Izpit iz predmeta Programiranje 1, 22. januar 2011 ob 14.30.

Vse rešitve shranite **v eno samo datoteko s končnico .py** in jo oddajte prek Učilnice. Vse funkcije naj imajo takšna imena, kot jih predpisuje naloga. **Pozorno preberite** naloge in ne rešujte le na podlagi primerov!

Da rešitev ne bi imela trivialnih napak, **jo preverite s testi** v ločeni datoteki na Učilnici. Za rešitev naloge lahko dobite določeno število točk, **tudi če ne prestane testov**. Funkcija, ki prestane vse teste, **še ni nujno pravilna**.

Pri reševanju nalog je dovoljena vsa literatura na poljubnih medijih, ves material, ki je objavljen na Učilnici, vključno z objavljenimi programi; njihova uporaba in predelava se ne šteje za prepisovanje.

Izpit morate pisati na fakultetnih računalnikih, ne lastnih prenosnikih.

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

1. Sosedi

Za neko krožno ulico (hiše so razporejene v krogu) imamo seznam, ki pove koliko ljudi živi v vsaki od hiš. Napišite funkcijo `stevilo_sosedov(prebivalci)`, ki za vsako hišo pove koliko ljudi živi v sosednjih dveh hišah. Za seznam `[1,2,0,5]` naj funkcija vrne `[7,1,7,1]`. Predpostavite lahko, da so v vsaki krožni ulici vsaj tri hiše.

2. Trki

Podana je funkcija `skrij(beseda)`, ki na določen način predeluje besede: kot argument dobi besedo (npr. “HANA”) in kot rezultat vrne predelano besedo (“A6N3H1”). Kako funkcija deluje, ni pomembno. Še več, testi lahko uporabljajo različne funkcije za predelavo besed.

Napiši funkcijo `trk(besede)`, ki dobi seznam besed in vrne par besed, ki ju funkcija skrij predela v isto besedo. Če je takšnih parov več, lahko vrne poljubnega med njimi. Če se noben par ne spremeni v isto besedo, naj vrne `None`.

3. Poštar iz Hamiltona

Nek duhovit poštar raznaša pošto izključno v neki zelo zelo dolgi ulici. Pri tem ne hodi od hiše do hiše po vrsti: odločil se je, da ne bo nikoli naredil poti med dvema hišama, ki si ne delita vsaj ene skupne števke. Od hiše številka 75 lahko gre k hiši 78 (zaradi sedmice), 52 (zaradi petice), 107 (zaradi sedmice), ne pa k 43 ali 88, ki s 75 nimata nobene skupne števke. Seveda gre vsak dan le k hišam, za katere ima kako pošto.

Zaradi te čudne navade mora vsak dan dobro splanirati svojo pot, da bo raznesel vso pošto – pa še tako se mu kak dan zgodi, da ne more raznositi vse pošte, saj ni poti, ki bi ustrezala pravilom.

Napišite funkcijo `postar(naslovi)`, ki kot argument dobi seznam številk, h katerim mora prinesti pošto, kot rezultat pa vrne `True`, če bo lahko raznosil vso pošto in `False`, če ne.

Nasvet: mogoče vam bo lažje, če najprej napišete funkcijo preveri_zaporedje(zaporedje), ki dobi neko zaporedje številk in pove, ali je to pravilna pot v tem smislu, da si vsak zaporeden par številk deli vsaj eno števko. Potem napišete zahtevano funkcijo postar tako, da le-ta kliče preveri_zaporedje(zaporedje).

Namig: toplo priporočam, da pred reševanjem preverite, kaj izpiše program

```
from itertools import permutations

for x in permutations(["Ana", "Berta", "Cilka", "Dani"]):
    print(x)
```

4. Največ sester

Imejmo rodovnike, ki so podani tako, kot so bili na predavanjih iz rekurzije. Napišite funkcijo `najvec_sester(ime, rodovnik)`, ki pove, koliko sester ima oseba z največ sestrami. Funkcija dobi kot argument slovar z rodovnikom in ime osebe, med katere potomci iščemo (sestre podane osebe nas **ne** zanimajo). Delo si poenostavimo tako kot na predavanjih: ime je žensko, če se konča s črko "a".

Pazi: v družini z otroki Ana, Berta, Jože sta dve hčeri in nekdo (Jože) ima dve sestri. V družini z otrokoma Ano in Berto sta prav tako dve hčeri, vendar ima vsako od njiju le eno sestro.

Nasvet: mogoče vam bo lažje, če najprej napišete funkcijo `sester_pod(ime, rodovnik)`, ki pove, koliko sester ima tisti otrok osebe `ime`, ki ima največ sester.

5. Blagajna

Definirajte razred `Blagajna` z metodami

- `dodaj(self, bankovec)`, ki v blagajno doda bankovec s podano vrednostjo;
- `vzemi(self, bankovec)`, ki iz blagajne vzame bankovec s podano vrednostjo in vrne `True`; če v blagajni ni nobenega bankovca s to vrednostjo, pa ne naredi ničesar in vrne `False`;
- `bilanca(self)`, ki vrne vsoto vseh bankovcev v blagajni.