Izpit iz predmeta Programiranje 1, 2. september 2010 ob 10.00 **Čas pisanja: 90 minut**

Pri reševanju nalog je dovoljena domiselnost. Bližnjice do pravilne rešitve nalog so vredne pohvale. Dovoljena je raba standardnih Pythonovih knjižnic (`os`, `random`, `math`...), prepovedane pa so dodatne knjižnice, kot so PIL, numpy, PyQt... Ali je neka knjižnica standardna knjižnica ali ne, boste najlažje prepoznali po tem, ali je opisana v Pythonovi dokumentaciji.

Funkcije naj imajo takšna imena, kot jih predpisuje naloga. V rešitvah naj bo označeno, kateri del kode predstavlja rešitev katere naloge**. Če naloga zahteva, naj funkcija *vrne rezultat*, mora *vrniti rezultat*, ne pa *izpisovati*. Funkcija mora podatke dobiti prek argumentov, ne prek zunanjih (npr. globalnih) spremenljivk.**

Vse naloge so vredne enako število točk.

Rešitve vseh nalog shranite v eno samo datoteko .py in jo oddajte prek tega strežnika na enak način, kot ste oddajali domače naloge. Uporabniki prenosnikov bodo rešitve oddajali na USB ključih.

Pri reševanju nalog je dovoljena vsa literatura na poljubnih medijih. Prepovedana pa je uporaba interneta, z izjemo učilnice (različice, do katere imate dostop s teh računalnikov) in vsi drugi načini komunikacije. Na računalniku mora biti izključena brezžična povezava in bluetooth ter ugasnjen Skype in drugi programi ter priprave, ki jih je mogoče uporabiti za komuniciranje. Kršilci teh pravil bodo zapustili izpit, katerega opravljanje se bo štelo kot neuspešno. Hujše kršitve bomo prijavili disciplinski komisiji za študente.

1. Ujeme

Napiši funkcijo **ujeme**, ki kot vhod prejme dve besedi in vrne novo besedo, ki vsebuje tiste črke, v katerih se besedi ujemata, črke na ostalih mestih pa so zamenjane s pikami. Če besedi nista enako dolgi, naj bo nova beseda toliko dolga, kolikor je dolga krajša izmed podanih besed.

Primer

```
>>> ujeme("ROKA", "REKE")
'R.K.'
>>> ujeme("ROKA", "ROKAV")
'ROKA'
>>> ujeme("CELINKE", "POLOVINKE")
'..L....'
```

Namig. Če znaš uporabiti zip, bo še lažje, kot brez njega.

2. Grep

Napiši funkcijo **grep**, ki kot argument dobi opis imena datoteke (npr. *.txt ali *.* ali c:\d\fakulteta\m*.py) in podniz, kot rezultat pa vrne seznam datotek, ki vsebujejo ta podniz.

Primer

```
>>> grep("c:\\d\\risanje\\*.py", "zelva")
['c:\\d\\risanje\\koch.py', 'c:\\d\\risanje\\neobjektno.py',
'c:\\d\\risanje\\objektno.py', 'c:\\d\\risanje\\tavajoci.py',
'c:\\d\\risanje\\tavajoci2.py', 'c:\\d\\risanje\\tavajoci3.py',
'c:\\d\\risanje\\tavajoci4.py']
```

Seznam torej vsebuje imena tistih vseh datotek v direktoriju c:\d\risanje in s končnico *.py, ki vsebujejo besedo zelva.

Namig. Ne bo težko. Na pomoč ti priskoči funkcija glob, ki jo najdeš v modulu glob. Kot argument ji podaš ime datoteke (z zvezdicami) in vrne seznam vseh datotek, ki ustrezajo vzorcu.

>>> glob.glob("c:\\d\\risanje*.py")

['c:\\d\\risanje\\besede.py', 'c:\\d\\risanje\\crte.py', 'c:\\d\\risanje\\koch.py', 'c:\\d\\risanje\\sierpinski.py']

Odtod je naloga lahka: prebereš datoteke in pogledaš, ali je v njih, kar iščeš.

3. Bralca bratca

Brata Peter in Pavel morata za domače branje prebrati vse knjige na knjižni polici. Nepridiprava sta se odločila, da jih bo vsak prebral približno pol (po številu strani), o drugi polovici pa se bo pustil poučiti od brata: Peter bo bral knjige z leve strani police, Pavel z desne. Sestavi funkcijo **razdeliKnjige**, ki kot argument dobi seznam z debelinami knjig in vrne par (terko), ki pove, koliko knjig naj prebere eden in koliko drugi.

Primer

```
>>> razdeliKnjige([500, 100, 100, 100, 900])
(4, 1)
>>> razdeliKnjige([500, 100, 100, 100, 900, 100])
(4, 2)
>>> razdeliKnjige([50, 86, 250, 13, 205, 85])
(3, 3)
>>> razdeliKnjige([50, 86, 150, 13, 205, 85])
(4, 2)
>>> razdeliKnjige([5, 5])
(1, 1)
>>> razdeliKnjige([5])
(0, 1)
>>> razdeliKnjige([])
(0, 0)
```

Prvi primer razkriva namen vaje: Peter prebere štiri knjige, ki imajo skupaj 800 strani, Pavla pa doleti Dostojevski. V drugem primeru dobi Pavel še eno knjigo; čeprav bi bilo pravičneje, če bi jo Peter (in bi imela vsak 900 strani), smo pač dogovorjeni, da eden bere z leve drugi z desne...

Namig: ne ukvarjaj se s Pavlom, temveč le s Petrom: koliko knjig mora prebrati, da jih bo približno pol? Pri "srednji knjigi" preveri, ali bomo bližje polovici, če jo prebere (in morda preseže polovico) ali jo pusti Pavlu (in prebere malo manj kot pol).

4. Ploščine

Ploščina poligona, katerega oglišča so podana s koordinatami (x_1, y_1), (x_2, y_2), … (x_n, y_n), izračunamo po formuli

$$p = \frac{1}{2}\left|\left(x_1 y_2 - x_2 y_1\right) + \left(x_2 y_3 - x_3 y_2\right) + \ldots + \left(x_{n-1} y_n - x_n y_{n-1}\right) + \left(x_n y_1 - x_1 y_n\right)\right|$$

Napiši funkcijo **ploscina**, ki kot argument prejme seznam oglišč poligona in kot rezultat vrne njegovo ploščino. Pazi na zadnji člen! (Iznajdljivost pa je dobrodošla!)

Primer

```
>>> ploscina([(0, 0), (1, 0), (1, 1), (0, 1)])
1.0
>>> ploscina([(0, 0), (1, 0), (1, 1)])
0.5
>>> ploscina([(0, 0), (1, 0), (2, .5), (1, 1), (1, 2), (.5, 1), (0, 2)])
2.0
```