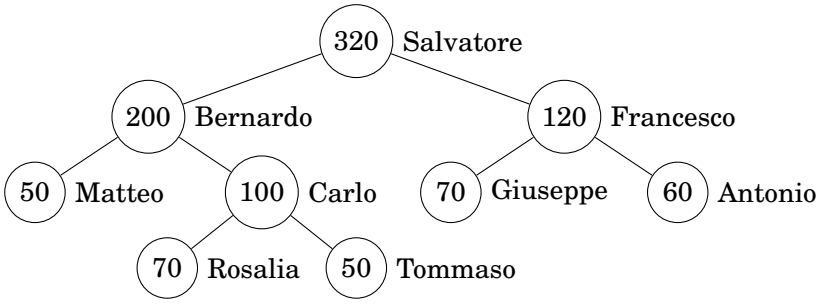
Programiranje I: 1. izpit

3. svečan 2014

Čas reševanja je 150 minut. Doseženih 100 točk šteje za maksimalno oceno. Veliko uspeha!

1. naloga (Mafija, 10 + 10 + 10 točk)

V neki mafijski organizaciji so člani urejeni hierarhično. Vsakdo razen botra (vrhovnega šefa) ima natanko enega nadrejenega. Vsak mafijec ima lahko pod seboj največ dva podrejena (levega in desnega). Primer takšne mafijske organizacije:



Mafijci morajo zbirati denar s kriminalnimi dejavnostmi. Tisti, ki imajo podrejene, pa ga poleg tega poberejo še od svojih podrejenih. Ves "prisluženi" in prejeti denar morajo oddati svojemu nadrejenemu.

Boter je posumil, da nekateri člani goljufajo. Nekaj denarja, ki ga poberejo od podrejenih, zadržijo zase. Od vsakega člana je pridobil podatek o tem, koliko denarja je oddal naprej. Podatke je shranil v podatkovno strukturo `Drevo`, ki je že implementirana.

a) (10 točk) Botra zanima, koliko denarja zaradi goljufov "ponikne". Želi, da sestavite funkcijo `naloga1a(self)`, ki vrne skupno vsoto denarja, ki ponikne. Primer (če `d` ustreza zgornji sliki):

```
>>> d.naloga1a()
30
```

b) (10 točk) Ko je boter dognal, koliko denarja ponikne, je totalno po[redacted]. Pri priči hoče imeti imena vseh goljufov! Napišite funkcijo `naloga1b(self)`, ki vrne množico goljufov. Vsak goljuf naj bo predstavljen z naborom. Prva kompotenta naj bo ime goljufa, druga komponenta pa količina denarja, ki ga je utajil. Primer:

```
>>> d.naloga1b()
{('Carlo', 20), ('Francesco', 10)}
```

c) (10 točk) Botru se dozdeva, da so najbolj pridne *majhne ribe*. To so tisti mafijci, ki nimajo pod seboj nobenega podrejenega. Tistim, ki imajo podrejene, se reče *velike ribe*. Napišite funkcijo `naloga1c(self)`, ki vrne par (nabor) dveh števili, pri čemer je:

- 1. število skupna vsota denarja, ki ga zaslužijo majhne ribe;
- 2. število skupna vsota denarja, ki ga zaslužijo velike ribe (brez pobirkov od podrejenih).

Primer:

```
>>> d.naloga1c()
(300, 50)
```

2. naloga (Poravnava besedila, 20 + 15 točk)

Besedilo je zaporedje znakov, ki so zapisani v neki datoteki. *Prazni znaki* so: presledek, tabulator in skok v novo vrstico (`' '`, `'\t'` in `'\n'`). *Beseda* pa bo za nas maksimalni strnjeni podniz, ki ne vsebuje praznih znakov. V besedilu

```
Na␣to␣se␣cesar␣začudi␣in␣pravi:␣"Ako␣so␣brusi,␣pokaj␣so␣pa␣v␣vrečah?"
```

podniz `"Ako` po naši definiciji je beseda, `Ako` pa ne (saj ni maksimalen).

a) (20 točk) Sestavite funkcijo `naloga2a(vhodna, izhodna, w)`, ki besedilo iz datoteke z imenom `vhodna` prepiše na datoteko z imenom `izhodna`, pri tem pa naj odstrani odvečne prazne znake. Velja naj naslednje:

- vsaka vrstica naj bo kar se da dolga;
- nobena vrstica naj ne vsebuje več kot `w` znakov (pri čemer znaka `'\n'` na koncu vrstice ne štejemo);
- besede znotraj iste vrstice naj bodo ločene s po enim presledkom (ne glede na to s katerimi in koliko praznimi znaki so ločene v originalnem besedilu).

Predpostavite, da dolžina nobene besede ni več kot `w`.

Primer: Če je na datoteki `krpan.txt` besedilo

```
"I␣kaj
pa␣nosiš␣v
␣␣tovoru?"␣cesar␣dalje␣vpraša.

Krpan␣se␣naglo␣izmisli␣in␣reče:␣"I␣kaj?␣Kresilno␣gobo␣pa␣nekaj␣brusov
sem␣naložil,␣gospod!"
```

naj bo po klicu funkcije

```
>>> naloga2a('krpan.txt', 'lepo.txt', 30)
```

v datoteki `lepo.txt` besedilo

```
"I␣kaj␣pa␣nosiš␣v␣tovoru?"
cesar␣dalje␣vpraša.␣Krpan␣se
naglo␣izmisli␣in␣reče:␣"I␣kaj?
Kresilno␣gobo␣pa␣nekaj␣brusov
sem␣naložil,␣gospod!"
```

b) (15 točk) Sestavite funkcijo `naloga2b(vhodna, izhodna, w)`, ki besedilo oblikuje tako, da bo še obojestransko poravnano. Obojestransko poravnavo dosežemo s tem, da med besede znotraj vrstice postavimo več presledkov. Presledki naj bodo razporejeni tako, da je razlika med dolžino največje in najmanjše luknje[1] znotraj iste vrstice kvečjemu 1, ta večje luknje pa naj bodo vse na desni strani vrstice.

Primer: Po klicu funkcije

```
>>> naloga2b('krpan.txt', 'obojestransko.txt', 30)
```

naj bo v datoteki `obojestransko.txt` besedilo

```
"I␣kaj␣␣pa␣␣nosiš␣␣v␣␣tovoru?"
cesar␣dalje␣vpraša.␣␣Krpan␣␣se
naglo␣izmisli␣in␣reče:␣"I␣kaj?
Kresilno␣gobo␣pa␣nekaj␣␣brusov
sem␣␣␣␣␣naložil,␣␣␣␣␣␣gospod!"
```

[1] Za tiste, ki niste uganili: *luknja* pomeni maksimalno strnjeno zaporedje enega ali več presledkov.

3. naloga (Izštevanka, 25 točk)

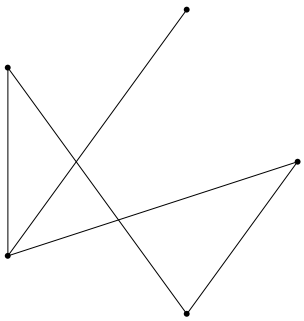
> Cinca Binca, v luknji miš,
> če jo ujameš, ne loviš.
> Kdor pa miške ne ulovi,
> ta za kazen naj miži.
> En, dva, tri,
> en, dva, tri,
> zdaj mižiš – ti!

V *Mathematici* sestavite funkcijo `naloga3[l_, k_]`, ki "simulira" izštevanko. Predstavljajmo si, da je seznam `l` cikličen (udeleženci stojijo v krogu). Šteti začnemo pri prvem elementu in preštejemo do `k`. Tistega, pri katerem smo se ustavili, izločimo. Nato ponovno štejemo do `k`. S štetjem začnemo pri tistem elementu, ki je desno od pravkar izločenega. Elementov, ki smo jih že izločili, pri štetju ne upoštevamo več. Izštevanko se gremo toliko časa, dokler ne izločimo vseh elementov. Funkcija naj vrne seznam elementov v takem vrstnem redu, kot smo jih izločali pri izštevanki. Primer:

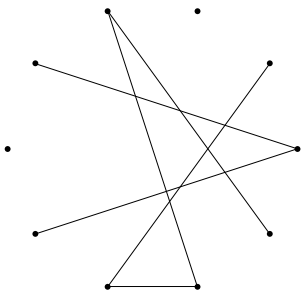
```
In[1]:= naloga3[{"Jani", "Tone", "Cilka", "Boris", "Franci", "Ana", "Pero"}, 5]
Out[1]= {"Franci", "Cilka", "Tone", "Boris", "Pero", "Jani", "Ana"}

In[2]:= naloga3[{"Jaka", "Tomaž", "Barbara", "Gašper", "Petra"}, 1337]
Out[2]= {"Tomaž", "Barbara", "Petra", "Jaka", "Gašper"}
```

4. naloga (Naključni grafi, 30 točk)

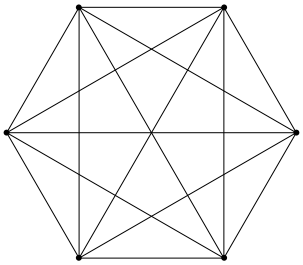
V *Mathematici* sestavite funkcijo `naloga4[n_, p_]`, ki izriše sliko naključnega grafa z `n` vozlišči. Naključni graf nastane tako, da za vsak par vozlišč med njima z verjetnostjo `p` potegnemo povezavo. Vozlišča naj bodo ekvidistančno razporejena po enotski krožnici.



`naloga4[5, 0.5]`



`naloga4[10, 0.1]`



`naloga4[6, 1.0]`