Polsemestrálny test
teoretická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

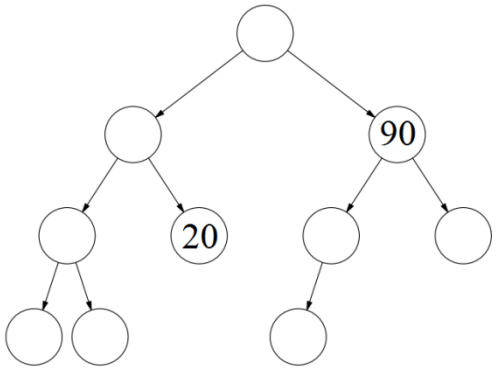
Píšte prosím čitateľne!

Hodnotenie, vyplní opravujúci:

Meno a priezvisko:	Skupina PAZ:	

1/1	2/1	3/2	4/1	5/1	6/2	7/1	8/2	9/2	10/1.5	11/8	∑/22.5

1. (1b) V binárnom vyhľadávacom strome vpravo označte všetky uzly, v ktorých sa môže nachádzať hodnota 15.



2. (1b) Zakrúžkovaním vyberte tie hodnoty, ktoré môžu byť maximálnymi hodnotami uloženými v binárnom vyhľadávacom strome vpravo?

   10 30 45 80 90 91 92 100 130

3. (2b) Predpokladajme, že rad q obsahuje hodnoty:

   9, 6, 21, 5, 10, 14, 3, 4, 6, 13, 2, 4, 6, 14

   Aký obsah bude mať rad q po volaní spracuj(q)?

```
public void spracuj(Queue<Integer> q) {
      int i1, i2;
      q.offer(-1);
      i1 = q.poll();
      while (i1 > 0) {
            i2 = q.poll();
            if (i2 > i1)
                  q.offer(i2);
            else {
                  q.offer(i1);
                  i1 = i2;
            }
      }
}
```

4. (1b) Nech pole [25, 17, 15, 13, 12, 10, 9, 6] reprezentuje binárny strom, ktorý je haldou. Nech k je výška daného stromu. Aký minimálny počet uzlov je potrebné pridať do stromu tak, aby mal výšku k + 1 (resp. aby sa jeho výška zväčšila o 1) a zároveň, aby ešte stále ostal haldou?

5. (1b) Aký je maximálny počet uzlov stromu s výškou k? Ak sa strom skladá len z koreňa, ktorý je zároveň listom, povieme, že má výšku 0.

6. (2b) Doplňte na vyznačené miesto v metóde vhodné príkazy tak, aby výsledná metóda realizovala upravenie podhaldy, ktorej koreň je na indexe *idxKorena* a posledný uzol na indexe *idxPosledneho,* v prípade, že v koreni podhaldy (*idxKorena)* sa nachádza hodnota narúšajúca vlastnosť byť haldou. Táto metóda robí presne to, čo metóda *halding* z prednášky o HeapSorte, len je napísaná iným spôsobom.

```
public static void uhalduj(int[] p, int idxKorena, int idxPosledneho) {
  int aktualny = idxKorena;

  while (true) {
    int najIdx = aktualny;
      if ((2*aktualny+1<=idxPosledneho) && (p[najIdx]<p[2*aktualny+1]))
        najIdx = 2*aktualny + 1;

      if ((2*aktualny+2<=idxPosledneho) && (p[najIdx] < p[2*aktualny+2]))
        najIdx = 2*aktualny + 2;

      if (_____________________________________________________________)
        break;

      int tmp = p[aktualny];
      p[aktualny] = p[najIdx];
      p[najIdx] = tmp;

      aktualny = najIdx;
  }
}
```

7. (1b) Nájdite a opravte chybu v metóde `maximum` triedy `SpajanyZoznam`, ktorá vráti maximálnu hodnotu uloženú v neprázdnom spájanom zozname.

```
public int maximum() {
      int vysledok = Integer.MIN_VALUE;
      Uzol aktualny = prvy;
      while (aktualny != null) {
            aktualny = aktualny.dalsi;
            vysledok = Math.max(vysledok, aktualny.hodnota);
      }

      return vysledok;
}
```

8. (2b) Uvažujme nasledovný kód, ktorý spracúva pole `p` veľkosti n. O metóde *spracujZaciatok* vieme, že nejako v lineárnom čase (vzhľadom na k) spracúva prvých k prvkov poľa `p`, t.j., volanie *spracujZaciatok*(`p, k`) má časovú zložitosť $\Theta(k)$. Aká bude časová zložitosť celého fragmentu kódu vzhľadom na dĺžku poľa `p` (označenú ako n)? Odhadnite čo najtesnejšie a svoj odhad zdôvodnite.

```
int k = p.length;
boolean jeDruhy = true;
while (k > 0) {
      if (jeDruhy)
            spracujZaciatok(p, k);

      jeDruhy = !jeDruhy;
      k = k / 2;
}
```

9. (2b) Navrhnite spôsob, ako možno v čase $O(n.logn)$ zistiť, či n-prvkové pole obsahuje nejaké 2 rovnaké hodnoty.

10. (1.5b) Predpokladajte, že máte už naprogramovanú funkciu `pivotuj`, ktorá zrealizuje pivotizáciu v časti poľa *p* ohraničenom indexami `odIdx`, `poIdx` a vráti pozíciu pivota v poli po pivotizácií. S využitím tejto metódy napíšte metódu, ktorá zrealizuje triedenie QuickSort.

```
public static int pivotuj(int[] p, int odIdx, int poIdx);

public static void quickSort(int[] p, int odIdx, int poIdx) {









}
```

11. (1b za správnu a -0.5b za nesprávnu odpoveď) **Rozhodnite** o pravdivosti tvrdení:

A: Najväčší prvok v binárnom vyhľadávacom strome je aj v pre-order, aj v in-order, aj v post-order zápise posledný.

Áno Nie

B: Uvažujme 2 polia dĺžky n. V čase $O(n.logn)$ môžeme zistiť, či jedno pole je permutáciou druhého.

Áno Nie

C: Triedenia výberom (MinSort, SelectionSort) spraví na každom n-prvkovom poli $O(n^2)$ krokov. Zároveň však platí, že počet vykonaných krokov je vždy $\Theta(n^2)$.

Áno Nie

D: Bublinkové triedenia (BubbleSort) spraví na každom n-prvkovom poli $O(n^2)$ krokov. Zároveň však platí, že počet vykonaných krokov je vždy $\Theta(n^2)$.

Áno Nie

E: Ak má n-prvkový binárny vyhľadávací strom aspoň n/2 listov, potom jeho výška je $O(log\ n)$.

Áno Nie

F: Ak máme usporiadané n-prvkové pole, potom v čase $O(log\ n)$ môžeme vypočítať súčet hodnôt v tomto poli.

Áno Nie

Poznámka: krok = elementárna operácia

Zdôvodnite svoju odpoveď k ľubovoľnej otázke okrem otázky F (2b):