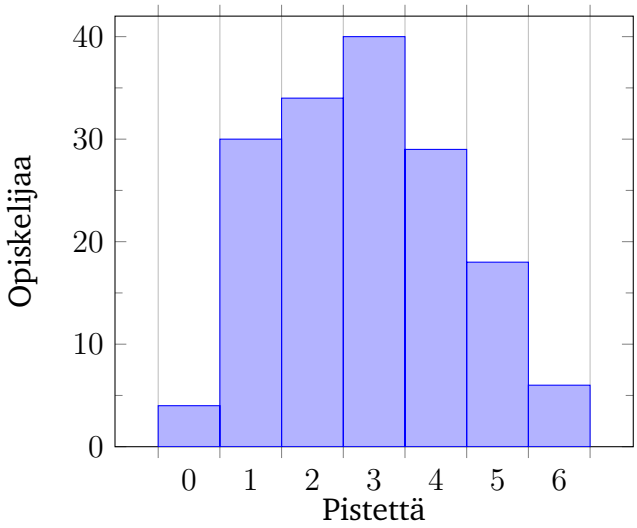
[1] http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2019-04-12-tentti1.pdf

JYVÄSKYLÄN YLIOPISTO

Tehtävä 1 (Tarkastaja: Mikko Röyskö)

Yleiset huomiot

Tehtävä osoittautui hieman hankalaksi. Erityisesti kohta 4 oli monille vaikea.



Malliratkaisu

1A ja D, 2C, 3A, 4D, 5D, 6C.

- Kohta 1: Oikea vastaus oli A. D-vaihtoehdon huonojen sanavalintojen vuoksi iso osa opiskelijoista tulkitsi että tässä tarkoitetaan sisäkkäistä silmukkaa, joten sekin vaihtoehto hyväksyttiin. Vastaukset jakautuivat melko tasaisesti A- ja D-vaihtoehtojen välille (n=149, 93%).

- Kohta 2: Oikea vastaus oli C, jonka vastasi 54 tenttijää (34%). Ohjelmassa voi olla useita samannimisiä aliohjelmia joiden parametrien tyypit eroavat toisistaan, mutta paluuarvon tyyppi on sama (kuormittaminen). Tästä löytyy esimerkiksi luentomonisteesta, kohta 6.5. A-vaihtoehdon valitsi iso joukko vastaajia (n=57), mutta se on kuitenkin väärin.

- Kohta 3: Oikea vastaus oli A, jonka valitsi 71 opiskelijaa (44%). 74 valitsi vaihtoehdon C, mutta esimerkiksi `public static string PalautaKissa() {return "kissa";}`.

- Kohta 4: Oikea vastaus oli D, jonka valitsi vain 24 opiskelijaa (15%). Suurin osa (n=122) vastasi vaihtoehdon A, eli ”Koira istuu Koira”. Tässä olisi pitänyt huomata että koska kyseessä on `StringBuilder`-olio (rivi 3), on kyseessä olioviitteen (*ei* merkkijonon sisällön) kopiointi. Näin ollen seuraavalla rivillä oleva `sb1.Append("istuu");` muuttaa sen olion sisältöä, johon sekä `sb1` että `sb2` viittaavat.

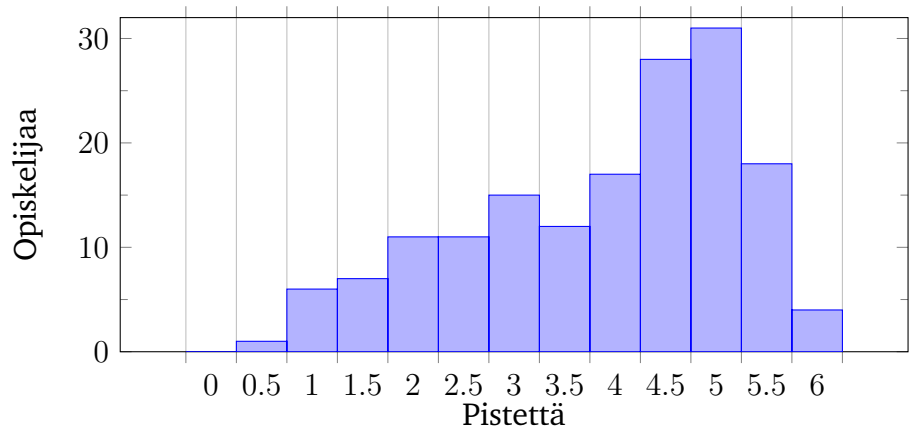
- Kohta 5: Oikea vastaus oli D, jonka vastasi 83 opiskelijaa (52%). C-kohtaa veikkasi 47, mutta se on selkeästi väärin. Esimerkkinä demo 10 tehtävä 1-2.

- Kohta 6: Oikea vastaus oli C, jonka vastasi 83 opiskelijaa (52%). Seuraavaksi suosituin vastaus oli D 35 vastauksella.

Pisteytys ja virheet

Oikeasta vastauksesta tuli 1 piste, väärästä tai tyhjästä 0 pistettä.

Tehtävä 2 (Tarkastaja: Mikko Röyskö)



- Kohta 1: Tämä osattiin hyvin, lähestulkoon kaikki vastanneet saivat > 0.5 p. `for`-, `while`-, `do-while`- ja `foreach`-silmukat olivat hyvin hallussa useimmilla, ja niitä mainittiin suht tasaisesti. Rekursio on myös toistorakenne.

  Pisteytys:

  - Jokainen toistorakenne, jota on käytetty oikein, 0.33 p.
  - Jos toistorakenne oli osattu vain nimetä, tai sitä selkeästi ei osattu käyttää, 0.1 p.

  Yleisimmät virheet:

  - Yksittäisistä lievistä virheistä ei rokotettu. Toistuvat tai vakavat syntaksi- tai muotoiluvirheet, -0.1 p.
  - `while` tai `do-while`-silmukasta puuttui juoksevan muuttujan esittely tai sen kasvatus, -0.1 p.
  - `foreach`-silmukan juoksevan muuttujan tyypin puuttuminen, -0.1 p.
  - Dokumentaatiota, aliohjelmia ja luokkia ei vaadittu. Niistä pystyi kuitenkin saamaan miinuspisteitä mikäli ne sisälsivät selkeitä syntaksivirheitä.

- Kohta 2: Osattiin joko hyvin tai huonosti. Kymmenjärjestelmän vastaukseksi hyväksyttiin joko 264 tai 8, sillä tehtävässä ei käsketty nimenomaan tätä summaksi saatua 8-bittistä binäärilukua muuttamaan kymmenjärjestelmään. Tehtävässä olisi pitänyt huomata että saatu vastaus on suurempi kuin 8 bittiä, jolloin ensimmäinen bitti tipahtaa pois. Jos binäärivastaus oli muuten oikein niin tästä rokotettiin hieman. Oikeasta kymmenjärjestelmän luvusta sai 0.5 p. ja oikeasta binääristä toiset 0.5 p. Mikäli binääriluvussa oli ylimääräinen bitti, eli vastauksena oli `1 0000 1000`, sai 0.25 p.

$$\begin{array}{r} {\scriptstyle 1\ 1111\ 111} \\ 0\,1011\,1011 \\ +\,0\,0100\,1101 \\ \hline 1\,0000\,1000 \end{array}$$

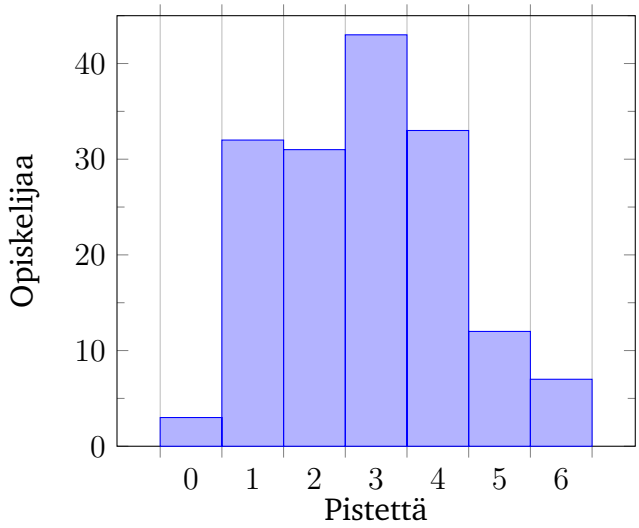
Vastauksessa tulee yhdeksäs bitti (`1`), joka ei mahdu vaadittuun kahdeksaan bittiin, jolloin jäljelle jää `0000 1000`. Vastaukseksi hyväksyttiin joko muuttamalla tämä 8-bittinen luku kymmenjärjestelmään, eli 8, mutta myös binääriluvut yhteenlaskettuna; `10111011` = 187 ja `01001101` = 77 $\rightarrow$ 264. Muutama opiskelija oli käyttänyt 2-komplementtia laskuissa, jolloin `10111011` = -69. Nyt $77-69 = 8$, joten tämäkin ratkaisu tuotti oikean tuloksen.

- Kohta 3: Yhden vian löytämisestä sai 0.25 p, ja sen oikeasta korjauksesta 0.25 p. Virheellisestä korjauksesta 0 p. Mikäli oli lueteltu enemmän kuin neljä vikaa ja korjauksissa oli virheitä, annettiin pistevähennyksiä. Kysymys sisälsi 9 erilaista virhettä, joista kaikki onnistuttiin löytämään. Alla on lueteltu viat esiintymisjärjestyksessä sekä niiden korjaukset.
  - Paluuarvon tyyppi tulisi olla `double`.
  - Aliohjelman nimi tulisi kirjoittaa isolla alkukirjaimella, `Keskiarvo`.
  - `List` tarvitsee kaveriksi alkioiden tyypin, esimerkiksi `List<int>` tai `List<double>`. Myös taulukoksi muuttaminen hyväksyttiin.
  - Funktio ei tarkista onko lista tyhjä tai `null`. Tämä tulisi tarkistaa `if`-lauseella ennen silmukkaa.
  - summa on `int`-tyyppinen, jonka seurauksena lopussa olevasta jakolaskusta tulee myös kokonaisluku. Korjaukseksi hyväksyttiin joko summan muuttaminen `double`-tyyppiseksi, tai tekemällä ns. eksplisiittinen tyyppimuunnos `return`-lauseen yhteydessä, esimerkiksi `return (double)summa / luvut.Count;`.
  - Silmukka menee yhden kierroksen liian pitkälle. Ehdossa tulisi olla < eikä <=.
  - Koska kyseessä on lista, alkioiden määrä saadaan `luvut.Count`-ominaisuudesta. Taulukoilla sen sijaan toimisi `luvut.Length`.
  - Summauksessa =+ on väärinpäin. Tulisi olla +=.
  - Listaa (tai taulukkoa) indeksoidaan väärin. Kaarisulkujen sijasta tulisi käyttää hakasulkuja, `luvut[i]`.
- Kohta 4: C eli 24. Osattiin hyvin, 104 oikeaa vastausta.

Tehtävä 3 (Tarkastaja: Mikko Röyskö)

Yleiset huomiot

Tehtävä osoittautui aavistuksen haastavaksi. Erityisesti kohta 6 oli monille vaikea.



Malliratkaisu

1B, 2B, 3B, 4B, 5C, 6A.

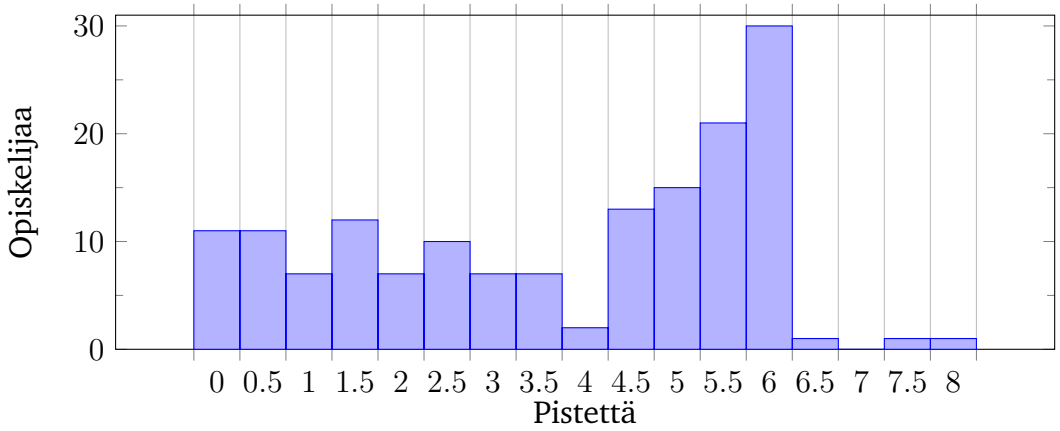
- Kohta 1: Oikea vastaus oli B, jonka valitsi 76 vastaajaa (47%). A-vaihtoehto oli seuraavaksi suosituin 44 vastauksella. Tässä pitää kuitenkin huomata että vaihtoehdossa nimenomaan lukee tyhjä, eikä `null`.
- Kohta 2: Oikea vastaus oli B. Osallistujista 105 vastasi (65%) oikein.
- Kohta 3: Oikea vastaus oli B, jonka valitsi 62 vastaajaa (38%). D-vaihtoehto sai 56 vastausta. Monissa ohjelmointikielessä, kuten myös C#:ssa kirjaimia voi käsitellä numeroina, joten `char` ja `int` vertailu on mahdollista. Kokeile mitä seuraava koodi tulostaa: `for(char c = 'A'; c < 256; c++) { Console.Write(c); }`.
- Kohta 4: Oikea vastaus oli B, jonka vastasi 58 vastaajaa (36%). Suosittu vastaus oli D, jonka valitsi 70 vastaajaa. Tämän sai ratkaistua noudattamalla aivan tavallisia matemaattisia laskujärjestykseen liittyviä sääntöjä, sulut ensin, sitten vasemmalta oikealle.
  - Lähtotilanne: `(true||false)&&!true||2<3`.
  - Ensin sulkujen sisältö: `true||false` on `true`, eli `true&&!true||2<3`.
  - Huutomerkki kääntää totuusarvon ympäri, eli `true&&false||2<3`.
  - `true&&false` on tietysti `false`, joten jäljellä on `false||2<3`.
  - Kaksi on pienempi kuin kolme, `false||true`.
  - Jäljellä on sama kuin mitä alussa oli sulkujen sisällä, eli tulos on `true`.
- Kohta 5: Oikea vastaus oli C, jonka vastasi 117 vastaajaa (73%). Tämä kysymyshän oli lähestulkoon sama kuin T1.6.
- Kohta 6: Oikea vastaus oli A, jonka vastasi 39 vastaajaa (24%). Yleisin vastaus oli B jonka vastasi 102 vastaajaa. Tämä oli hieman hankala, sillä monet ajattelivat ensimmäisenä että luvut lasketaan yhteen, ja tämä sitten lisätään merkkijonoon.

Eli `"1"+5+6` → `"1"+11` → `"111"`. Tämä on kuitenkin väärin, sillä plusmerkit lasketaan vasemmalta oikealle siinä järjestyksessä missä ne esiintyvät, ellei mukana ole sulkuja. Täten merkkijonon muodostuminen menee askeleittain `"1"+5+6` → `"15"+ 6` → `"156"`. Mikäli tehtävänannossa olisi ollut sulut, `"1"+ (5 + 6)`, olisi B-vaihtoehto ollut oikein.

Pisteytys ja virheet

Oikeasta vastauksesta tuli 1 piste, väärästä tai tyhjästä tai usean vaihtoehdon valitsemisesta 0 pistettä.

Tehtävä 4 (Tarkastaja: Panu Lappalainen)

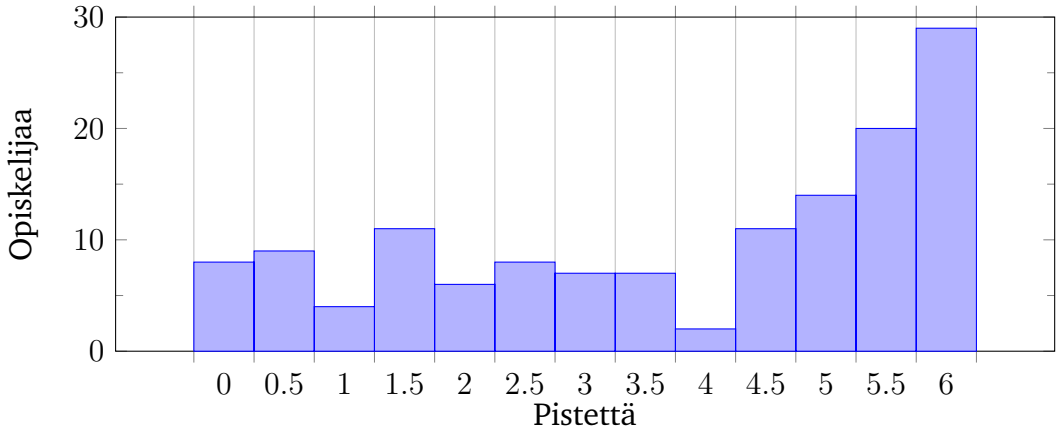


Tehtävässä sai valita yhden kahdesta kohdasta joista ensimmäisestä sai 6 ja jälkimmäisestä 8 pistettä. Suurin osa (87 %) vastasi A-kohtaan. A-kohta meni keskimäärin hieman paremmin. Kumpaankin kohtaan tuli paljon erilaisia, toisistaan hyvinkin paljon poikkeavia ratkaisuja. Mallivastaukset eivät ole ainoita oikeita ratkaisuja.

Yleisiä huomioita

- Valitettavan moni (38 %) oli jättänyt dokumentaation kokonaan tai osittain kirjoittamatta, yleisin virhe oli `<param>`- tai `<returns>`-kohdan puuttuminen.
- Vertailuoperaattorina oli käytetty yhtä =-merkkiä kahden sijaan.
- Virheitä isojen ja pienten kirjainten kanssa, esim `List<int>` oli monelle vaikea.
- Huolimattomuus: satunnaisesti puuttuvia puolipisteitä; aliohjelman päättävä loppusulku puuttui lähes puolesta vastauksista (ei rangaistu).

A-kohta



A-kohdassa tuli kirjoittaa `Collatz`-niminen funktio, joka laskee annetun kokonaisluvun aloittavan lukujoukon Collatzin konjektuurin mukaisesti. Tehtävässä oli tarkoitus käyttää silmukkaa tai rekursiota ja moduloehtoa (`%`).

Malliratkaisu (silmukka)

```
/// <summary>
/// Toteuttaa Collatzin konjektuurin mukaisen lukujoukon.
/// </summary>
/// <param name="luku">lukujonon ensimmäinen luku</param>
/// <returns>Collatzin konjektuurin mukainen lukujoukko</returns>
public static List<int> Collatz(int luku)
{
    List<int> luvut = new List<int> { luku };
    while (luku != 1)
    {
        if (luku % 2 == 0) luku = luku / 2;
        else luku = luku * 3 + 1;
        luvut.Add(luku);
    }
    return luvut;
}
```

Malliratkaisu (rekursio)

```
public static List<int> Collatz(List<int> lista)
{
    int luku = lista[lista.Count - 1];
    if (luku == 1) return lista;
    if (luku % 2 == 0) luku = luku / 2;
    else luku = 3 * luku + 1;
    lista.Add(luku);
    return Collatz(lista);
}
```

Yleisimpiä vaikeuksia

- kertolaskussa ei oltu käytetty kertomerkkiä, tai oli käytetty $*$-merkin sijaan pistettä tai $\times$-merkkiä, ja jopa “murtolukuja” näkyi jakolaskussa
- `if-else`-rakenteessa `else`n perään oltiin kirjoitettu toinen "ehto"
- silmukkarakennetta käytettiin väärin tai ei ollenkaan
- "`not all code paths return a value`": todella moni oli käsitellyt erikoistapaukset hyvin, mutta itse listan palauttaminen unohtui
- yleisestikin listan luominen tuotti vaikeuksia, erityisesti sulkujen osalta

Pisteytys kohdittain

Keskiarvo/**maksimipisteet**

`d`: 0.3/**0.5** p. Kurssin käytännön mukainen XML-dokumentaatio

`f`: 0.4/**0.5** p. Funktion esittelyrivi (näkyvyys, paluuarvo, nimi, parametri)

`s`: 0.6/**1.0** p. Toimiva silmukka tai rekursio

`e`: 0.8/**1.0** p. Järkevä ehtolause

`L`: 0.6/**1.0** p. Listan käsittely (alustus ja lisäys)

`m`: 0.4/**0.5** p. Laskut (vaati sijoituksen muuttujaan, listaan lisäyksen tai funktiokutsun parametrina olemisen, pelkkä ”lasku” ei riittänyt)

`p`: 0.3/**0.5** p. Palautus (oikea paluuarvon tyyppi kaikissa oikeissa kohdissa)

`t`: 0.5/**1.0** p. Toimii

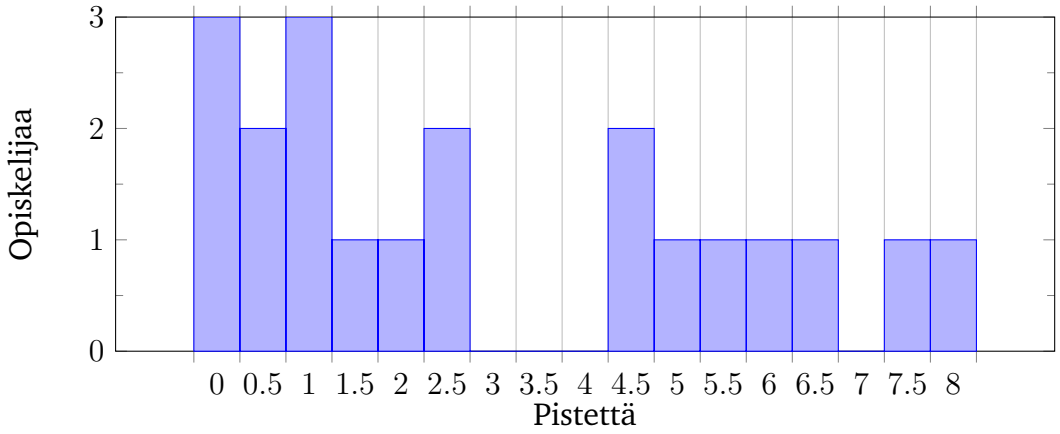
Samasta virheestä ei rangaistu "kahdesti", eli yksi virhe ym. kategorioissa ei vaikuttanut muihin kategorioihin. Esimerkiksi Toimii-kategoriasta saattoi saada täydet pisteet, vaikka palautus puuttuikin jostain kohtaa. Pistevähennyksiä sai kategorioiden ulkopuolella tehdyistä virheistä (esim. muuttujien nimeäminen) korkeintaan 1 pistettä kuitenkin siten, että pistevähennykset korkeintaan puolittivat tehtävästä kerätyt pisteet, puolen pisteen tarkkuudella ylöspäin pyöristäen.

Onnistuneita silmukkarakenteita oltiin tehty `while`-, `do-while`- ja jopa `for`-silmukoilla. Harvoista rekursiota yrittäneistä kukaan ei saanut toimivaa rekursiota toteutettua.

Apufunktioita käytettäessä myös itse kirjoitetut funktiot otettiin arvostelussa huomioon. Moni käytti myös apumuuttujia.”Turhista” tai käyttämättömistä muuttujista ei rangaistu.

Tehtävässä ei pyydetty kirjoittamaan luokkaa, pääohjelmaa tai testejä, eikä niiden tai minkään ylimääräisen sisällyttämisestä myönnetty tai vähennetty pisteitä.

B-kohta



B-kohtaan vastasi vain 13 % tenttijöistä. Kohdassa tuli kirjoittaa `NaytaArvatut`-niminen funktio, joka palauttaa alkuperäisen lauseen käsiteltynä siten, että parametrina annetut arvatut kirjaimet näytettiin, ja muut peitettiin `*`-merkillä. Tehtävästä sai lisäksi bonuspisteitä erikoismerkkien näyttämisestä.

Malliratkaisu

```
/// <summary>
/// Palauttaa alkuperäisen lauseen näyttäen siitä vain arvatut kirjaimet.
/// </summary>
/// <param name="lause">alkuperäinen lause</param>
/// <param name="arvatut">arvatut kirjaimet</param>
/// <returns>arvaamattomien kirjainten osalta sensuroitu merkkijono</returns>
public static string NaytaArvatut(string lause, string arvatut)
{
    StringBuilder sb = new StringBuilder();
    string erikoismerkit = "!?., ";
    for (int i = 0; i < lause.Length; i++)
    {
        char merkkiPienena = Char.ToLower(lause[i]);
        if (arvatut.IndexOf(merkkiPienena) >= 0) sb.Append(lause[i]);
        else if (erikoismerkit.IndexOf(merkkiPienena) >= 0) sb.Append(lause[i]);
        else sb.Append("*");
    }
    return sb.ToString();
}
```

Myös `String.Contains()`- tai `Char.IsLetter()`-metodien käyttö hyväksyttiin. Sisäkkäisillä silmukoilla saatiin niin ikään täysiä pisteitä. Parametrien tyypit sai itse valita, esimerkiksi `arvatut`-parametrin tyypiksi hyväksyttiin `string`, `char[]` ja `List<char>`.

Pisteytys

Keskiarvo/**maksimipisteet**

`d`: 0.3/**0.5** p. Kurssin käytännön mukainen XML-dokumentaatio

`f`: 0.3/**0.5** p. Funktion esittelyrivi (näkyvyys, paluuarvo, nimi, parametri)

`s`: 0.9/**1.5** p. Silmukka

`e`: 0.5/**1.0** p. Ehtolause

`sb`: 0.4/**1.0** p. `StringBuilder`in käsittely (alustus ja lisäys)

`p`: 0.3/**0.5** p. Palautus (oikea paluuarvotyyppi kaikissa oikeissa kohdissa)

`t`: 0.3/**1.0** p. Toimii

`B`: 0.2/**2.0** p. Erikoismerkkien käsittely automaattisesti (ei erillisillä ehtolauseilla)

`StringBuilderin` käyttö oli todella pahasti hukassa vain 35 % onnistumisprosentilla.

Kuten a-kohdassa, samasta virheestä ei rangaistu "kahdesti". Pistevähennyksiä sai kategorioiden ulkopuolella tehdyistä virheistä (esim. muuttujien nimeäminen) korkeintaan 1 pistettä kuitenkin siten, että pistevähennykset korkeintaan puolittivat tehtävästä kerätyt pisteet, puolen pisteen tarkkuudella ylöspäin pyöristäen (eli opiskelijan eduksi).

Apumuuttujat ja apufunktiot arvioitiin samalla tavalla kuin a-kohdassa.

Tehtävässä ei pyydetty kirjoittamaan luokkaa, pääohjelmaa tai testejä, eikä niiden sisällyttämisestä myönnetty tai vähennetty pisteitä.

Suurin osa B-kohtaan vastanneista ei edes yrittänyt käsitellä erikoismerkkejä joten bonuspisteitä ei paljoakaan jaettu.