Izpit iz predmeta Programiranje 1, 6. februar 2014 ob 12.00. Čas pisanja: 90 minut.

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 zapustiti izpit, katerega opravljanje se bo štelo kot neuspešno. Hujše kršitve bomo prijavili disciplinski komisiji.

1. Ostanki

Količina nektarja v cvetovih enodimenzionalnega vrta (recimo potke) je opisana s seznamom števil, na primer `[5, 1, 3, 2, 5, 7, 3, 12, 2]`. Po vrtu (proti vsem pričakovanjem) leta čebela; vsakič, ko pride na cvet, pobere ves nektar v njem. Pot čebele je opisana s seznamom pozitivnih in negativnih števil: 4 pomeni pomik za štiri polja desno, -7 pomeni pomik za sedem polj levo. Čebela začne na polju 0 in se pomika, kot zahteva seznam. Pomikanje se ustavi ko

- je konec poti (pridemo do konca seznama korakov) ali pa
- čebela pride na cvet brez medu (ker ga je že obrala ali pa je bil prazen že na začetku).

Napišite funkcijo `ostanki(vrt, pot)`, ki simulira pot čebele in vrne količino nektarja, ki **ostane v vrtu**, ko se čebela ustavi. Predpostaviti smete, da pot ne vsebuje skokov prek meja vrta.

Funkcija ne sme spremeniti nobenega od podanih seznamov. (Namig: skopirajte seznam.)

Nasvet: ne pozabi, da čebela obišče en cvet več, kot je dolga pot. Če je pot [5, -2], bo obrala ničti, peti in tretji cvet.

2. Pretakanje

Imejmo posodi A in B, ki držita 5 in 8 litrov. Poleg tega imamo ogromno posodo P, ki drži 1000 litrov. Posodi A in B sta v začetku prazni, P pa napol polna mleka. Med posodami bomo pretakali mleko tako, da bomo vedno napolnili posodo, v katero točimo, ali pa izpraznili posodo, iz katere točimo. Pretakanje A->B torej pomeni, da točimo mleko iz A v B toliko časa, da je B polna do roba ali pa da je A prazna.

Pretakanje mleka med posodami opišemo s seznamom nizov. Tako `["P->A", "A->B", "P->A", "A->B"]` pomeni, da najprej napolnimo posodo A iz posode P, nato pretočimo A v B (ker je B večja, bo šla vanjo vsa vsebina posode A), nato spet napolnimo A in spet točimo A v B. Ker je v B že pet litrov, ki smo jih natočili prej, bodo šli vanjo le še trije litri mleka iz posode A in v A bosta na koncu ostala dva litra.

Napišite funkcijo `pretakanje(s)`, ki prejme seznam, podoben gornjemu in kot rezultat vrne količino mleka v prvi in v drugi posodi (kot dve števili – na tak način, kot funkcije v Pythonu vračajo več rezultatov).

Predpostaviti smete, da nikoli ne točimo iz iste posode v isto (na primer A -> A).

3. Izplačilo

Napiši funkcijo `izplacilo(bankovci, znesek)`, ki dobi slovar, ki pove, koliko katerih bankovcev je v blagajni in znesek, ki ga je potrebno izplačati. Če je slovar enak `{100: 8, 20: 7, 10: 4}`, je v blagajni 8 stotakov, 7 dvajsetakov in 4 desetaki. Znesek je število (`int`), recimo 130. Znesek vedno izplačujemo tako, da začnemo z večjimi bankovci in nadaljujemo proti manjšim. Če je potrebno izplačati 130 evrov, bomo izdali en bankovec za 100, enega za 20 in enega za 10 in ne, recimo, šestih

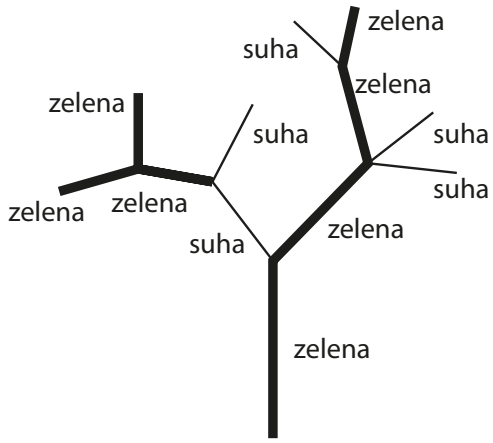
bankovcev za 20 in enega za 10 ali pa enega za 100 in treh za 10 – razen, kadar ne gre drugače, ker, recimo, nimamo bankovca za 100.

Funkcija **ne sme** vračati ničesar, temveč naj **spremeni slovar** `bankovci`, ki ga je dobila kot argument, tako, da bo odražal število bankovcev, ki ostanejo v blagajni po izplačilu. Če katere vrste bankovcev ni več, jih vrzite tudi iz slovarja: če je potrebno izplačati 800 evrov, mora biti vsebina slovarja po tem `{10: 4, 20: 7}` in ne `{100: 0, 20: 7, 10: 4}`. Če je potrebno izplačati 950 evrov, bo vsebina slovarja `{10: 3}`. Predpostaviti smete, da ima blagajna vedno dovolj bankovcev, da lahko opravi izplačilo.

Namiga: najprej razmisli, kako bi nalogo rešil ročno. Klic `sorted(bankovci.items(), reverse=True)` morda vrne kaj uporabnega.

4. Črv

Na spodnji sliki je drevo s suhimi in zelenimi vejami ter njegova predstavitev s slovarjem. (Kako lahko raste zelena veja iz suhe, ne vem, ampak tako pač je – v tej nalogi).




```
{"zelena": {
    "suha": {
        "zelena": {
            "zelena": {},
            "zelena": {}
        },
        "suha": {}
    },
    "zelena": {
        "zelena": {
            "suha": {},
            "zelena": {}
        },
        "suha": {},
        "suha": {}
    }
}}
```

Črv začne lesti po drevesu od spodaj. Zlesti poskuša čim dalj, vendar nikoli ne gre na suho vejo. Napišite funkcijo `crv(drevo)`, ki prejme drevo v takšni obliki (seveda pa ne nujno točno takšnega drevesa!) in vrne najdaljšo možno pot, ki jo lahko črv naredi po tem drevesu.

5. Računalnik

Imamo razred `Racunalnik` (koda na desni), ki ima metodo izračun, ki prejme tri argumente. Zadnji argument je geslo. Metoda najprej pokliče metodo `preveri_geslo`, ki preveri pravilnost gesla in vrne `True` (pravilno) ali `False` (napačno). Če je geslo pravilno, `izracunaj` izračuna nekaj zapletenega iz prvih dveh argumentov in vrne rezultat. Če je napačno, `izracunaj` ne vrne ničesar.

```
class Racunalnik:
    def izracunaj(self, a, b, geslo):
        if self.preveri_geslo(geslo):
            return a + b

    def preveri_geslo(self, geslo):
        return True
```

Trenutno metoda `preveri_geslo` vedno vrne `True`. Iz razreda Racunalnik izpeljite razreda `Racunalnik_Sodo` in `Racunalnik_Sudo`, ki bosta pametneje preverjala gesla: prvi opravi izračun, če kot geslo podamo katerokoli sodo število. Drugi opravi izračun, če kot geslo podamo niz `"sudo"`.

Razred `Racunalnik` je že napisan in ga ne smete spreminjati. Poleg tega v izpeljanih razredih ne smete definirati metode izracunaj; ta je že napisana in mora ostati takšna, kot je. Nalogo morate rešiti na tak način, kot se to z objekti počne.