Izpit iz predmeta Programiranje 1, 12. februar 2020 **Čas pisanja: 60 minut (s podaljšenjam: 90 minut)**

Rešitev 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 kvaliteta rešitve.

Dovoljena je uporaba vseh materialov na Učilnici in druge literature na poljubnih medijih. Prepovedan je dostop do vseh drugih spletnih strani in vsaka oblika komunikacije, razen s profesorjem oz. asistentom.

1. Menjave

Napiši funkcijo `menjave(razpored, zamenjave)`, ki prejme seznam z začetnim razporedom nekih oseb, na primer `["Ana", "Berta", "Cilka", "Dani", "Ema]`, in seznam parov števil, ki predstavljajo menjave. Seznam menjav `[(0, 4), (1, 2), (0, 2)]` pomeni, da se najprej zamenjata osebi na mestih 0 in 4, nato osebi na mestih 1 in 2, nato osebi na mestih 0 in 2.

Funkcija mora spremeniti podani seznam razpored tako, da bo ustrezal končnemu razporedu. V gornjem primeru je to `["Berta", "Cilka", "Ema", "Dani", "Ana"]`. Kot rezultat pa mora funkcija vrniti množico vseh oseb, ki so bile kdaj na mestu 0. V gornjem primeru so to `{"Ana", "Ema", "Berta"}`.

2. Pari

Napiši funkcijo `najblizji_par(s)`, ki prejme seznam *različnih* števil in vrne par najbližjih števil. Števili naj bosta urejeni po velikosti. Klic `najblizji_par([2, -2, 7, 10, 5, 20])` vrne `(5, 7)`. Kadar obstajata dva para z enako razliko, naj vrne tistega z manjšimi števili.

Napiši funkcijo `pari(s)`, ki prejme seznam števil in vrne seznam najbližjih parov, dobljenih po naslednjem postopku. Prvi par sta najbližji števili v seznamu `s` (izbrani na enak način kot v prejšnjem odstavku). Drugi par sta najbližji števili izmed preostalih števil v seznamu. Tretji par sta najbližji števili izmed preostalih ... in tako naprej. Če ima seznam liho število elementov, preostali element ignoriramo.

Klic `pari([2, 5, 6.5, -2, 10, 20])` vrne `[(5, 6.5), (-2, 2), (10, 20)]`.

3. Bomboni

Ana in Berta sta dobili oštevilčene škatlice. V vsaki je en bombon. Nato istočasno odpirata škatlice: vsaka odpre eno, vzame bombon (če je še tam) in jo zapre nazaj. Ker sta Ana in Berta še majhni, sproti pozabljata, katero škatlico sta že izpraznili. Če hkrati poskušata odpreti isto škatlico, vzame bombon mama. Da se ne kregata.

Recimo, da Ana odpira škatlice v takšnem vrstnem redu:
`[4, 1, 4, 7, 4, 3, 5, 6, 8, 5, 3, 2, 4, 6]` in Berta v takšnem:
`[1, 3, 5, 4, 6, 1, 2]`. Najprej dobita bombone iz škatlic 4 in 1. Nato nesrečna Ana odpre škatlo 1, kjer ni več bombona, Berta pa poje bombon iz škatle 3. Nato Ana odpre škatlico 4 (smola!) Berta poje bombon iz škatlice 5 ...

Ana je dobila bombone iz škatlic 4, 7 in 8, Berta pa bombone iz škatlic 1, 3, 5 in 6. Seveda se lahko zgodi, da ena odpira škatlice dlje (npr. Ana, ki jih odpira še dolgo po tem, ko Berta že bruha).

Napiši funkcijo `bomboni(s, t)`, ki prejme dva seznama, kot sta zgornja, in vrne par števil, ki pove, koliko bombonov je dobil kdo. V gornjem primeru vrne 3 in 4. Funkcija naj se izvede v doglednem času tudi, če je bombonov veliko.

4. Izmenična vsota

Napiši rekurzivno funkcijo `izmenicna_vsota(s)`, ki vrne vrednost `s[0] - s[1] + s[2] - s[3] + s[4] - s[5] + ...`

5. Naloge

Napiši razred `Naloge` z naslednjimi metodami.

- `dodaj(ime_naloge, rok)` zabeleži, da je potrebno do podanega roka opraviti nalogo s podanim imenom.
- `opravi(ime_naloge, cas)` pokličemo, ko opravimo nalogo; pri tem podamo čas, ko smo jo opravili.
- `naslednja_naloga()` vrne ime naloge z najzgodnejšim rokom ali `None`, če ni čakajočih nalog.
- `cakajocih()` vrne število čakajočih nalog.
- `zamujenih()` vrne število nalog, ki so bile opravljene po podanem roku.

Konstruktur dodaj po želji. Argumentov ne sme pričakovati.

Prvi dve metodi naj ne vrneta ničesar. V kakšnih enotah je podan čas, te ne zanima. Nalog ne dodajamo in ne opravljamo nujno po vrsti.