CS-A1111 Ohjelmoinnin peruskurssi Y1 Ylimääräinen tentti 13.2.2018

Kirjoita jokaisen vastauspaperisi alkuun kurssin nimi, kokeen päivämäärä, nimesi, opiskelijanumerosi, vastauspaperiesi kokonaismäärä sekä allekirjoituksesi.

Tärkeitä ohjeita vastausten kirjoittamiseen Kun kirjoitat ohjelmakoodia, käytä kahden ruudun levyisiä sisennyksiä. Jos sisennyksiä ei ole käytetty tai niistä ei saa selvää, vähennetään siitä pisteitä. Kirjoitettavaan ohjelmakoodiin ei tarvitse lisätä kommentteja. Missään tehtävässä tulostusta ei tarvitse muotoilla. Voit myös olettaa, että käyttäjän antama syöte on virheetöntä, ellei tehtävässä erikseen käsketä käsittelemään virhetilanteita.
Tentissä ei saa käyttää laskimia eikä lisämateriaalia.

1. Kohdissa a, b, c ja d kerro, mitä annettu ohjelma tulostaa. Vastausta ei tarvitse perustella. Kohdissa e, f ja g kerro, mitä tehtävässä esitetty funktio tekee. Älä selitä funktion toimintaa käsky käskyltä, vaan selitä parilla lauseella, mikä on funktion tarkoitus (esimerkiksi: "funktio laskee ja palauttaa parametrina annetussa listassa olevien lukujen summan"). Funktioille annettavien parametrien luonne on selitetty kunkin kohdan yhteydessä. Huomaa, että annetuissa ohjelmissa tai funktioissa voi olla myös virheitä. Kerro siinä tapauksessa, mitä annettu virheellinen ohjelma tulostaa tai miten virheellinen funktio toimii - ei siis sitä, miten ohjelman tai funktion pitäisi toimia, jos siinä ei olisi virheitä.

a) (2 p)

```
def main():
    hinta = 150
    if hinta > 100:
        print("Kallis.")
    if hinta < 50:
        print("Halpa.")
    else:
        print("OK.")

main()
```

b) (2 p)

```
def main():
    a = 1
    b = 2
    c = 3
    a = b
    b = c
    c = a
    print(a, b, c)

main()
```

c) (3 p)

```
def main():
    lista = [10, 20, 5, 15, 10]
    tulos = 0
    i = 0
    while i < len(lista):
        if lista[i] < 15:
            tulos = tulos + lista[i]
        i = i + 1
        print(tulos)

main()
```

d) (5 p)

```
def vaihda(luku, luvut):
    luvut[0] = luvut[0] + luku
    luku = 77
    return luku

def main():
    lista = [20, 50, 60]
    tulos = 10
    vaihda(tulos, lista)
    print(tulos)
    for alkio in lista:
        print(alkio)

main()
```

e) Funktiolle annetaan parametreina kaksi kokonaislukuja sisältävää listaa, joilla on sama pituus. (4 p)

```
def mysteeri1(lista1, lista2):
    tulos = 0
    i = 0
    while i < len(lista1):
        if lista1[i] == lista2[i]:
            tulos = tulos + lista1[i]
        i += 1
    return tulos
```

f) Funktiolle annetaan parametreina kaksi merkkijonoa, joilla on sama pituus. (4 p)

```
def mysteeri2(jono1, jono2):
    tulos = ""
    for i in range(len(jono1)):
        tulos = tulos + jono1[i] + jono2[i]
    return tulos
```

g) Funktiolle annetaan parametrina kokonaislukuja sisältävä lista, jossa on vähintään kaksi alkiota (5 p)

```
def mysteeri3(lista):
    i = 1
    while i < len(lista):
        if lista[i] < lista[i-1]:
            return False
        else:
            return True
        i = i + 1
```

2. a) Eräällä kurssilla opiskelija saa arvosanan tentistä ja harjoitustyöstä. Jos vähintään toinen arvosanoista on 0, opiskelijan kurssisuoritus on hylätty ja hän saa kurssiarvosanan 0. Muussa tapauksessa opiskelijan kurssiarvosanaksi tulee yleensä tenttiarvosana. Jos kuitenkin harjoitustyöarvosana on 5, niin hyväksyttyä tenttiarvosanaa (ei kuitenkaan 5:sta) kasvatetaan yhdellä. Kirjoita Python-ohjelma, joka pyytää käyttäjältä opiskelijan tentti- ja harjoitustyöarvosanan sekä tulostaa kurssiarvosanan. (10 p.)

   b) Tutkijalla on bakteeriviljelmä, jonka lämpötilan pitää pysyä määrätyllä välillä, jotta viljelmä ei vahingoitu. Lämpötila mitataan tunnin välein. Jos yksi tai kaksi mitatuista lämpötiloista ei ole sallitulla välillä, viljelmä vielä kestää sen. Mutta jos vähintään kolme lämpötiloista ei ole sallitulla välillä, viljelmä voi vahingoittua. Kirjoita Python-funktio `onko_kunnossa(lampotilat, alaraja, ylaraja)`, joka saa parametreina mitatut lämpötilat sisältävän listan ja sallitun välin ala- ja ylärajan. Funktio palauttaa arvon `True`, jos kaikki mitatut lämpötilat ovat sallitulla välillä tai korkeintaan kaksi niistä on välin ulkopuolella, ja arvon `False`, jos vähintään kolme listan lämpötiloista on sallitun välin ulkopuolella. Huomaa, että välin ulkopuolella olevien arvojen ei tarvitse olla listassa peräkkäin, vaan niiden välissä voi olla sallitulla välillä olevia arvoja. Kirjoita vain pyydetty funktio, älä muita ohjelman osia. (20 p)

3. Kuntoilija on kerännyt tiedot harjoittelustaan tekstitiedostoon niin, että yhdellä rivillä on aina ensin yhden harjoituksen päivämäärä, sitten harjoituksen kesto minuuteissa, harjoituksen taso (rasittavuutta kuvaava kokonaisluku) ja lopuksi lyhyt kuvaus harjoituksen laadusta tekstinä. Eri tiedot on erotettu toisistaan kaksoispisteellä. Tiedoston rivit voisivat näyttää esimerkiksi seuraavilta:

```
6.6.2015:75:4:kevyt kuntosaliharjoittelu
7.6.2015:30:8:kovavauhtinen juoksulenkki
9.6.2015:90:4:reipas kavely
```

   Kirjoita Python-ohjelma, joka laskee tiedostosta kaikkien niiden harjoitusten yhteisajan, joilla on käyttäjän haluama taso. Ohjelma pyytää käyttäjältä tiedot sisältävän tiedoston nimen ja halutun tason. Ohjelman pitää tulosta yhteisaika niin, että kokonaiset tunnit ja niiden yli menevät minuutit ovat erikseen. Jos esimerkiksi ohjelma laskee yllä annetuista tiedoista tason 4 yhteisajan, pitää sen tulostaa 2 tuntia 45 minuuttia eikä 165 minuuttia.

   Ohjelman on käsiteltävä seuraavat virhetilanteet:
   - Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
   - Tiedoston jollain rivillä harjoituksen keston tai tason paikalla ei ole kokonaisluku.

   Näissä tapauksissa ohjelma ilmoittaa käyttäjälle, millainen virhe on sattunut, ja lopettaa toimintansa. Ohjelman ei siis tarvitse jatkaa rivien lukemista virheellisen rivin jälkeen. Voit myös olettaa, että tiedoston jokaisella rivillä on täsmälleen neljä toisistaan kaksoispisteellä erotettua osaa. Ohjelman ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä päivämäärän lisäksi muuta tekstiä. (20 p)

4. Eräs nettitiliyritys tarjoaa käyttöön maksutilin, jonka avulla netissä kauppaa käyvät yksityishenkilöt voivat huolehtia tavaroiden maksusta luotettavalla tavalla. Tili toimii siten, että tavaran ostaja perustaa maksutilin ja tallettaa sille rahaa. Kun hän haluaa tehdä oston, ilmoittaa hän maksutilille, että sille tehdään varaus halutulle summalle, joka vastaa myytävän tavaran hintaa. Tiliyritys välittää myyjälle tiedon siitä, että tarvittava varaus on tehty ostajan tililtä. Tällöin myyjä uskaltaa lähettää tavaran ostajalle. Kun ostaja ilmoittaa tiliyritykselle saaneensa tavaran, tiliyritys nostaa varatut rahat ostajan tililtä ja välittää ne myyjälle. Näin ostajan ei tarvitse varsinaisesti maksaa tavaraa ennen sen vastaanottamista, ja toisaalta myyjän ei tarvitse tavaraa lähettäessään ottaa sitä riskiä, että ostaja saa tavaran, mutta ei maksa sitä. Ostajalla voi olla samanaikaisesti kesken useita kauppoja, eli tililtä voi olla varattu rahoja useaan eri tarkoitukseen.

Kirjoita maksutilin tietojen säilyttämiseen luokka `Maksutili`. `Maksutili`-oliolla on oltava seuraavat kentät:

- `__omistaja` tilin omistajan (ostajan) nimi
- `__saldo` tilillä tällä hetkellä oleva rahasumma (sisältää myös voimassa olevien varausten arvon)
- `__varattu` tilillä tällä hetkellä voimassa olevien varausten yhteissumma (tehtävän helpottamiseksi eri ostoja varten tehtyjä varauksia ei tässä mitenkään eritellä, vaan pidetään yllä tietoa vain siitä, mikä on varausten summa yhteensä).

Määrittele luokkaan seuraavat metodit. (Jos metodin kuvauksessa ei ole kerrottu mitään metodin palauttamasta arvosta, metodin ei tarvitse palauttaa mitään.)

- `__init__(self, nimi)` luo uuden `Maksutili`-olion. Tilin omistajan nimi annetaan parametrina. Uuden tilin saldo on nolla eikä tilillä ei ole varauksia.
- `kerro_omistaja(self)` palauttaa tilin omistajan nimen.
- `talleta_rahaa(self, summa)` tallettaa tilille rahaa (eli kasvattaa tilin saldoa) parametrina annetun summan, jos parametri on postitiivinen. Jos parametri on negatiivinen tai nolla, metodi ei tee mitään.
- `tee_varaus(self, summa)` tekee tililtä varauksen parametrina annetulle summalle, jos tilin saldo riittää varauksen tekemiseen ja annettu summa on positiivinen. Tilillä voimassa olevien varausten yhteissumma ei saa varauksen tekemisen jälkeen ylittää tilin saldoa. Jos varauksen tekeminen onnistuu, metodi palauttaa arvon `True`. Jos tilin saldo ei riitä varauksen tekemiseen tai metodille annettu parametri on negatiivinen tai nolla, metodi palauttaa arvon `False`.
- `maksa_varaus(self, summa)` maksaa tililtä aikaisemmin tehdyn, parametrina annetun summan suuruisen varauksen. Käytännössä varauksen maksaminen tarkoittaa sitä, että sekä tilin saldoa että nykyisten varausten yhteissummaa vähennetään annetulla summalla. (Maksun välittäminen myyjälle hoidetaan jotenkin muuten eikä se näy tässä metodissa.) Varauksen maksu onnistuu vain siinä tapauksessa, että parametrina annettu summa on positiivinen ja että se on korkeintaan yhtäsuuri kuin tilillä tällä hetkellä voimassa olevien varausten summa. Jos varauksen maksu onnistuu, metodi palauttaa arvon `True`. Jos varauksen maksu ei onnistu, metodi ei muuta lainkaan tilin saldoa eikä varausten summaa. Tällöin metodi palauttaa arvon `False`. (Tehtävän helpottamiseksi tässä ei siis tarkisteta sitä, että aikaisemmin on tehty varaus, joka on juuri samansuuruinen kuin nyt suoritettava maksu. Riittää, että voimassa olevia varauksia on yhteensä vähintään yhtä suuresta arvosta kuin mitä on nyt suoritettava maksu. Järjestelmä tarkistaa jotenkin muuten, että kukaan myyjä ei saa enempää maksuja kuin mitä häntä varten on tehty varauksia. Tätä tarkistusta ei kirjoiteta tässä tehtävässä.)
- `peru_varaus(self, summa)` peruu tililtä aikaisemmin tehdyn, parametrina annetun summan suuruisen varauksen, jota ei vielä ole maksettu. Varauksen peruminen tarkoittaa käytännössä sitä, että varausta vastaava summa vähennetään voimassa olevien varausten summasta. Varauksen peruminen onnistuu vain, jos annettu summa on positiivinen ja korkeintaan yhtäsuuri kuin voimassa olevien varausten summa. Jos peruminen onnistuu, metodi palauttaa arvon `True`. Jos peruminen ei onnistu, metodi ei muuta mitään ja palauttaa arvon `False`. Tätä metodia voidaan käyttää esimerkiksi kaupan peruuntuessa.
- `__str__(self)` palauttaa merkkijonon, joka sisältää maksutilin omistajan nimen, tilin saldon ja tällä hetkellä voimassa olevien varausten summan.

Kirjoita lisäksi pääohjelma, joka luo kaksi `Maksutili`-oliota ja sen jälkeen tallettaa molemmille niistä rahaa. Sitten pääohjelman pitää tehdä varaus ensiksi luodulle tilille ja tulostaa, onnistuiko varauksen tekeminen. Sen jälkeen ohjelman pitää maksaa varaus ensiksi luodulta tililtä ja kertoa, onnistuiko se. Lopuksi ohjelman pitää tulostaa molempien maksutilien tiedot (omistajan nimi, tilin saldo ja tällä hetkellä voimassa olevien varausten summa). Voit päättää tilien omistajat sekä talletuksissa, varauksessa ja maksussa tarvittavat rahasummat itse. Pääohjelman ei siis tarvitse kysyä mitään käyttäjältä. (25 p)