Izpit iz predmeta Programiranje 1, 23. januar 2011 ob 12.30.

Vse rešitve shranite **v eno samo datoteko s končnico .py** in jo oddajte prek Učilnice. Vse funkcije naj imajo enaka imena, kot jih predpisuje naloga. **Pozorno preberite** naloge in ne rešujte le na podlagi primerov!

Da rešitev ne bi imela trivialnih napak, jo preverite s testi v ločeni datoteki na Učilnici. Za rešitev naloge lahko dobite določeno število točk, tudi če ne prestane vseh testov. Funkcija, ki prestane vse teste, še ni nujno pravilna.

Pri reševanju nalog je dovoljena vsa literatura na poljubnih medijih, ves material, ki je objavljen na Učilnici, vključno z objavljenimi programi; njihova uporaba in predelava se ne šteje za prepisovanje.

Izpit morate pisati na fakultetnih računalnikih, ne lastnih prenosnikih.

Študenti s predolgimi vratovi in podobnimi hibami bodo morali zapustili izpit, katerega opravljanje se bo štelo kot neuspešno. Hujše kršitve bomo prijavili disciplinski komisiji.

Čas pisanja: 90 minut.

1. Trdnjava

Figuro prestavljamo po neskončni šahovnici. Premik opišemo z nizom, dolgim dva znaka; prvi je črka U, D, L ali R, ki predstavlja smer (up, down, left in right) in drugi je številka med 0 in 9, ki pove, za koliko polj prestavimo figuro v podani smeri. Začetna pozicija je (0, 0) - uporabljali bomo kar takšno notacijo, ne "šahovske".

Napiši funkcijo `trdnjava(s)`, ki kot argument dobi seznam potez in vrne `True`, če se trdnjava kadarkoli med premikanjem (ne nujno na koncu!) ponovno znajde na izhodiščnem polju (0, 0) in `False`, če se ne.

Primer

```
>>> trdnjava(['U5', 'D5'])
True
>>> trdnjava(['U5', 'D6']) # trdnjavo smo premaknili PREK izhodiščnega polja, ne nanj!
False
>>> trdnjava(['R5', 'L5', 'R3'])
True
>>> trdnjava(['R5', 'U5', 'L5', 'D5', 'R5', 'U5'])
True
>>> trdnjava(['R5', 'U4', 'L3'])
False
>>> trdnjava(['L3', 'U2', 'D2', 'R1', 'R1', 'R1', 'R1', 'R1', 'R1', 'R1'])
True
>>> trdnjava(['L3', 'U2', 'D2', 'R1', 'R1'])
False
>>> trdnjava(['U0'])
True
>>> trdnjava(['U0', 'U1'])
True
```

2. Primerjanje seznamov

Napiši funkcijo `primerjaj(s, t)`, ki primerja dva (neprazna) seznama, takole:

- če sta seznama enako dolga in so vsi elementi `s` manjši od istoležnih elementov `t`, naj vrne `-1`,
- če sta seznama enako dolga in so vsi elementi `s` večji od istoležnih elementov `t`, naj vrne `1`,
- sicer vrne `0`.

Primer

```
>>> primerjaj([1, 2, 3, 4], [2, 3, 4, 5])
-1
>>> primerjaj([2, 3, 4, 5], [1, 2, 0, 0])
1
>>> primerjaj([1, 2, 3], [4, 5, 6, 7])
0
>>> primerjaj([1, 0], [0, 1])
0
```

3. Natakar

Ko je prišel natakar, so naročile:

- Ana je naročila torto,
- Berta je naročila krof,
- Cilka je naročila kavo,
- Ana je naročila še kavo,
- Berta je rekla, da ne bo krofa,
- Cilka je rekla, da ne bo torte (no, saj je niti ni naročila; to natakar mirno ignorira),
- Berta je naročila torto.

Vse skupaj zapišemo takole: `[("Ana", "torta"), ("Berta", "krof"), ("Cilka", "kava"), ("Ana", "kava"), ("Berta", "-krof"), ("Cilka", "-torta"), ("Berta", "torta")]`. Seznam torej vsebuje pare nizov `(oseba, jed)`, pri čemer se jed včasih začne z `"-"`, kar pomeni, da stranka prekliče naročilo te jedi oz. pijače.

Napiši funkcijo `narocila(s)`, ki prejme takšen seznam in vrne slovar, katerega ključi so imena strank, vrednost pri vsakem ključu pa je seznam vsega, kar mora stranka na koncu dobiti.

Namig: če uporabljaš `defaultdict`, dobiš iz njega seznam tako, da pokličeš funkcijo `dict`. Če je, na primer, `po_strankah` objekt tipa `defaultdict`, bo `dict(po_strankah)` običajen slovar.

Primer

```
>>> narocila([('Ana', 'torta'), ('Berta', 'krof'), ('Cilka', 'kava'), ('Ana', 'kava'),
('Berta', '-krof'), ('Cilka', '-torta'), ('Berta', 'torta')])
{'Cilka': ['kava'], 'Berta': ['torta'], 'Ana': ['torta', 'kava']}
>>> narocila([('Ana', 'torta'), ('Ana', '-torta')])
{'Ana': []}
>>> narocila([('Ana', '-torta')])
{'Ana': []}  # Tu sme funkcija vrniti tudi prazen slovar, {}
```

4. Editor

Napišite funkcijo `crka(c, a)`, ki kot argument prejme dva niza dolžine 1. Prvi niz predstavlja črko in drugi "akcijo". Funkcija naj vrne naslednje:

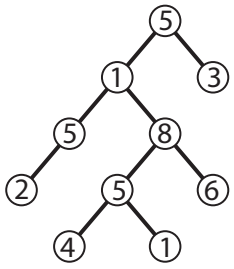
- če je akcija enaka `"U"`, vrne veliko črko `c`: `crka("t", "U")` vrne `"T"`;
- če je akcija števka med `"0"` in `"9"`, vrne toliko ponovitev črke `c`: `crka("t", "3")` vrne `"ttt"`;
- če je akcija enaka `"x"`, vrne prazen niz: `crka("t", "x")` vrne prazen niz;
- če je akcija enaka `"."`, vrne nespremenjeno črko `c`: `crka("t", ".")` vrne `"t"`.

Poleg tega napišite funkcijo `kodiraj(beseda, koda)`, ki prejme besedo in akcije za posamezne črke, ter vrne novo besedo, v kateri je vsaka črka predelana tako, kot zahteva ustrezna črka iz akcije. Predpostaviti smete, da sta niza beseda in koda enako dolga.

Primer

```
>>> kodiraj("beseda", ".U5.x.")
bEssssea
```

5. Največji



Spomnimo se potomcev Karla Velikega, ki smo jih zložili v drevo, katerega koren je bil Karel, pod njim pa so bili njegovi potomci. V tej nalogi bomo imeli podobno drevo, v njem pa ne bodo imena temveč pozitivna cela števila. Vsako število bo imelo največ dva potomca; imenovali ju bomo levi in desni potomec.

Vozlišča drevesa predstavlja razred `Stevilka` (uporabite definicijo, ki je v testnih primerih). Ta ima metodo `vrednost()`, ki vrne število v vozlišču, metodo `levi()`, ki vrne levega potomca in metodo `desni()`, ki vrne desnega potomca. Če levega oz. desnega potomca ni, metoda vrne `None`.

Iz razreda `Stevilka` izpeljite razred `StevilkaPlus`, ki ima poleg podedovanih metod še metodo `najvecje`, ki vrne največje število v drevesu. Na primeru s slike mora metoda vrniti 8.