Polsemestrálny test
teoretická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Píšte prosím čitateľne!

Hodnotenie, vyplní opravujúci:

Meno a priezvisko:	Skupina PAZ:	

1/2	2/1	3/1.5	4/1,5	5/1	6/1	7/1	8/1,5	9/2	10/7	11/3	Σ22,5

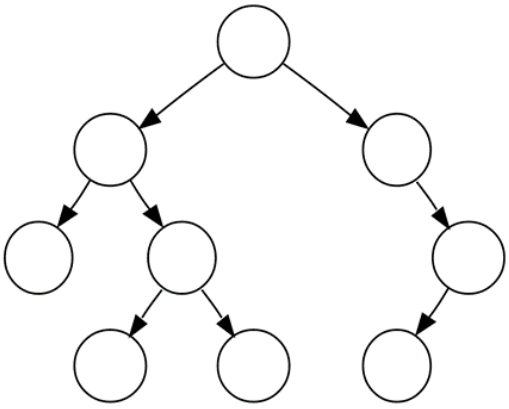
Zdôvodnenia majú byť stručné (1-3 vety) a zachycujúce podstatné argumenty.

1. (1+1b) Majme metódu `fraktal` uvedenú nižšie.
   a) Nakreslite obrázok, ktorý vznikne po volaní metódy `fraktal(4,100)`.
   b) Koľko krát sa zavolá metódá `fraktal` s úrovňou nula počas vykonávania metódy `fraktal(4,100)`?

   Počet volaní: _____

```
public void fraktal(int uroven, double dlzka) {
     if (uroven > 0) {
          for (int i = 0; i < uroven; i++) {
               turn(360.0 / uroven);
               step(dlzka);
               fraktal(uroven - 1, dlzka / 2);
               step(-dlzka);
          }
     }
}
```

2. (1b) Do stromu vpíšte znaky tak, aby jeho postorder postupnosť bola ZLOZITOST.



3. Selection sort vykoná vždy $O(n^2)$ operácii, na rozdiel od Bubble sortu, ktorý môže skončiť skôr, keď už je pole usporiadané. Preto sa váš spolužiak rozhodol takto vylepšiť aj Selection sort, viď kód nižšie, zmeny v kóde sú vyznačené červenou farbou. Metoda `jeUsporiadane` ma zistiť, či už je pole usporiadane. Túto metódu využíva aby ukončil Selection sort, keď už je pole usporiadané.

   (0.5b) Dopíšte kód metódy `jeUsporiadane`, aby mala, čo najlepšiu časovú zložitosť.
   (0.5+0.5b) Pomocou O notácie zapíšte aké sú asymptotické časové zložitosti metód `jeUsporiadane`, a takto vylepšenej metódy `selectionSort`.

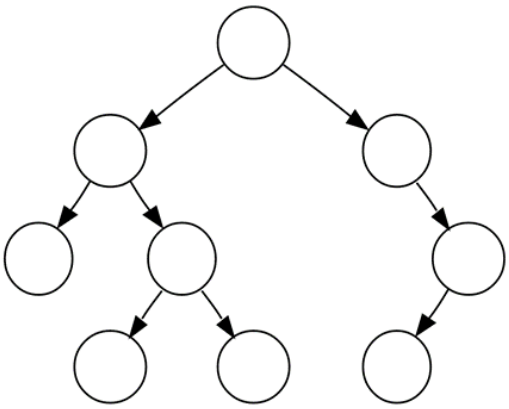
```
public static void selectionSort(int[] p, int odIdx, int poIdx) {
  if (odIdx == poIdx)
       return;
  vymen(p, odIdx, indexNajmensieho(p, odIdx, poIdx));
  if (!jeUsporiadane(p))
       selectionSort(p, odIdx + 1, poIdx);
}

public static boolean jeUsporiadane (int[] p) {









}
```

   Časová zložitosť metódy `jeUsporiadane`:

   Časová zložitosť metódy `selectionSort`:

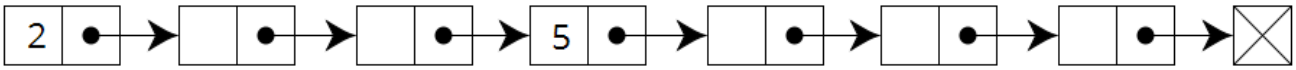
4. (0.5+1b) Uvažujme binárny vyhľadávací strom s 9 vrcholmi nakreslený dole. V tomto strome označte znakom X ten vrchol (alebo tie vrcholy), v ktorých môže byť uložená 4. najväčšia hodnota.

   Nakreslite binárnu haldu s 10 vrcholmi a v nej označte znakom X ten vrchol (alebo tie vrcholy), v ktorých môže byť uložená 4. najväčšia hodnota.



5. (1b) Nájdite a opravte (dopíšte, prepíšte, škrtnite, ...) 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;
}
```



6. (1b) Uvažujme spájaný zoznam nižšie a metódu triedy `SpajanyZoznam`.

```
public void zahada() {
  Uzol aktualny = prvy;
  while (aktualny != null) {
    if ((aktualny.dalsi != null) &&
          (aktualny.dalsi.hodnota % 2 == 0)) {
      aktualny.dalsi = aktualny.dalsi.dalsi;
    } else {
      aktualny = aktualny.dalsi;
    }
  }
}
```

Doplňte do spájaného zoznamu chýbajúce hodnoty tak, aby po aplikovaní metódy `zahada` mal zoznam dĺžku práve 4.

7. (1b) Doplňte definíciu:
`Binárny strom nazveme binárnym vyhľadávacím stromom práve vtedy, ak ...`

8. (0.5b za správnu, -0.5b za nesprávnu a 0b za žiadnu odpoveď)

Nech $f(n) = O(g(n))$ **Rozhodnite** o pravdivosti tvrdení vyznačením (napr. zakrúžkovaním) možnosti Áno alebo Nie

a: $3.f(n) + 2.g(n) = O(g(n))$ Áno Nie

b: $4.f(n).g(n) = O(g(n))$ Áno Nie

c: $2.f(n) + 2.n = O(g(n))$ Áno Nie

9. (2b) Uvažujme fragment kódu z triedy reprezentujúcej binárny strom celých čísel. Upravte (dopíšte, prepíšte, škrtnite, ...) metódu `prechodSoZasobnikom` tak, aby po jej volaní na koreni stromu následne metóda `vypisZasobnik` volaná s rovnakým zásobníkom vypísala inorder prechod stromom.

```
public class Uzol {
   int hodnota;
   Uzol lavy;
   Uzol pravy;

   public void prechodSoZasobnikom(Deque<Integer> zasobnik){
      if (lavy != null)
          lavy.prechodSoZasobnikom(zasobnik)
      if (pravy != null)
          pravy.prechodSoZasobnikom(zasobnik)
   }

   public static void vypisZasobnik(Deque<Integer> zasobnik) {
      while (!zasobnik.isEmpty()) {
        System.out.print(zasobnik.pop + ", ");
}
```

10. (1b za správnu, -0.5b za nesprávnu a 0b za žiadnu odpoveď) **Rozhodnite** o pravdivosti tvrdení vyznačením (napr. zakrúžkovaním) možnosti Áno alebo Nie:

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: Pre pole obsahujúce kladné aj záporne celé čísla môžeme v čase $\Theta(n.\log n)$ zistiť najmenšie kladné číslo.

Áno Nie

C: Maximálna hodnota v každom binárnom vyhľadávacom strome je uložená v uzle, ktorý nemá pravého potomka.

Áno Nie

D: Uvažujme ľubovoľnú n-prvkovú postupnosť celých čísel uloženú v poli. Potom existuje algoritmus v čase $O(n)$ a s pamäťou $O(1)$ (t.j. bez pomocného poľa) taký, že dokáže preusporiadať čísla v poli tak, že najprv budú v poli párne čísla a potom nepárne čísla.

Áno Nie

E: Ak máme haldu uloženú v poli, potom je možné nájsť minimálny prvok pričom vždy stačí pristúpiť k nanajvýš $n/2$ prvkom a nemusíme pristupovať ku všetkým n prvkom poľa.

Áno Nie

F: Zistiť, či je pole haldov má rovnakú zložitosť ako uhaldovanie poľa.

Áno Nie

G: V usporiadanom spájanom zozname (hodnoty usporiadané od najmenšej po najväčšiu) vieme zistiť, či sa tam nachádza hľadaná hodnota v čase $O(n.\log n)$.

Áno Nie

11. (1+1+1b) Zdôvodnite svoju odpoveď ku ľubovoľným **trom** tvrdeniam z predošlej úlohy - **napíšte ktoré úlohy zdôvodňujete**.