Izpit iz predmeta Programiranje 1, 11. junij 2013 ob 13.00. **Čas pisanja: 90 minut.**

Rešitve shranite **v eno samo datoteko s končnico .py** in jo oddajte prek Učilnice. Imena funkcij naj bodo takšna, kot jih predpisuje naloga. Rešitev preverite s testnimi primeri. Za rešitev lahko dobite določeno število točk, tudi če ne prestane testov. Rešitev, ki prestane teste, še ni nujno pravilna.

Rešitve zahtevajo po manj kot desetih vrstic; prva in tretja sta brez težav (in brez izpeljanih seznamov) rešljivi v eni vrstici. Čeprav ne zahtevamo takšne učinkovitosti, pa bomo posebej okorno napisane rešitve kaznovali z nekaj odbitimi točkami.

Dovoljena je vsa literatura na poljubnih medijih in ves material, ki je objavljen na Učilnici. Strogo prepovedana je samo komunikacija. Študenti s predolgimi vratovi in podobnimi hibami bodo morali (najmanj!) zapustiti izpit, katerega opravljanje se bo štelo kot neuspešno.

1. Samo enkrat

Napiši funkcijo `samo_enkrat(s)`, ki pove, ali se v podanem nizu `s` noben znak ne pojavi več kot enkrat.

Primeri

```
>>> samo_enkrat("abcd")
True
>>> samo_enkrat("abcb)
False
>>> samo_enkrat("")
True
```

2. Veliko, a ne več kot

Napiši funkcijo `naj_pod(s, n)`, ki kot argument dobi seznam števil `s` in vrne podseznam z največjo vsoto, ki je še manjša ali enaka `n`. Če je seznamov z enako največjo vsoto več, naj vrne prvega med njimi.

Namig: z zanko pojdi prek vseh možnih koncev in znotraj nje z zanko prek vseh možnih začetkov (pred tem koncem). Za vsak takšen podseznam preveri, ali ima vsoto, ki je večja od največje doslej, a manjša ali enaka `n`, in si, če je tako, zapomni vsoto, začetek in konec.

Primeri

```
>>> naj_pod([2, 1, 5, 6, 11, 2, 3, 6], 16)
[11, 2, 3]
>>> naj_pod([2, 1, 5, 6, 11, 2, 3, 6], 3)
[2, 1]
>>> naj_pod([2, 1, 5, 6, 11, 2, 3, 6], 28)
[1, 5, 6, 11, 2, 3]
```

3. Palindrom

Niz je palindrom, če se naprej bere enako kot nazaj (predpostavimo, da nima presledkov). Napiši **rekurzivno** funkcijo `palindrom(s)`, ki ugotovi, ali je podani niz palindrom. Deluje naj tako: niz je palindrom, če je krajši od dveh znakov ali pa sta prvi in zadnji znak enaka, tisto, kar je med njima, pa je palindrom.

Primeri

```
>>> palindrom("abcba")
True
>>> palindrom("abccba")
True
>>> palindrom("abcda")
False
```

4. Slepa polja

V neki igri je mogoče z vsakega polja priti na določena druga polja. Možne poteze opišemo s slovarjem, ki kot ključe vsebuje imena polj, kot vrednosti pa množice polj, na katere je mogoče priti s posameznega polja. Tako

```
poti = {"a": {"b", "c"}, "b": {"a", "c", "e", "f"}, "c": {"d", "e"}, "e": {"a"}}
```

pomeni, da s polja `a` pridemo na `b` in `c`; s polja `b` na `a`, `c`, `e` in `f`; s polja `c` na `d` in `e`; s polja `e` na `a`. Nekatera polja so "slepa": v gornjem primeru sta to polji `d` in `f`, s katerih ni mogoče priti na nobeno drugo polje. Slepa polja lahko prepoznamo po tem, da se ne pojavljajo kot ključi (v slovarju ne bomo imeli nikoli praznih množic).

Napiši funkcijo `slepa_polja(s)`, ki dobi slovar poti (poljuben, ne nujno takšen, kot je gornji) in vrne množico slepih polj.

Primeri

```
>>> slepa_polja(poti)
{"d", "f"}
>>> slepa_polja({"a": {"b", "c"}, "c": {"b"}})
{"b"}
>>> slepa_polja({"a": {"b", "c"}, "c": {"b"}, "b": {"c"}})
set()
```

5. Skupni prijatelji

V datoteki s testnimi primeri je del rešitve domače naloge Facebook. Dodaj metodo `naj_podobnik(a)`, ki za podano osebo `a` vrne tisto osebo, ki ima z `a` največ skupnih prijateljev. Kadar je možnih več odgovorov, vrne poljubnega med njimi.

Namig: uporabi obstoječo metodo `self.skupni_prijatelji(a, b)`.

Primeri

```
>>> mreza.naj_podobnik("Francka")
"Helga"
```