Izpit iz predmeta Programiranje 1, 26. junij 2019 Čas pisanja: 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 z asistentom.

Nekaj potencialno koristnih metod seznamov. Seveda ne boste potrebovali vseh. Toliko sem jih napisal v izogib spoilerjem.

- `s.insert(i, x)` na `i`-to mesto (ali na konec, če je `i == len(s)`) vrine element `x`
- `s.append(x)` na konec seznama doda `x`
- `del s[i]` iz seznama odstrani `i`-ti element
- `s.pop()` vrne zadnji element seznama in ga odstrani iz seznama
- `s.pop(i)` vrne `i`-ti element seznama in ga odstrani iz njega
- `s.sort()` uredi elemente seznama
- `s.index(x)` vrne indeks elementa `x`

1. Zaporedna števila

Imamo nek naraščajoč seznam celih števil, na primer `[3, 4, 5, 8, 10, 11, 12, 13, 18, 19, 20]`. Napiši funkcijo `zaporedne_skupine(s)`, ki prejme kak tak seznam in vrne seznam seznamov zaporednih števil. V gornjem primeru bi vrnila `[[3, 4, 5], [8], [10, 11, 12, 13], [18, 19, 20]]`.

Napiši tudi funkcijo `zaporedni_intervali(s)`, ki prejme enak argument kot gornja, vendar vrne seznam intervalov, predstavljenih s parom (spodnja_meja, zgornja_meja). V gornjem primeru vrne `[(3, 5), (8, 8), (10, 13), (18, 20)]`. Pri popravljanju izpita bom cenil, če bo ta, druga funkcija napisana v eni vrstici.

2. Dodajanje v intervale

Napiši funkcijo `dodaj(e, intervali)`, ki prejme neko število `e` in seznam intervalov, kot ga vrača funkcija `zaporedni_intervali`. Funkcija `dodaj_v_intervale` ne vrne ničesar, temveč spremeni podani seznam tako, da intervali vključujejo tudi število `e`. Pri tem združi morebitne sosednje intervale, ki jih novo število "stakne".

Recimo, da so `intervali` seznam `[(3, 5), (8, 8), (10, 13), (18, 20)]`.

- `dodaj(1, intervali)` ga spremeni v `[(1, 1), (3, 5), (8, 8), (10, 13), (18, 20)]`.
- Če namesto tega dodamo 2, se spremeni v `[(2, 5), (8, 8), (10, 13), (18, 20)]`.
- Če dodamo 3, 4, ali 5, 8, 10 ..., se ne zgodi nič, ker so ta števila že vključena v nek interval.
- Če bi dodali 6, bi dobili `[(3, 6), (8, 8), (10, 13), (18, 20)]`.
- Če dodamo 9, dobimo `[(3, 5), (8, 13), (18, 20)]`: `(8, 8)` in `(10, 13)` se združita v `(8, 13)`.
- Če dodamo 16, dobimo `[(3, 5), (8, 8), (10, 13), (16, 16), (18, 220)]`.
- Če dodamo 25, dobimo `[(3, 5), (8, 8), (10, 13), (18, 20), (25, 25)]`.

Nasvet: odličen začetek funkcije je

```
for i, (od, do) in enumerate(intervali):
    if e <= do + 1:
        break
else:
    intervali.append((e, e))
    return
```

Po tem vemo, da je potrebno število dodati pred i-ti interval, na njegov začetek ali na njegov konec. Morda pa je število v njem in ni potrebno storiti ničesar. Poleg tega je i-ti interval morda potrebno združiti s prejšnjim ali naslednjim. Funkcija mora le preveriti vse možnosti (tri mesta dodajanj in dve združevanji). Da katere ne spregledaš, bodo pomagali testi.

Predpostaviti smeš, da so intervali, ki jih dobi funkcija, pravilno urejeni ter se ne stikajo ali prekrivajo.

3. Nepalindrom

Beseda ABCDEFEDCBA je palindrom, ker se nazaj bere enako kot naprej (če bi se ga dalo prebrati). ABCDEFEMCKA pa ni palindrom, ker se dva para (B in K ter D in M) ne ujemata. Rekurzivno funkcijo `palindrom(s)`, ki preveri, ali je niz palindrom, znamo pisati. Pa pokaži, da znaš napisati **rekurzivno** funkcijo `nepalindrom(s)`, ki vrne število neujemajočih se parov. Za gornjo besedo torej vrne 2.

4. Številka intervala

Napiši funkcijo `slovar_intervalov(stevila)`, ki bo omogočala tole.

```
>>> s = slovar_intervalov([3, 4, 5, 8, 10, 11, 12, 13, 18, 19, 20])
>>> s[12]
(10, 13)
>>> s[5]
(3, 5)
>>> s[8]
(8, 8)
>>> s[15]
KeyError: 15
```

Kaj vrača ta funkcija, iz tega primer odkrij sam(a).

`KeyError` je seveda resnična napaka, ki se zgodi "sama od sebe", ne pa nekaj, kar vrne ali izpiše tvoja funkcija.

5. Maček

Med sestavljanjem tega izpita me maltretira mlad maček. Mogoče je lačen. Hrane si ne lovi sam, temveč mu jo moram jaz, zato hvaležno praska in grize mene.

Napiši razred `Macek` s tremi metodami.

- `shrani_mis(teza)` pomeni, da je maček (a ne naš) ulovil miš s podano težo in jo shranil v posodo za miši.
- `pojej_mis()` pomeni, da maček vzame miš iz posode in jo poje. Ker je posoda ozka, ne izbrska miši iz dna, temveč vzame tisto, ki je na vrhu, torej zadnjo ujeto (a še nepojedeno) miš. Če metodo pokličemo, ko je posoda prazna, se ne zgodi nič.
- `sitost()` vrne `skupno` težo pojedenih miši. Miši, ki še niso pojedene, temveč le ulovljene, seveda ne prispevajo k sitosti.

Morebitne druge potrebne metode (kot so `grizi`, `praskaj`, `lulaj`, `kakaj`, `__init__`, `unicuj_zavese_kavc_in_ostalo_pohistvo`) lahko dodajaš po želji.