Izpit iz predmeta Programiranje 1, 29. januar 2020 ob 12.00. **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. Odstrani enake

Napiši funkcijo `odstrani_enake(s, t)`, ki prejme dva enako dolga seznama. Funkcija primerja istoležne pare njunih elementov in če sta oba elementa enaka, ju pobriše iz seznamov. Funkcija ne vrača ničesar, temveč spreminja prejeta seznama. V spodnjem primeru pobriše pare 3, 2 in 4.

```
>>> a = [1, 3, 2, 13, 2, 4, 4]
>>> b = [8, 3, 13, 8, 2, 4, 5]
>>> odstrani_enake(a, b)
>>> a
[1, 2, 13, 4]
>>> b
[8, 13, 8, 5]
```

2. Diff

Napiši funkcijo `diff(datoteka1, datoteka2)`, ki prejme imeni dveh datotek in vrne seznam razlik med njima. Vrnjeni seznam naj vsebuje trojke (številka vrstice, niz z vrstico iz prve datoteke, niz z vrstico iz druge datoteke). Datoteki vsebujeta enako število vrstic. Prva vrstica naj ima številko 1. Vrsticam odstrani znake za novo vrstico.

Za datoteki v okvirčkih na desni funkcija vrne `[(3, 'Cecilija', 'Cilka'), (5, 'Ema', 'Eva Ema')]`.

Ana Berta Cecilija Dani Ema

Ana Berta Cilka Dani Eva Ema

3. Mesta

Imamo seznam, ki vsebuje imema tekmovalcev in število točk, ki so jih dosegli, npr. `[("Ana", 15), ("Berta", 13), ("Cilka", 12), ("Dani", 12), ("Ema", 8), ("Fanči", 6), ("Greta", 6)]`. Seznam je že urejen po padajočem številu točk. Točke so nenegativna cela števila.

Napiši funkcijo `mesta(s)`, ki prejme takšen seznam in vrne takšen seznam, v katerem bo i-ti element vseboval seznam vseh, ki so uvrščeni na i-to mesto. Za gornji primer vrne `[["Ana"], ["Berta"], ["Cilka", "Dani"], [], ["Ema"], ["Fanči", "Greta"], []]`. Cilka in Dani sta v istem seznamu, ker si delita tretje mesto in četrti seznam je prazen, ker nihče ni četrti. Podobno je z Fanči in Greto: sta v istem seznamu in naslednji seznam je prazen. Isto mesto si lahko deli tudi več oseb – celo vse. Za nekaj primerov poglej (tudi) teste.

4. Dovolj lihih

Napiši rekurzivno funkcijo `dovolj_lihih(s, n)`, ki vrne `True`, če je v seznamu `s` vsaj `n` lihih števil in `False`, če jih ni toliko. (Rekurzivna naj bo prav ta funkcija s prav temi argumenti. Ne piši pomožnih funkcij.)

5. Parkirišče

Napiši razred Parkirisce, obdarjen z naslednjimi metodami.

- Konstruktor prejme število parkirnih mest na parkirišču.
- `prosto()` vrne `True`, če je na parkirišču kaj prostih mest in `False`, če jih ni.
- `parkiraj(registracija, cas)` pokličemo, ko ob podanem času na parkirišče pripelje avto s podano registracijo. Če je parkirišče polno, metoda ne naredi ničesar, sicer pa zabeleži, kar je potrebno za delovanje ostalih metod. Metoda ne vrne ničesar.
- `odpelji(registracija, cas)` pokličemo, če avto s podano registracijo ob podanem času zapusti parkirišče. Metoda vrne ceno parkirnine; ta znaša toliko evrov, kolikor je začetih ur: če je avto parkiran 2.45 ure, plača 3 evre. (V modulu `math` se nahaja potencialno uporabna funkcija `ceil`.) Predpostaviti smeš, da je avto, ki je odpeljal s parkirišča, dejansko bil parkiran na parkirišču.
- `zasluzek()` vrne vsoto vseh parkirnin za avte, ki so že zapustili parkirišče.

Časi po podani v urah. Če avto pripelje ob 10 uri in 45 minut, je argument `cas` enak 10.75 (10 in tri četrt). (To je preprosto: minute naj te ne vznemirjajo, za izračun parkirnine pa čase parkiranja le zaokrožaš navzgor.)