Programmeerimise 1. vaheeksam

Neljapäev, 29.10.2015, kell 10:15 – 12:00

1. Porgandi hind (7p) ylesanne1.py

Paul läheb turule porgandeid ostma, kuid teda häirib, et müügisedelitele on tihti kirjutatud palju tarbetut infot. Kirjuta funktsioon porgandi_hind, mis võtab argumendiks müügisedelile kirjutatud sõne ning tagastab seal kirjas olnud hinna **täisarvuna**. Võib eeldada, et sõnes esineb täpselt 0 või 1 tühikute vahel esinevat täisarvu ning arvu leidumise korral tähistab see alati hinda. Juhul, kui sildile hinda märgitud polnud, tuleks tagastada "Küsi hinda!".

Näide:

```
>>> porgandi_hind("Porgandid 5 € kg")
5
>>> porgandi_hind("Suured mahlased porgandid! Osta kohe!")
Küsi hinda!
>>> porgandi_hind("26")
26
```

Vihje:

```
>>> "34".isnumeric()
True
```

2. Üldfüüsiline test (13p) ylesanne2.py

Kaitseväelastel on igal aastal kohustuslik sooritada üldfüüsilise testi (ÜFT) nime kandev füüsilise ettevalmistuse test, mis koosneb kolmest alast: käte kõverdamine toenglamangust (2 min jooksul), istesse tõus selililamangust (2 min jooksul) ning 3200 m jooks. Test loetakse sooritatuks, kui punktide kogusumma on vähemalt 190.

ÜFT tulemused on antud tekstifailina, kus igal real on ühe kaitseväelase ühe ala tulemused kujul:

Kaitseväelase nimi;ala kood;tulemus

Kätekõverduste kood on `kk`, istessetõusu kood on `it` ja jooksu kood on `jooks`. Jooksu tulemus on märgitud kulunud sekundites (täisarvuna).

Read on järjestatud kaitseväelase nime järgi st ühe inimese tulemused on järjestikustel ridadel.

Alamülesanne A. Kirjuta funktsioon `arvuta_punktid`, mis võtab argumendiks ühe ala koodi ja tulemuse ning tagastab tulemusele vastavad punktid (täisarvuna). Punkte arvestame järgmiselt:

- kätekõverdused: *korduste arv + 25;*
- istessetõus: *korduste arv + 15;*
- jooks: (*1300 – kulunud sekundite arv) / 5, tulemus ümardada lähima täisarvuni.*

Näide:

```
>>> arvuta_punktid("kk", 60)
1800
>>> arvuta_punktid("jooks", 1300)
-100
```

Alamülesanne B on lehe pöördel ...

Programmid peaks küsima kasutajalt vaid neid andmeid ja väljastama vaid seda, mida ülesanne nõuab (kui üldse on nõutud). ***Kui sa mõnda ülesande tingimust ei oska täita, siis lihtsusta ülesannet – näiteks lahenda ülesanne mingi erijuhu jaoks. Kui funktsiooniga ei oska, siis tee ilma.*** *Lahendused tuleb salvestada ülesande juures näidatud failinimega ja laadida Moodle'isse. Kindluse mõttes on soovitatav need saata ka enda praktikumijuhendaja e-mailile.*

Alamülesanne B. Kirjuta programm, mis loeb andmed failist nimega *tulemused.txt* (kodeeringus UTF-8) ja väljastab iga kaitseväelase kohta tema nime ja tema punktide summa ehk ÜFT tulemuse ja ka selle, kas läbis ÜFT või mitte.

Näide: kui tulemused.txt sisu on

```
Kaarel Kapsas;kk;70
Kaarel Kapsas;it;130
Kaarel Kapsas;jooks;900
Madis Maasikas;kk;34
Madis Maasikas;it;60
Madis Maasikas;jooks;1300
```

siis ekraanile peaks ilmuma

```
Kaarel Kapsas läbis testi (320p)
Madis Maasikas ei läbinud testi (134p)
```

Vihje: abi on tsüklist, mis teeb nii palju kordusi kui on failis kaitseväelasi. Tsükli igal kordusel tuleks lugeda 3 rida.

Lihtsustus (-2p): kui sa ei oska ülesannet sellisel kujul lahendada, siis lahenda ülesanne sellise faili jaoks, kus ühe inimese andmed on samal real. Näitefailiks võid siis võtta tulemused_lihtsam.txt. Selles failis on andmed kujul

Nimi;kätekõverduste tulemus;istessetõusude tulemus;jooksu tulemus.

Programmeerimise 1. vaheeksam

Neljapäev, 29.10.2015, kell 12:15 – 14:00

1. Avaldise väärtustamine (7p) ylesanne1.py

Kirjuta funktsioon `väärtusta_avaldis`, mis võtab argumendiks sõnena antud lihtsa aritmeetilise avaldise (nt `2 + 4 - 7`) ja tagastab selle avaldise väärtuse. Võib eeldada, et tehtemärkidest kasutatakse ainult plussi ja miinust, sulge pole, tehtemärkidest kummalgi pool on täpselt üks tühik ja kõik avaldises esinevad arvud on positiivsed täisarvud.

Näited:

```
>>> väärtusta_avaldis("1 + 2 + 3")
6
>>> väärtusta_avaldis("1 + 2 - 3 + 45 + 234 - 3245")
-2966
```

Lihtsustus (-2p): kui jääd hätta miinuste ja plussidega, siis kirjuta lahendus selliste avaldiste jaoks, kus ainsateks teheteks on liitmised.

2. Bussipeatused (13p) ylesanne2.py

Failis *peatused.txt* (kodeering UTF-8) on ühe bussiliini sõidugraafik, kus igal real on komaga eraldatult peatuse nimi, sõiduaeg eelmisest peatusest ning kõndimisaeg sellest peatusest mingisse fikseeritud sihtpunkti (nt Herne tänava lõppu). Aeg võib olla antud minutites või sekundites (vaata täpsemalt allolevast näitest). Peatused on esitatud korrektses järjestuses, st buss läbib neid vastavalt nende järjekorrale failis.

Kirjuta programm, mis kuvab ekraanile kõik peatused, mis tuleb läbida selleks, et esimesest peatusest (meie näites Aura juurest) alustades jõuaks võimalikult kiiresti sihtpunkti (meie näites Herne tänava lõppu). NB! Arvestada tuleb bussisõidu aega ja mahatuleku peatusest sihtpunkti kõndimise aega. Kõige lõpus tuleb väljastada ka kogu ajakulu (minutites ja sekundites).

Näide. Kui faili sisu on

```
Aura;0 minutit;24 minutit
Vabaduse;2 minutit;17 minutit
Raeplats;55 sekundit;14 minutit
Palmihoone;1 minut;8 minutit
Kroonuaia;1 minut;6 minutit
Kloostri;40 sekundit;8 minutit
Hurda;1 minut;3 minutit
Kreutzwaldi;1 minut;4 minutit
```

siis peab programmi väljund olema

```
Aura
Vabaduse
Raeplats
Palmihoone
Kroonuaia
Aega kulub 11 minutit ja 0 sekundit.
```

Kui bussi kasutamine ajavõitu ei anna, peab programm väljastama ainult kõndimiseks kuluva aja.

Alamülesanne. Programmis peab olema funktsioon `sekundid`, mis võtab argumendiks sõne kujul `"1 minut"`, `"1 sekund"`, `"`*<täisarv>*` minutit"` või `"`*<täisarv>*` sekundit"` ning tagastab täisarvu, mis näitab, mitut sekundit antud sõne kujutab.

Näited:

```
>>> sekundid("3 minutit")
180
>>> sekundid("1 minut")
60
>>> sekundid("50 sekundit")
50
```

Programmid peaks küsima kasutajalt vaid neid andmeid ja väljastama vaid seda, mida ülesanne nõuab (kui üldse on nõutud). ***Kui sa mõnda ülesande tingimust ei oska täita, siis lihtsusta ülesannet – näiteks lahenda ülesanne mingi erijuhu jaoks. Kui funktsiooniga ei oska, siis tee ilma.*** *Lahendused tuleb salvestada ülesande juures näidatud failinimega ja laadida Moodle'isse. Kindluse mõttes on soovitatav need saata ka enda praktikumijuhendaja e-mailile.*

Programmeerimise 1. vaheeksam

Neljapäev, 29.10.2015, kell 16:15 – 18:00

1. Tühikute korrastamine (7p) *ylesanne1.py*

Kirjuta funktsioon `korrasta_tühikud`, mis võtab argumendiks teksti ja tagastab korrastatud teksti, kus kirjavahemärkide ees ei ole tühikut ja kirjavahemärgi järel (v.a. teksti lõpus oleva kirjavahemärgi lõpus) on täpselt üks tühik. Teksti alguses ja lõpus olevad tühikud tuleb eemaldada. Kirjavahemärkideks loeme punkti, koma, hüüumärki ja küsimärki.

Näited:

```
>>> korrasta_tühikud("See on lause, kus pole midagi valesti.")
'See on lause, kus pole midagi valesti.'
>>> korrasta_tühikud("  Siin on see , see  , ja see probleem  ! Siin ka . ")
'Siin on see, see, ja see probleem! Siin ka.'
```

Lihtsustus (-2p): kui 4 kirjavahemärgiga arvestamine läheb liiga keeruliseks, siis korrasta ainult komade ümbrus ning kogu teksti algus ja lõpp.

2. Saatjate domeenid (13p) *ylesanne2.py*

Failis *mbox.txt* (kodeering UTF-8) (allikas http://www.py4inf.com/code/mbox.txt) on mbox formaadis kellelegi saabunud kirjad. Kirjuta programm, mis kuvab ekraanile kõik **erinevad** e-maili domeenid (st e-maili aadressis @-märgist paremale jääv osa), millelt on kirju saadetud. Selleks tuleb otsida välja kõik read, mis algavad tekstiga `"From: "`. Eeldame, et ühegi kirja põhitekstis ükski rida nii ei alga.

Lõpuks tuleb ekraanile kuvada kõige pikem domeen (kui pikimaid on mitu, siis võib kuvada neist suvalise) ning saadud kirjade arv.

Antud näitefaili korral peab ekraanile ilmuma:

```
uct.ac.za
media.berkeley.edu
umich.edu
iupui.edu
caret.cam.ac.uk
gmail.com
indiana.edu
et.gatech.edu
vt.edu
lancaster.ac.uk
ucdavis.edu
ufp.pt
txstate.edu
stanford.edu
whitman.edu
rsmart.com
fhda.edu
bu.edu
unicon.net
loi.nl
utoronto.ca
Kõige pikem domeen: media.berkeley.edu
Kokku kirju: 1797
```

Alamülesanne. Programmis peab olema funktsioon `eralda_domeen`, mis võtab argumendiks saatja rea (nt. `"From: cwen@iupui.edu"`) ja tagastab e-maili aadressi domeeni.

Näide:

```
>>> eralda_domeen("From: cwen@iupui.edu")
iupui.edu
```

Lihtsustus (-2p): kui sa ei oska e-maili domeeni eraldada, siis tee ülesanne tervete e-maili aadressidega.

Programmid peaks küsima kasutajalt vaid neid andmeid ja väljastama vaid seda, mida ülesanne nõuab (kui üldse on nõutud). ***Kui sa mõnda ülesande tingimust ei oska täita, siis lihtsusta ülesannet – näiteks lahenda ülesanne mingi erijuhu jaoks. Kui funktsiooniga ei oska, siis tee ilma.*** *Lahendused tuleb salvestada ülesande juures näidatud failinimega ja laadida Moodle'isse. Kindluse mõttes on soovitatav need saata ka enda praktikumijuhendaja e-mailile.*