Algoritmi in podatkovne strukture 1, PISNI IZPIT, 26.1.2015

Od literature je dovoljen samo lastnoročno z navadnim svinčnikom napisan list A4 in podpisan s kemičnim svinčnikom, ki ga je potrebno oddati skupaj z izpitom. Naloge so enakovredne. Čas pisanja 90 min. Komentirajte programe! Ustni izpit za tiste, ki želijo popravljati oceno: v torek, 3.2.ob 10h v kabinetu prof. Kononenka (R2.07).

1. Dana je podatkovna struktura izraznega drevesa (operator je lahko PLUS, MINUS,TIMES ali DIVIDE):

```
public class ArithmeticExprNode {
  int operator ;
  double value ;
  ArithmeticExprNode left, right ;
}
```

   (a) Sestavi algoritem, ki pri danem korenu drevesa izpiše aritmetični izraz. Pri tem bodi pozoren, da dodaš oklepaje tam, kjer so potrebni.

   (b) Izberi ustrezne parametre in oceni časovno zahtevost svojega algoritma.

2. Za vsakega od naslednjih algoritmov argumentirano definiraj obliko grafa, tako da bo časovna zahtevnost izvajanja enaka podani (n je število vozlišč in m število povezav):

   (a) Primov algoritem: $O(m)$

   (b) algoritem Dijkstra: $O(n)$

   (c) Kruskalov algoritem: $O(n)$

3. Za dani program so bili izmerjeni naslednji časi izvajanja za različne velikosti vhodnih podatkov:

velikost	10	15	20	25
čas	50	288	750	1512

   Katera funkcija najbolj ustreza zahtevnosti tega programa v odvisnosti od velikosti vhodnih podatkov (odgovor argumentiraj z oceno konstant v enačbi T(n)= a f(n)+c): $O(logn)$, $O(n)$, $O(nlogn)$, $O(n^2)$ ali $O(n^3)$?

4. Na začetku je drevo prazno. Za dano zaporedje vstavljanja in brisanja elementov:
   vstavi: 80, 33, 84, 70, 18, 41, 61; briši: 18, 33
   Nariši 2 drevesi: eno po končanem vstavljanju in eno po končanem brisanju, če operacije izvajaš na
   a) kopici; b) AVL drevesu; c) B-drevesu reda 3

5. Dokaži a) parcialno in b) totalno pravilnost funkcije, ki preveri, če se dano naravno število y nahaja v polju naravnih števil $x[i],\ i = 1..max$.

```
boolean z(int x[], int y) {
// zacetni pogoj:  max > 1 & y pripada N & za i=1..max: x[i] pripada N
//  ciljni pogoj:  z & obstaja 1<=i<=max : x[i]=y ali
//                 not z & za vsak 1<=i<=max: x[i]!=y
  int min = 1;
  while (min <= max && !(x[min] == y))
      min++ ;
  if (!(min <= max)) z = false ; else z = true ;
}
```