Programiranje I: 1. izpit

4. februar 2013

Čas reševanja je 120 minut. Veliko uspeha!

1. naloga (25 točk)

a) Sestavite funkcijo `naloga1a(t)`, ki v *urejeni* tabeli `t` velikosti n, ki vsebuje vsa števila od 0 do n razen enega, poišče manjkajoči element.

Časovna zahtevnost funkcije naj bo $O(\log n)$.

```
>>> naloga1a([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13])
12
>>> naloga1a([0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13])
7
>>> naloga1a([0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
2
```

b) Sestavite funkcijo `naloga1b(t)`, ki v *neurejeni* tabeli `t` velikosti n, ki vsebuje vsa števila od 0 do n razen enega, poišče manjkajoči element.

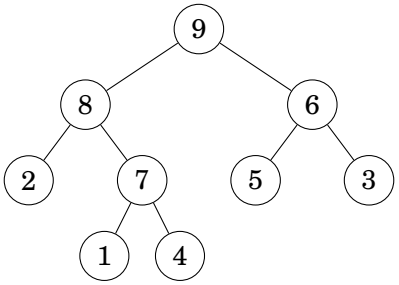
Časovna zahtevnost funkcije naj bo $O(n)$, funkcija pa naj pri svojem delu ne porabi več kot $O(1)$ dodatnega prostora.

```
>>> naloga1b([6, 3, 2, 11, 10, 0, 8, 4, 1, 5, 9, 13, 7])
12
>>> naloga1b([11, 6, 5, 1, 8, 4, 2, 13, 10, 12, 3, 9, 0])
7
>>> naloga1b([12, 10, 9, 13, 6, 8, 3, 4, 0, 5, 1, 11, 7])
2
```

Pri obeh podnalogah lahko predpostavite, da funkcija za vhod dobi tabelo ustrezne oblike.

2. naloga (25 točk)

Pravimo, da je drevo *sebično*, kadar je vrednost v vsakem korenu večja od vseh vrednosti pod njim. Primer sebičnega drevesa:



a) Razredu `Drevo` dodajte metodo `naloga2a(self)`, ki vrne `True`, kadar je dano drevo sebično, in `False`, kadar ni.

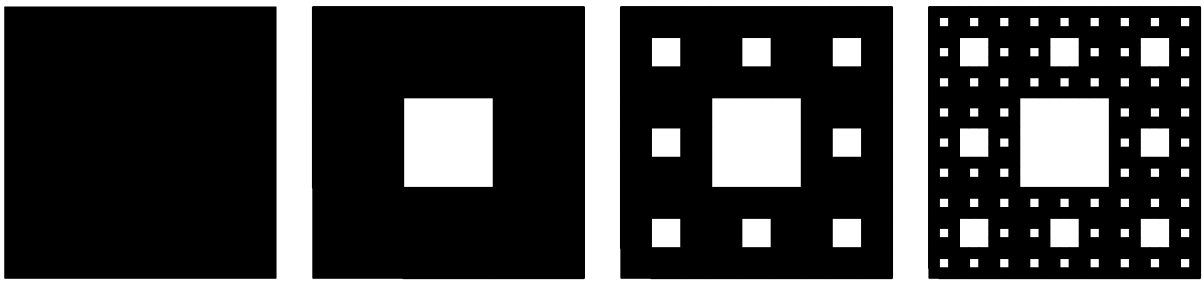
Časovna zahtevnost metode naj bo $O(n)$, kjer je n število vrednosti v drevesu.

b) Razredu `Drevo` dodajte metodo `naloga2b(self,x)`, ki v dano sebično drevo doda vrednost `x` tako, da je razširjeno drevo še vedno sebično. Pri tem lahko elemente drevesa poljubno preurejate, pazite le, da iz drevesa ne pobrišete nobenega elementa in da ne dodate nobenega drugega elementa poleg `x`.

Časovna zahtevnost metode naj bo $O(\log d)$, kjer je d globina drevesa.

3. naloga (25 točk)

V *Mathematici* sestavite funkcijo `naloga3[n_]`, ki izriše sledeče fraktale:



`naloga3[0]` `naloga3[1]` `naloga3[2]` `naloga3[3]`

4. naloga (25 točk)

a) Sestavite funkcijo `naloga4a[f_, sezzz_]`, ki na "listih" (vseh kosih, ki niso seznami) gnezdenega seznama `sezzz` uporabi funkcijo `f`. Na primer:

```
In[1]:= naloga4a[f, {1, {2, 3, 4}, 5}]
Out[1]= {f[1], {f[2], f[3], f[4]}, f[5]}
In[2]:= naloga4a[f, {1, {2, {3, 4}, 5}, {6, 7, {}, 8}, 9}]
Out[2]= {f[1], {f[2], {f[3], f[4]}, f[5]}, {f[6], f[7], {}, f[8]}, f[9]}
In[3]:= naloga4a[f, {1, {2, {1, 2}, 1}, 2}]
Out[3]= {f[1], {f[2], {f[1], f[2]}, f[1]}, f[2]}
```

b) Sestavite funkcijo `naloga4b[sezzz_]`, ki liste gnezdenega seznama `sezzz` krožno prestavi za eno mesto v levo.

```
In[1]:= naloga4b[{1, {2, 3, 4}, 5}]
Out[1]= {2, {3, 4, 5}, 1}
In[2]:= naloga4b[{1, {2, {3, 4}, 5}, {6, 7, {}, 8}, 9}]
Out[2]= {2, {3, {4, 5}, 6}, {7, 8, {}, 9}, 1}
In[3]:= naloga4b[{1, {2, {1, 2}, 1}, 2}]
Out[3]= {2, {1, {2, 1}, 2}, 1}
```