Izpit iz predmeta Programiranje 1, 15. februar 2017 ob 12.00. Čas pisanja: 90 minut.

Vse rešitve shranite **v eno samo datoteko s končnico .py** in jo oddajte prek Učilnice. Za rešitev naloge lahko dobite določeno število točk, **tudi če ne prestane testov**. Funkcija, ki prestane vse teste, **še ni nujno pravilna**. Upošteva se tudi eleganca rešitve.

Dovoljena je uporaba vseh materialov na Učilnici in druge literature na poljubnih medijih. Prepovedana je vsaka komunikacija.

1. Ne maram

Imamo seznam parov oseb, ki nočejo biti skupaj, na primer `[("Ana", "Cilka"), ("Berta", "Ana"), ("Berta", "Dani")]`. Imamo vrstni red oseb, na primer, `["Ana", "Berta", "Cilka", "Dani", "Ema"]`. Potrebujemo samo še funkcijo, `preveri_vrsto(vrsta, prepovedani)`, ki vrne `True`, če bodo uvrščenke zadovoljne z vrsto in `False`, če ne. Za gornji primer vrne `False`, ker Ana in Berta nočeta biti ena zraven druge.

Vrsta ne vsebuje nujno nizov, lahko so tudi številke. Oba seznama sta lahko tudi zelo dolga. Če bo vaša funkcija delovala le s kratkimi seznami, boste bodili 2/3 točk.

2. Odprepovej

Napiši funkcijo `odprepovej(vrsta, prepovedani)`, ki reši problem prepovedanih sosedov tako, da iz seznama prepovedani pobriše tiste pare, ki sedijo skupaj, čeprav ne bi hoteli - tako da bo izgledalo, kot da je vse v redu.

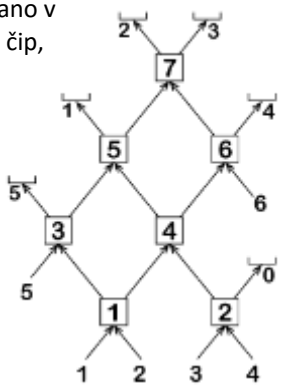
Funkcija ne vrača ničesar, temveč spremeni podani seznam. Zadnji odstavek prejšnje naloge velja tudi za to.

3. Najbolj oddaljena

Imamo seznam krajev in njihovih koordinat, kot v domači nalogi Razdalje med kraji. Druga koordinata gre v smeri sever-jug in narašča proti severu. Napiši funkcijo `najbolj_oddaljena(kraj, kraji)`, ki vrne imeni tistih dveh krajev, ki sta južno od podanega (podani ni vključen!) in sta najbolj oddaljena eden od drugega.

4. Kdo dobi čip

Napiši funkcijo `kdo_dobi(bot, boti)`, ki kot argument dobi številko bota in mrežo, opisano v enaki obliki kot v domači nalogi. Funkcija naj vrne množico vseh botov, ki bi lahko dobili čip, ki ga ima podani bot. Množica naj vsebuje tudi podanega bota.



Če je `boti` mreža na desni, klic `kdo_dobi(4, boti)` vrne množico `{4, 5, 6, 7}`. Klic `kdo_dobi(7, boti)` vrne `{7}`. Klic `kdo_dobi(2, boti)` vrne `{2, 4, 5, 6, 7}`.

Funkcija naj seveda deluje tudi pri nekoliko večjih (čeprav ne nujno tudi zelo velikih) mrežah. Najbrž bo rekurzivna.

5. Ladja

Napiši razred `Ladja` z naslednjimi metodami.

- Konstruktor prejme nosilnost ladje.
- `natovori(teza)` natovori paket s podano težo. Če skupna teža s tem preseže nosilnost ladje, z nje vržejo toliko paketov, da je teža spet v okviru dovoljene. Pakete odstranjujejo v enakem vrstnem redu, kot so jih nalagali, torej najprej najstarejšega. Metoda ne vrne ničesar.
- `skupna_teza()` vrne skupno težo vseh paketov na ladji.
- `odstranjenih()` vrne število paketov, ki so bili doslej odstranjeni z ladje zaradi preseganja nosilnosti.
- Če je ladja objekt tipa `Ladja`, potem `len(ladja)` vrne število vseh paketov na ladji.

Recimo, da imamo ladjo z nosilnostjo 42.

- Nanjo natovorimo paket s težo 30 in nato paket s težo 10. Skupna teža je 40.
- Dodajo paket s težo 21. Ker je nosilnost presežena (30 + 10 + 21 = 61 > 42), z ladje vržejo prvi paket, 30. Skupna teža je zdaj 10 + 21 = 31, na ladji sta dva paketa in število odstranjenih je 1.
- Nato natovorijo paket s težo 41. Nosilnost je presežena (10 + 21 + 41 = 72 > 42), zato bodo vrgli z ladje prvi paket (10). Ker je nosilnost še vedno presežena, vržejo z ladje še drugi paket. Ostane le en paket, skupna teža je 41, število doslej odstranjenih je 3.
- Nato natovorijo paket s težo 50. Z ladje vržejo paket s težo 41. Ker je nosilnost še vedno presežena, odstranijo še pravkar naloženi paket (50). Skupna teža je 0, število odstranjenih je 5.