Izpit iz predmeta Programiranje 1, 15. februar 2011

Pri reševanju nalog je dovoljena raba standardnih Pythonovih knjižnic (`os`, `random`, `math`...), razen pri nalogi, kjer je to eksplicitno prepovedano. Ali je neka knjižnica standardna knjižnica ali ne, boste najlažje prepoznali po tem, ali je opisana v Pythonovi dokumentaciji.

Funkcije morajo imeti takšna imena, kot jih predpisuje naloga. Če naloga zahteva, naj funkcija *vrne rezultat*, mora *vrniti rezultat*, ne pa *izpisovati*. Funkcija mora podatke dobiti prek argumentov, kot določa naloga.

Rešitve vseh nalog shranite v eno samo datoteko s končnico .py in jo oddajte prek Učilnice tako, kot ste oddajali domače naloge. Datoteka naj bo takšna, da jo je mogoče izvesti s Pythonom (morebitne komentarje oblikujte kot komentarje!)

Izpitu so priloženi testni primeri. Uporabite jih. (Poženite svojo skripto, nato skripto s testi.) Zavedajte, da je lahko program napačen tudi, če pravilno reši vse pripravljene testne primere.

Pri reševanju nalog je dovoljena vsa literatura na poljubnih medijih.

Študenti s predolgimi vratovi in podobnimi neželenimi lastnostmi bodo morali zapustili izpit, katerega opravljanje se bo štelo kot neuspešno. Hujše kršitve bomo prijavili disciplinski komisiji za študente.

Nalogi A in B se ne točkujeta, temveč sta obvezni za uspešno opravljen izpit. Kdor ju ne reši pravilno, je s tem že padel.

Vse ostale naloge so vredne enako število točk.

A. Pari poznanstev (obvezna naloga!)

Vzemi **SVOJO** rešitev domače naloge Mreža poznanstev. (Samo tisti, ki naloge niso oddali, oziroma jim je bila ocenjena negativno, naj vzamejo objavljeno rešitev.) Spremeni program tako, da bo mimogrede izpisal vse pare znancev – torej vsak par ljudi, ki se kdaj pojavi skupaj na kaki sliki. Pri tem *ni potrebno paziti,* da bi se vsak par izpisal le enkrat; izpiše se lahko poljubnokrat. Poskrbite pa, da se ne bo izpisalo, da je nekdo sam svoj znanec. (Namig: `print` smete dodati kar v kodo,ki sestavlja slovar.)

Spremenite jo, da bo delovala tako:

```
Osama Abdul
Abdul Osama
Osama Abdulah ibn Husein
Abdulah ibn Husein Osama
Osama Ahmed
Osama Mustafa
Ahmed Osama
Ahmed Mustafa
Mustafa Osama
Mustafa Ahmed
Abdulah ibn Husein Husein
Husein Abdulah ibn Husein
Ahmed Mustafa
Ahmed Jibril
```

in tako naprej

B. Sestavljena števila (obvezna naloga!)

Vzemi **SVOJO** rešitev domače naloge Razcep na prafaktorje. (Samo tisti, ki naloge niso oddali, oziroma jim je bila ocenjena negativno, naj vzamejo objavljeno rešitev.) Preimenuj funkcijo `praštevilo` v `sestavljeno_stevilo` in jo spremeni tako, da bo vrnila True, če je podano število sestavljeno (torej, če *ni praštevilo*).

.

1. Sekunde

Ljudje čas radi zapisujemo v obliki, kot je, na primer, 8:12:34, kar bi, recimo, pomenilo 12 minut in 34 sekund čez osem ali 17:4:45, kar je štiri minute in 45 sekund čez peto popoldan. Računalniki čas raje merijo v sekundah – za potrebe te naloge, ga bomo merili v sekundah od polnoči. Napiši naslednje funkcije:

- `cas_v_sekunde(s)`, ki prejme čas v »človeški« obliki in kot rezultat vrne čas od polnoči. Upoštevajte, da sme človek napisati bodisi 10:05:20 bosidi 10:5:20 – funkcija naj zna brati oboje.
- `sekunde_v_cas(s)`, ki prejme čas v sekundah od polnoči in vrne niz v človeški obliki. Funkcija naj rezultat vrne v obliki 10:05:20, ne 10:5:20.
- `razlika_casov(s1, s2)`, ki prejme dva časa v človeški obliki in vrne razliko med njima, spet zapisano v človeški obliki. Predpostaviti smeš, da je `s2` večji od `s1`.

Pri reševanju naloge je prepovedano uporabljati Pythonove module.

Primer

```
>>> cas_v_sekunde("10:00:00")
36000
>>> cas_v_sekunde("0:0:0")
0
>>> cas_v_sekunde("10:05:12")
36312
>>> sekunde_v_cas(36312)
'10:05:12'
>>> sekunde_v_cas(0)
'00:00:00'
>>> razlika_casov("10:00:00", "10:11:5")
'00:11:05'
>>> razlika_casov("10:00:00", "12:11:5")
'02:11:05'
>>> razlika_casov("10:11:30", "12:05:35")
'01:54:05'
```

2. Uporabniške skupine

Unix zapisuje skupine uporabnikov v datoteko `/etc/group`. Vsaka vrstica opisuje eno skupino uporabnikov. V njej so štirje podatki, ločeni s podpičji: ime skupine, geslo, številski id in seznam uporabnikov v skupini. Uporabniška imena so ločena z vejicami. Na primer vrstica

`ordinary:x:504:ana,berta,cilka,dani`

pravi, da so v skupini `ordinary` uporabnice z imeni `ana`, `berta`,`cilka` in `dani`. Geslo in številka sta nepomembni.

Napiši funkcijo `beri_skupine(fname)`, ki kot argument prejme ime datoteke `groups` (uporabite datoteko, ki je priložena izpitu), kot rezultat pa vrne slovar, katerega ključi so uporabniki, vrednosti pa skupine, v katerih je ta uporabnik (torej nekako obratno od tega, kar je v groups, kjer za skupino izvemo, katere uporabnike vsebuje).

Primer

```
>>> users = beri_skupine("groups")
>>> users["ana"]
['ana', 'ordinary', 'wusers']
>>> users["cilka"]
['cilka', 'ordinary', 'taccess', 'wusers']
>>> users["dani"]
['dani', 'ordinary']
```

3. Hamming

Napiši funkcijo `naj_prileg(s, sub)`, ki v nizu `s` poišče mesto, ki se mu niz `sub` najbolj prilega. Funkcija naj vrne indeks v nizu `s`, število ujemajočih se črk in podniz znotraj `s`.

Primer

```
>>> naj_prileg("Poet tvoj nov Slovencem venec vije", "vine")
(24, 3, 'vene')
```

Besede "vine" v nizu ni, pač pa se z njim najbolj ujeme "vene", ki se nahaja na 24 znaku, saj se ujemata v treh črkah. Zato je rezultat funkcije (24, 3, "vene").

Kadar se enako dobro prilega več mest, naj funkcija vrne prvo izmed njih.

Predpostaviti smete, da je iskani podniz neprazen in krajši od dolžine niza, v katerem iščete.

Pri reševanju naloge je prepovedano uporabljati modul BLAST. ☺

4. Vsa imena

Imena (*identifier*) spremenljivk, funkcij, razredov, modulov... v Pythonu se začnejo z veliko ali malo črko ali podčrtajem, ki jim sledi poljubno število (lahko tudi nič) črk, podčrtajev in števk. Izjema so ključne besede (*keywords*):

```
keywords = ["and", "del", "from", "not", "while", "as", "elif", "global",
            "or", "with", "assert", "else", "if", "pass", "yield", "break",
            "except", "import", "print", "class", "exec", "in", "raise",
            "continue", "finally", "is", "return", "def", "for", "lambda",
            "try"]
```

Ključne besede niso imena. Napiši funkcijo imena(fname), ki kot argument prejme ime datoteke s programom, kot rezultat pa vrne seznam vseh imen v njem. Seznam naj bo urejen in vsako ime naj se pojavi le enkrat.

Pri tem izločite »imena«, ki so v resnici komentarji, torej, ne pregledujte delov vrstic, ki sledijo prvemu znaku #. »Imen«, ki so v resnici v nizih, torej znotraj narekovajev, vam ni potrebno izločati.

Primer

```
>>> imena("resitve-11-02-15.py")

['A', 'Za', 'append', 'beri_skupine', 'c1', 'c2', 'cas_v_sekunde', 'compile',
'difference', 'file', 'findall', 'fname', 'group_id', 'group_list', 'group_name', 'h',
'i', 'imena', 'int', 'keywords', 'len', 'line', 'm', 'naj_prileg', 'najpos', 'najuj',
'password', 'pos', 'range', 'razlika_casov', 're', 're_ident', 's', 's1', 's2',
'sekunde_v_cas', 'set', 'sorted', 'split', 'spremenljivke', 'strip', 'sub', 'uj',
'update', 'user', 'users', 'w', 'z_', 'zip']
```

5. Največ n-krat

Napiši funkcijo `najvec_n(l, n)`, ki kot argument prejme seznam `l` in največje dovoljeno število ponovitev elementov na seznamu, `n`. Funkcija vrne nov seznam, v katerem se vsak element pojavi največ `n`-krat, vse nadaljnje ponovitve pa se pobrišejo.

Če ti kaj pomaga, smeš predpostaviti, da seznami niso dolgi.

Primer

```
>>> najvec_n([1, 2, 3, 1, 1, 2, 1, 2, 3, 3, 2, 4, 5, 3, 1], 3)

[1, 2, 3, 1, 1, 2, 2, 3, 3, 4, 5]
```

Namig: nikar ne briši, sestavljaj nov seznam!