Računalništvo (fizika): 2. izpit

7. 5. 2018

Naloge rešujte na strežniku Tomo, `https://www.projekt-tomo.si`.

Brskanje po spletni učilnici, nalogah na strežniku Tomo, video zapiskih predavanj in *uradni* dokumentaciji za Python *je dovoljeno*. Brskanje po spletnih straneh, ki vsebujejo rešitve nalog in nasvete o tem, kako se rešuje naloge, *ni dovoljeno*. Komunikacija s komerkoli, razen z asistentom, *ni dovoljena*.

Čas reševanja je 180 minut. Doseženih 100 točk šteje za maksimalno oceno. Veliko uspeha!

1. naloga (25 točk)

Študentski referat vsako leto popiše študente. Študenti so nagajivi in med ime in priimek pogosto vstavijo še svoj vzdevek ali kako smešnico.

Podatki o študentih so predstavaljni v nizu znakov (stringu), podatki za vsakega študenta so v svoji vrstici, na primer:

```
Maja Polanec
Peter Veliki Oven
Miha kdor to bere je osel Meh
```

Sestavite funkcijo `ocisti(podatki)`, ki sprejme podatke in vrne prečiščene podatke, se pravi, da v vsaki vrstici obdrži le prvo in zadnjo besedo. Zgornji podatki v prečiščeni obliki so:

```
Maja Polanec
Peter Oven
Miha Meh
```

2. naloga (30 točk)

Pri igri Scrabble imamo na voljo nekaj črk, iz katerih lahko sestavljamo besede. Zanima nas, katere nize lahko sestavimo, če uporabimo vse črke. Na primer, če imamo črke A, A, B in C, lahko sestavimo nize AABC, AACB, ABAC, ABCA, ACAB, ACBA, BAAC, BACA, BCAA, CAAB, CABA in CBAA.

Razpoložljive črke predstavimo s slovarjem, ki vsako črko preslika v število, ki pove, kolikokrat lahko uporabimo črko. Na primer, črke A, A, B, C predstavimo s slovarjem

```
{'A' : 2, 'B' : 1, 'C' : 1}
```

Sestavite funkcijo `nizi(crke)`, ki sprejme slovar razpoložljivih črk in vrne množico vseh besed, ki jih lahko sestavimo.

3. naloga (40 točk)

V pomoč študentom bomo sestavili razred `Urnik`, v katerega lahko študent vnese svoj tedenski urnik. Urnik vsebuje dogodke za vse dni v tednu, med 8:00 in 16:00. Vsak dogodek je predstavljen z nizom, traja pa natanko eno uro. Zadnji vnos v dnevu je torej dogodek, ki se prične ob 15:00 in zaključi ob 16:00.

a) (10 točk) Razredu `Urnik` dodajte metodo `__init__(self)`, ki inicializira prazen urnik.

b) (10 točk) Razredu `Urnik` dodajte metodo `dodaj(self, dan, ura, dogodek)`, ki na koledar doda dogodek, ki ga opisuje niz `dogodek`. Argument `dan` je število med 1 in 7, (ponedeljek do nedelje), `ura` pa število med 8 in 15.

c) (10 točk) Razredu `Urnik` dodajte metodo `dogodek(self, dan, ura)`, ki vrne opis dogodka, ob podanem dnevu in uri. Če dogodka ni, naj metoda vrne `None`.

d) (10 točk) Študente zanima tudi, kdaj imajo proste večje bloke časa, torej, kdaj imajo zaporedne ure brez dogodkov. Razredu `Urnik` dodajte metodo `prosti_blok(self, dolzina)`, ki na urniku poišče prvi prosti blok dolžine vsaj `dolzina`. Vrne naj par števil `(dan, ura)`, ki opisujeta čas začetka prostega bloka. Števili `dan` in `ura` naj bosta kot v podnalogi (b) (od 1 do 7, ter od 8 do 15). Če želenega prostega bloka ni, naj metoda vrne `False`.

Opomba: Podrobnosti implementacije so prepuščene vam. Zaradi tega Tomo preverja pravilnost le delov (c) in (d).

4. naloga (25 točk)

V računalniški igrici potuje možiček po polju velikosti $n \times n$. Iz polja s koordinatama (x, y) se sme premakniti na polji s koordinatama $(x+1, y)$ in $(x, y+1)$. Na nekaterih poljih preži nevarnost, zato se jim mora izogniti. Zanima nas dolžina najkrajše varne poti med poljema $(0,0)$ in $(n-1, n-1)$.

Igralno polje predstavimo s tabelo `t` velikost $n \times n$. Če je polje (x, y) varno, je `t[x][y]` enako `True`, sicer je `False`.

Sestavite funkcijo `najkrajsa_pot(t)`, ki sprejme opis igralnega polja `t` in vrne dolžino najkrajše *varne* poti od $(0,0)$ do $(n-1, n-1)$, ali `None`, če ni poti od $(0,0)$ do $(n-1, n-1)$. Funkcija mora delovati učinkovito. Katero vrsto algoritma ste uporabili? Odgovor zapišite v komentar k rešitvi.