Tentti / 3.12

Vastaa yhteensä **neljään** tehtävään (huomaa että tehtävissä voi olla useita alakohtia). Vastaa **JOKAISEN tehtävänumeron vastaus erilliselle konseptipaperille** (joka tehtävälle eri tarkastaja). Jos vastaat useampaan kuin neljään, tarkastajilla on oikeus valita 4 **huonointa** vastausta. Tenttiaika 4 tuntia.

1. Java-koodi

1a). Kirjoita Javalla molemmat tarvittavat aliohjelmat, jotta pääohjelma kääntyisi ja toimisi aliohjelmien nimien edellyttämällä tavalla. Kirjoita myös Javadocin mukainen kommentointi. Mitä ohjelma tulostaisi? (4 p)

```
public static void main (String[] args){

    double[] lukuTaulu = {0.5, 1.2, 2.0, 3.5 };
    double luku1 = 1.5;
    double luku2 = 3.0;
    double luku3 = 0.5;

    double vastaus1 = summaa(luku1, luku1, luku2);
    double vastaus2 = summaa(lukuTaulu);

    System.out.println(vastaus1 + " " + vastaus2);
}
```

1b) Miksi edellisessä pääohjelmassa voi olla kaksi samannimistä aliohjelmakutsua? (1 p)

1c) Kääntyykö seuraava koodinpätkä Javassa? Perustele. (1 p)

```
String moro = "Hello World!";
int[] taulukko = {2, 10, 223, 22};

taulukko[1] = moro;
```

2. Aliohjelmat (ymmärtäminen)

Tämä ohjelma koostuu pääohjelmasta ja neljästä aliohjelmasta. Aliohjelman täydennettävät kohdat on merkitty alaviivalla ja aliohjelmien kommentit ovat puutteelliset. Kirjoita vastauspaperiin mitä **ohjelma tulostaa**, **aliohjelmien kommentit** ja **esittelyrivit** täydennettynä. Ehdota myös aliohjelmille **kuvaavampia nimiä**. (yht 6 p)

```
public static void main(String[] args) {
    String [] taulukko1 = { "1","2","3","4","5" };
    int [] taulukko2 = { 1,2,3,4,5 };
    ArrayList <String >lista = new ArrayList<String> ();
    lista.add("Jani"); lista.add("Pekka"); lista.add("Joose");

    tulosta(aliohjelma1(taulukko1));
    tulosta(aliohjelma2(taulukko2));
    tulosta(aliohjelma3(lista));
}
```

a)

```
/**
 * @param
 */
public static ______ tulosta( ______ tulostettava) {
   System.out.println(tulostettava);
}
```

b)

```
/**
 * @param
 * @return
 */
public static ______ aliohjelma1( ______ taulukko1) {
    String summa = "";
    for (int i = 4;i>=0;i--) {
        summa += taulukko1[i];
    }
    return summa;
}
```

c)

```
/**
 * @param
 * @return
 */
public static ______ aliohjelma2( ______ taulukko2) {
    int summa = 0;
    for (int i:taulukko2) {
        summa += i;
    }
    return "" + summa;
}
```

d)

```
/**
 * @param
 * @return
 */
public static ______ aliohjelma3(______ lista) {
    int luku = 3/2;
    return lista.get(luku);
}
```

3. Aliohjelmat (tuottaminen)

a) Tee aliohjelma, jolle tuodaan parametrina kahden (2) alkion kokoinen taulukko. Aliohjelma vaihtaa taulukon alkioiden arvoja keskenään. Arrays -luokan apuja ei saa käyttää (2p). Esimerkki:

```
int taulukko[] = {2,3}
taulukko = vaihdaPaikat(taulukko);
```

taulukon arvot nyt: {3,2}

b) Tee aliohjelma, jolle tuodaan parametrina merkkijono ja kirjain. Aliohjelma palauttaa merkkijonon, josta on korvattu kaikki annetun kirjaimen esiintymät välilyönneillä (space). Poikkeus: Jos merkkijono on "kissa", palautetaan "koira" (2p). Esimerkki ComTestin syntaksilla:

```
korvaaKirjaimet("kilpikonna", 'k') === " ilpi onna"
korvaaKirjaimet("kissa", 'k') === "koira"
korvaaKirjaimet("koira", 'k') === " oira"
```

c) Tee aliohjelma, jolle tuodaan parametrina taulukko. Aliohjelma palauttaa taulukon, jossa annetun taulukon alkiot ovat päinvastaisessa järjestyksessä. (2 p) Esimerkki:

```
int taulukko[] = {1,2,3,4,5,6};
taulukko2 = takaperin(taulukko);
```

taulukko2:n arvot nyt: {6,5,4,3,2,1} ja taulukko on muuttumaton. Tehtävässä ei saa käyttää `Arrays`-luokan aliohjelmia.

4. Yleistä Java-tietoa

Vastaa alla oleviin kysymyksiin lyhyesti (max. konsepti yhteensä koko tehtävään).

4a) Mitä tarkoittaa kommentointi ja miksi koodia kannattaa kommentoida? Anna esimerkki kaikista Javan kolmesta kommentointityypistä. Mitä hyötyä on erityisesti javadoc-tyylisestä kommentoinnista? (2p)

4b) Kerro hyvistä koodauskäytännöistä. Miksi niitä kannattaa noudattaa? (1p)

4c) Kerro vaiheittain kuinka Java-lähdekoodista syntyy toimiva ohjelma. Kuinka nämä vaiheet suoritetaan komentoriviä käyttäen (ei siis Eclipsessä painamalla ctrl+F11)? Kerro vielä mitä hyötyä Java-virtuaalikoneesta on. (3p)

5. Muuttujat ja tietorakenteet

Älä vastaa kohtuuttoman pitkästi, vaan esitä kysytyt asiat selkeästi ja kuvaavasti (piirrokset eivät ole pakollisia, mutta ovat hyödyllisiä).

a) Selitä taulukon toiminta (esimerkiksi `int`-taulukko). (2p)

b) Pohdi kuinka dynaamiset tietorakenteet(`ArrayList`) eroavat taulukosta. (1p)

c) Selitä "tavallisen" alkeistietotyypin (arvo-muuttujan) ja oliotietotyypin (viitemuuttujan) ero. (2p)

d) Mitä tarkoitetaan vakiolla (constant), ja miten ne esitellään? (1p)

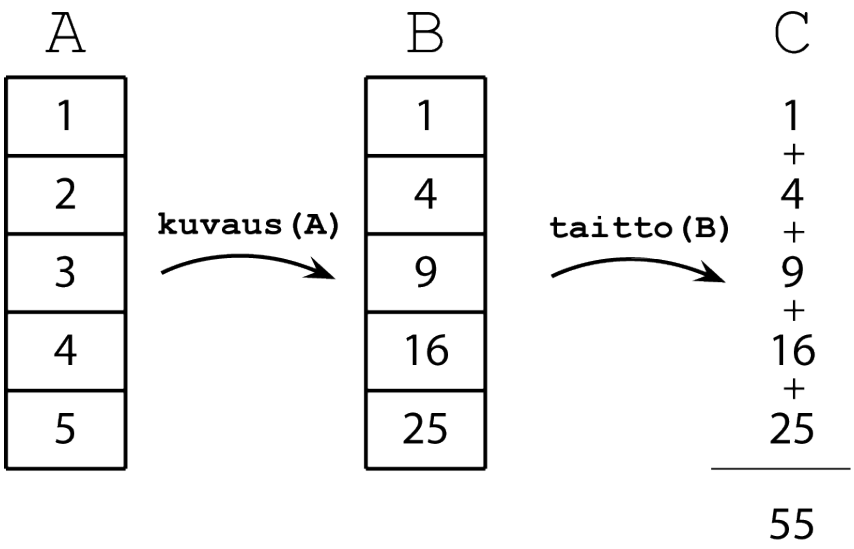
6. Taulukkojen käsittelyä

Tarkoituksena on muodostaa taulukosta taulukko, jonka jokainen alkio on sen ja sitä edeltävän alkioiden neliöiden summa. Sinulle on annettu kaksi aliohjelmaa lähdekoodeinen, `kuvaus`- ja `taitto`, joita saat käyttää tehtävässäsi.

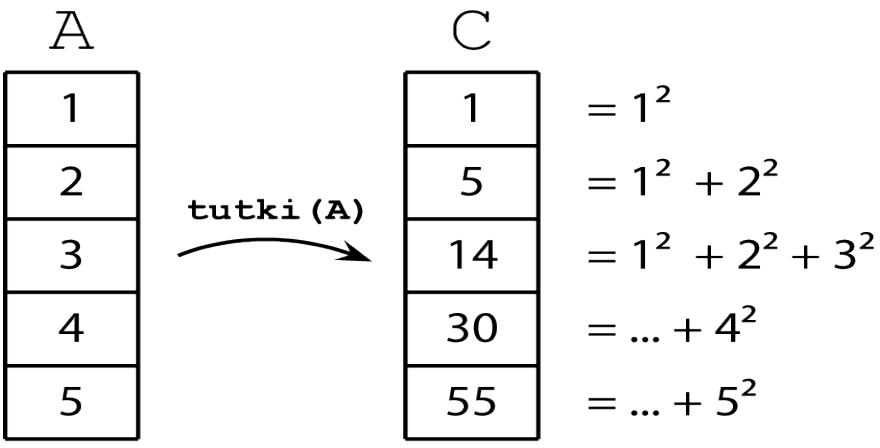
`kuvaus`-aliohjelma *kuvaa* jokaisen lähtötaulukon arvon joksikin uudeksi arvoksi, tässä tapauksessa neliökseen eli arvon kerrottuna itsellään, ts. luku 5 kuvautuu luvuksi $5^2 = 5 \times 5 = 25$. `taitto`-aliohjelma *taittaa* tulokset yhteen summaamalla alkiot keskenään, jolloin esimerkiksi alkiot 1, 3, 6 ja 8 taittuvat tulokseksi 1 + 3 + 6 + 8 = 18. Kummankin toiminnallisuus esitetty kuvassa 1.

Tehtävänäsi on nämä yhdistämällä luoda aliohjelma `tutki`, joka palauttaa uuden kokonaislukutaulukon. Aliohjelma summaa jokaisen taulukon alkion ja sitä edeltävän alkioiden neliöt keskenään, ja tallentaa tuloksen uuteen taulukkoon. Esimerkiksi alkiot 2, 3 ja 4 muodostavat taulukon { 2^2, $3^2 + 2^2$, $4^2 + 3^2 + 2^2$ } = { 4, 13, 29 }. Näin aliohjelma ”yhdistää” taitto- ja kuvausfunktiot.

Seuraavan esimerkin kuvassa `taitto`-funktiota on kutsuttu arvolla `n=5`, siksi tulos on `55`. Arvolla `n=3` kutsuttuna `taitto`-aliohjelma palauttaisi 14 esimerkin kuvassa.



Kuva 1. kuvaus-aliohjelma neliöi taulukon A alkiot uuteen taulukkoon B. taitto-aliohjelma summaa taulukon B alkiot yhteen ja palauttaa tuloksen kokonaislukuna.



Kuva 2. tutki-aliohjelma, joka yhdistää edellisten alkioiden summat. Kohdetaulukon C jokainen alkio on alkion ja sitä edeltävien alkioiden neliöiden summa.

Liite: Kuvaus- ja taitto-aliohjelmien lähdekoodit

```
/**
 * Neliöi jokaisen taulukon alkion ja palauttaa tulokset
 * uudessa taulukossa.
 *
 * @param taul  Lähtötaulukko
 * @return      Taulukko uusilla arvoillaan
 */
public static int[] kuvaus(int[] taul) {
    int[] uusi = new int[taul.length];
    for (int i = 0; i < taul.length; i++) {
        uusi[i] = taul[i] * taul[i];
    }
    return uusi;
}

/**
 * Summaa taulukon taul n ensimmäistä alkiota.
 *
 * @param taul  Taulukko
 * @param n     Summauksen lopetusindeksi
 * @return      Alkioiden summat
 */
public static int taitto(int[] taul, int n) {
    int summa = 0;
    for (int i = 0; i < n; i++) {
        summa += taul[i];
    }
    return summa;
}
```

Liite: String-luokan API

length

```
public int length()
```

Returns the length of this string. The length is equal to the number of Unicode code units in the string.

Specified by:

`length` in interface `CharSequence`

Returns:

the length of the sequence of characters represented by this object.

indexOf

```
public int indexOf(String str,
                   int fromIndex)
```

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. The integer returned is the smallest value `k` for which:

```
    k >= Math.min(fromIndex, this.length()) &&
this.startsWith(str, k)
```

If no such value of *k* exists, then -1 is returned.

Parameters:

`str` - the substring for which to search.

`fromIndex` - the index from which to start the search.

Returns:

the index within this string of the first occurrence of the specified substring, starting at the specified index.

---

substring

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

Parameters:

`beginIndex` - the beginning index, inclusive.

`endIndex` - the ending index, exclusive.

Returns:

the specified substring.

Throws:

`IndexOutOfBoundsException` - if the `beginIndex` is negative, or `endIndex` is larger than the length of this `String` object, or `beginIndex` is larger than `endIndex`.

---

replace

```
public String replace(char oldChar,
                      char newChar)
```

Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.

If the character `oldChar` does not occur in the character sequence represented by this `String` object, then a reference to this `String` object is returned. Otherwise, a new `String` object is created that represents a character sequence identical to the character sequence represented by this `String` object, except that every occurrence of `oldChar` is replaced by an occurrence of `newChar`.

Examples:

```
"mesquite in your cellar".replace('e', 'o')
        returns "mosquito in your collar"
"the war of baronets".replace('r', 'y')
        returns "the way of bayonets"
"sparring with a purple porpoise".replace('p', 't')
        returns "starring with a turtle tortoise"
"JonL".replace('q', 'x') returns "JonL" (no change)
```

Parameters:

`oldChar` - the old character.
`newChar` - the new character.

Returns:

a string derived from this string by replacing every occurrence of `oldChar` with `newChar`.
