CS/CSE-A1111 Ohjelmoinnin peruskurssi Y1 Tentti 15.2.2017

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. Opiskelijat, joiden äidinkieli ei ole suomi, saavat kuitenkin käyttää sanakirjaa, jos siinä ei ole merkintöjä (tentin valvoja tarkistaa sanakirjan). Nämä opiskelijat saavat halutessaan myös sekä suomen- että englannin- tai ruotsinkielisen kysymyspaperin.

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():
    osanottajamaara = 1500
    if osanottajamaara >= 1000:
        print("Paljon porukkaa.")
    if osanottajamaara < 100:
        print("Tulipa vahan osanottajia.")
    else:
        print("Ihan kohtuullinen osanotto.")

main()
```

b) (2 p)

```
def main():
    sinisia = 15
    vihreita = 10
    yhteensa = sinisia + vihreita
    vihreita = 30
    print(yhteensa)

main()
```

c) (3 p)

```
def main():
    lista = [10, 55, 20, 5, 40]
    tulos = 100
    i = 0
    while i < len(lista):
        if lista[i] > 30:
            tulos = tulos - lista[i]
        i = i + 1
    print(tulos)

main()
```

d) (5 p)

```
def vaihda(luku, luvut):
    luku = 5
    luvut[1] = luvut[1] + luku
    return luku

def main():
    lista = [10, 20, 30]
    tulos = 40
    vaihda(tulos, lista)
    print(tulos)
    for kokonaisluku in lista:
        print(kokonaisluku)

main()
```

e) Funktiolle annetaan ensimmäisenä parametrina lista, joka sisältää desimaalilukuja, sekä toisena ja kolmantena parametrina positiivinen desimaaliluku. Funktiossa kutsuttu funktio `abs` palauttaa sille parametrina annetun luvun itseisarvon. (4 p)

```
def mysteeri1(lista, luku1, luku2):
    tulos = 0
    i = 0
    while i < len(lista):
        if abs(luku1 - lista[i]) < luku2:
            tulos = tulos + 1
        i += 1
    return tulos
```

f) Funktiolle annetaan ensimmäisenä parametreina merkkijono, joka sisältää vähintään yhden merkin, ja toisena parametrina positiivinen kokonaisluku. (4 p)

```
def mysteeri2(merkkijono1, luku1):
    luku2 = len(merkkijono1)
    if luku2 >= luku1:
        uusi = merkkijono1[0:luku1]
    else:
        uusi = merkkijono1
        for i in range(luku1 - luku2):
            uusi += "-"
    return uusi
```

g) Funktiolle annetaan parametrina kaksi samankokoista listaa, jotka sisältävät kokonaislukuja. (5 p)

```
def mysteeri3(lista1, lista2):
    i = 0
    while i < len(lista1):
        if lista1[i] < lista2[i]:
            return True
        else:
            return False
        i += 1
```

2. a) Opiskelijajärjestösi on järjestämässä tapahtumaa ja pyytänyt tarjouksen tapahtumapaikan vuokrasta kolmesta eri yrityksestä. Yritys A pyytää koko vuokrausajalta 45 eur / h ja sen lisäksi siivousmaksun, jonka suuruus vaihtelee halutun vuokrauspäivän mukaan. Yritys B pyytää ensimmäiseltä viideltä tunnilta 80 eur / h ja sen jälkeen tulevilta tunneilta 40 eur / h, mutta ei laskuta siivouksesta erikseen. Yritys C pyytää koko vuokrausajalta 70 eur / h eikä laskuta siivouksesta erikseen. Kirjoita Python-ohjelma, joka kysyy käyttäjältä, kuinka moneksi tunniksi tila vuokrataan ja mikä on yrityksen A veloittama siivousmaksu. Ohjelma tulostaa, minkä yrityksen tarjous on halvin ja mitkä ovat tämän halvimman tarjouksen kokonaiskulut (vuokra koko ajalta ja mahdollinen siivousmaksu yhteensä). Jos sama halvin hinta on useammalla kuin yhdellä yrityksellä, riittää, että ohjelma ilmoittaa halvimmista yrityksistä ja hinnoista vain yhden (ei ole väliä minkä). (10 p.)

b) Käytettyjen vaatteiden kauppa myy vaatteita siten, että se pitää osan myyntihinnasta palkkiona itsellään, mutta antaa osan vaatteen alkuperäiselle omistajalle. Jos vaatteen myyntihinta on korkeintaan kaupan asettama raja, kaupan palkkio on 45 % vaatteen myyntihinnasta. Jos vaatteen myyntihinta on suurempi kuin tämä raja, kaupan palkkio on 30 % myyntihinnasta. Palkkio lasketaan siis erikseen kullekin myydylle vaatteelle. Kirjoita Python-funktio `laske_kokonaispalkkio(myyntihinnat, raja)`. Funktio saa ensimmäisenä parametrina listan, joka sisältää kaupan eräänä päivänä myymien vaatteiden myyntihinnat (kukin alkio on yhden vaatteen myyntihinta). Funktion toinen parametri on kaupan asettama raja myyntihinnalle palkkioprosentin määräytymisessä. Funktio laskee ja palauttaa arvonaan myydyistä vaatteista kaupalle yhteensä tulevan palkkion. Voit olettaa, että funktiolle on annettu järkevät parametrit. Tehtävässä ei tarvitse kirjoittaa muuta kuin pyydetty funktio. (20 p)

3. Erään jalkapallojoukkueen ottelutiedot on tallennettu tekstitiedostoon siten, että yhdellä rivillä on aina yhden ottelun tiedot. Ensimmäisenä rivillä on ottelun päivämäärä, toisena vastustajajoukkueen nimi, kolmantena joukkueen ottelussa tekemien maalien määrä (kokonaisluku) ja viimeisenä vastustajajoukkueen ottelussa tekemien maalien määrä (kokonaisluku). Eri tiedot on erotettu toisistaan kaksoispisteellä. Neljä tiedoston riviä voisivat näyttää esim. seuraavilta:

```
10.1.2016:Takahikian potkijat:4:2
20.2.2016:Karhulan kaverit:2:3
4.3.2016:Espoon pallo:2:2
14.3.2016:Hameenkylan huiput:5:1
```

Esimerkkitapauksessa joukkue on siis voittanut ensimmäisen ja neljännen ottelun, hävinnyt toisen ottelun, ja kolmas ottelu on päätynyt tasapeliin. Kirjoita Python-ohjelma, joka pyytää käyttäjältä ottelutulokset sisältävän tiedoston nimen. Ohjelma lukee tästä tiedostosta ottelutiedot ja tulostaa, kuinka monta voittoa, häviötä ja tasapeliä tiedoston otteluissa on. Esimerkkitapauksessa ohjelman pitäisi tulostaa

```
Voittoja 2
Tappioita 1
Tasapeleja 1
```

Ohjelman on käsiteltävä seuraavat virhetilanteet:

- Annetun nimistä tiedostoa ei ole olemassa tai tiedoston lukeminen ei onnistu jostain muusta syystä
- Tiedoston jollain rivillä maalien paikalla olevaa tekstiä ei voi tulkita kokonaisluvuksi.

Näissä tapauksissa ohjelma ilmoittaa käyttäjälle, millainen virhe on sattunut (siis joko että tiedostoa ei pystytä lukemaan tai että rivillä on virheellinen luku), ja lopettaa toimintansa. Ohjelman ei tarvitse jatkaa rivien lukemista virheellisen rivin jälkeen eikä sen tarvitse tulostaa tietoja otteluiden tuloksista, jos jollain rivillä on virheellinen luku. Voit myös olettaa, että tiedoston jokaisella rivillä on täsmälleen neljä toisistaan kaksoispisteellä erotettua osaa. Ohjelman siis ei tarvitse osata käsitellä esimerkiksi sellaisia virhetilanteita, joissa rivi on tyhjä tai ei sisällä päivämäärän lisäksi muuta tekstiä. (20 p)

4. Kirjoita Python-kielellä luokka `Otus` kuvaamaan eräässä tietokonepelissä toimivaa pelihahmoa.

`Otus`-oliolla on oltava seuraavat kentät:

- `__nimi` pelihahmon nimi.
- `__voima` pelihahmon voima, joka vaikuttaa siihen, miten hahmo menestyy taisteluissa. Voimaa kuvataan kokonaislukuna (mitä suurempi luku, sitä enemmän hahmolla on voimaa).
- `__ase_hallussa` kentän arvo on `True`, jos pelihahmolla on hallussaan ase, ja muuten `False`.

JATKUU SEURAAVALLA SIVULLA

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

- `__init__(self, nimi1, alkuvoima)` luo uuden `Otus`-olion. Luotavan pelihahmon nimi ja hahmon alkuperäinen voima on annettu parametreina. Jos viimeinen parametri on negatiivinen, pelihahmon voimaksi asetetaan 0. Uudella pelihahmolla ei ole hallussaan asetta.
- `kerro_nimi(self)` palauttaa pelihahmon nimen.
- `kerro_voima(self)` palauttaa pelihahmon voiman (sitä kuvaavan kokonaisluvun).
- `onko_ase_hallussa(self)` metodi palauttaa arvon `True`, jos pelihahmolla on hallussaan ase ja muuten arvon `False`.
- `poimi_ase(self)` pelihahmo ottaa aseen haltuunsa eli metodi muuttaa tarvittavien kenttien arvoja siten, että hahmolla on metodin suorituksen jälkeen hallussaan ase.
- `luovu_aseesta(self)` metodi muuttaa tarvittavien kenttien arvoja siten, että hahmolla ei ole sen suorituksen jälkeen hallussaan asetta.
- `lepaa(self, tunteja)` metodi kuvaa pelihahmon lepäämistä. Levon kesto tunneissa annetaan metodin parametrina. Lepo vaikuttaa siten, että pelihahmon voima kasvaa jokaista levättyä tuntia kohti 500:lla. Metodi muuttaa pelihahmon voimaa vain siinä tapauksessa, että parametrina annettu tuntimäärä on positiivinen.
- `liiku(self, askeleet)` metodi kuvaa pelihahmon liikkumista. Liikuttavien askelten määrä on annettu parametrina. Jokainen liikuttu askel pienentää hahmon voimaa yhdellä. Liikkuminen ei kuitenkaan saa viedä voimaa negatiiviseksi. Jos esim. hahmon voima on 2000 ja parametrina on annettu 2500, niin hahmo liikkuu vain 2000 askelta. Metodi palauttaa liikuttujen askelten määrän.
- `tappele(self, toinen_otus, kovuus)` pelihahmo haastaa tappeluun parametrina annetun toisen pelihahmon (toisen `Otus`-olion). Viimeisenä parametrina annetaan kokonaisluku 1, 2 tai 3, joka kertoo, kuinka vakavasta tappelusta on kysymys. Tappelun voittaja määräytyy seuraavasti: jos toisella tappelijoista on hallussaan ase ja toisella ei, niin se, jolla on ase, voittaa tappelun. Muussa tapauksessa tappelun voittaja on se, jolla on suurempi voima. Jos kummankin hahmon tilanne aseen suhteen on sama ja kummallakin on yhtäsuuri voima, tappelun voittaa nykyinen `Otus`-olio (`self`). Jos nykyinen hahmo voittaa tappelun, hänen voimansa kasvaa joko 3000:lla, 6000:lla tai 9000:lla sen mukaan, onko tappelun kovuus 1, 2 vai 3 (mitä suurempi kovuus, sitä suurempi voiman kasvu). Jos hän häviää tappelun, hänen voimansa vähenee samalla luvulla ja lisäksi hän menettää aseensa, jos hänellä on sellainen. Voima voi mennä häviön seurauksena myös negatiiviseksi. Metodi ei muuta parametrina saadun vastustajan voimaa tai asetietoa mitenkään (se olisi kyllä järkevää, mutta tekisi tehtävästä vähän vaikeamman). Metodi palauttaa arvon `True`, jos nykyinen pelihahmo voittaa tappelun, ja arvon `False`, jos nykyinen pelihahmo häviää tappelun. Jos tappelun kovuutta kuvaava viimeinen parametri ei ole 1, 2 tai 3, metodi ei tee mitään muuta kuin palauttaa arvon `False`.
- `__str__(self)` palauttaa merkkijonon, joka sisältää pelihahmon nimen, voiman ja joko tekstin "hahmolla on hallussaan ase" tai "hahmolla ei ole hallussaan asetta" sen mukaan, onko hahmolla tällä hetkellä hallussaan ase vai ei.

Kirjoita lisäksi pääohjelma, joka luo kaksi `Otus`-oliota ja ottaa molempien haltuun aseen. Sen jälkeen ohjelman pitää yrittää liikuttaa ensiksi luotua pelihahmoa 500 askelta ja tulostaa, montako askelta tämä oikeasti liikkui. Seuraavaksi toiseksi luodun pelihahmon pitää haastaa ensiksi luotu pelihahmo tappeluun. Pääohjelman pitää tulostaa, voittiko toinen pelihahmo (haastaja) tappelun. Lopuksi ohjelman pitää tulostaa molempien hahmojen tiedot (nimi, voima ja tieto siitä, onko hahmolla hallussaan ase). Pääohjelman ei tarvitse kysyä mitään käyttäjältä, vaan voit päättää olioiden luonnissa ja metodien kutsuissa tarvittavat tiedot itse. (25 p)