Midterm 2012, verzia A Meno a priezvisko: _____
obsahuje 5 príkladov, spolu 29>25 bodov skupina: _____

1A) [8 bodov] Už asi viete, že x<<1 je 2*x, x>>1 je x/2 a x&1 je x%2. Potom zistiť, čo počíta nasledujúca funkcia pre x>=0 je pomerne ľahké. Hint: f(7) = 7 a f(13) = 13.

```
public static int f(int x) {
  if (x > 0)
      return (f(x>>1) << 1)+(x&1);
  else
      return 0;
}
```

Pre istotu to vyjadrite pár slovami a máte prvý bod **[1 bod]**:

Jadrom úlohy je ale nasledujúca (a podobná) funkcia g s dvomi nezápornými argumentami x, y >= 0:

```
public static int g(int x, int y) {
    if (x>0)
        return (g(x>>1 , y)<<1)+(x&1);
    else if (y>0)
        return (g(x, y>>1)<<1)+(y&1);
    else
        return 0;
}
```

Vyjadrite pár (<=16) slovami, čo počíta rekurzívna funkcia g. Pre istotu a overenie vašej hypotézy Vám prezradíme, že g(7,7) = 63, g(7, 13) = 111, g(13, 7) = 125, g(13,13) = 221. Správna odpoveď je krátka, napr. tvaru "dĺžka binárneho zápisu čísla x+y". Nepopisuje ako rekurzia funguje, to nehodnotíme **[2body]**

- Vypočítajte g(10, 12) **[1 bod]**. g(10, 12) =
- Dokážte, alebo vyvráťte: „pre ľubovoľné prirodzené číslo v, existujú x,y také, že g(x,y) = v“ **[2 body]**:

☐ Tvrdenie platí a dôkaz je:

☐ Tvrdenie neplatí a kontra-príklad je:

- Definujte nerekurzívnu, a čo najelegantnejšiu, verziu g v Jave **[2 body]**:

Hint:

i	Integer.toBinaryString(i)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011
20	10100

————— riešenia príkladu 1 píšte na tento list (aj z druhej strany) alebo si vypýtajte ďalší list

Midterm 2012, verzia B Meno a priezvisko: _____
obsahuje 5 príkladov, spolu 29>25 bodov skupina: _____

1A) [8 bodov] Už asi viete, že x<<1 je 2*x, x>>1 je x/2 a x&1 je x%2. Potom zistiť, čo počíta nasledujúca funkcia pre x>=0 je pomerne ľahké. Hint: f(7) = 7 a f(13) = 13.

```
public static int f(int x) {
  if (x > 0)
      return (f(x>>1) << 1)+(x&1);
  else
      return 0;
}
```

Pre istotu to vyjadrite pár slovami a máte prvý bod **[1 bod]**:

Jadrom úlohy je ale nasledujúca (a podobná) funkcia h s dvomi nezápornými argumentami x, y >= 0:

```
public static int h(int x, int y) {
    if (y>0)
        return (h(x, y>>1)<<1)+(y&1);
    else if (x>0)
        return (h(x>>1 , y)<<1)+(x&1);
    else
        return 0;
}
```

Vyjadrite pár (<=16) slovami, čo počíta rekurzívna funkcia h. Pre istotu a overenie vašej hypotézy Vám prezradíme, že h(7,7) = 63, h(7, 13) = 125, h(13, 7) = 111, h(13,13) = 221. Správna odpoveď je krátka, napr. tvaru "dĺžka binárneho zápisu čísla x+y". Nepopisuje ako rekurzia funguje, to nehodnotíme **[2body]**

- Vypočítajte h(12, 14) **[1 bod]**. h(12, 14) =
- Dokážte, alebo vyvráťte: „pre ľubovoľné prirodzené číslo v, existujú x,y také, že h(x,y) = v“ **[2 body]**:

☐ Tvrdenie platí a dôkaz je:

☐ Tvrdenie neplatí a kontra-príklad je:

- Definujte nerekurzívnu, a čo najelegantnejšiu, verziu h v Jave **[2 body]**:

Hint:

i	Integer.toBinaryString(i)
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011
20	10100

———————— riešenia príkladu 1 píšte na tento list (aj z druhej strany) alebo si vypýtajte ďalší list

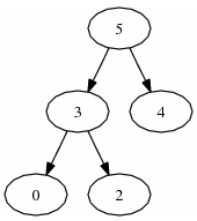
2) [6 bodov] Trieda BNode popisuje známu implementáciu binárneho stromu:

```
public class BNode<E extends Comparable<E>> {
    BNode<E> left;           // ľavý podstrom
    E key;                   // hodnota vrchola
    BNode<E> right;          // pravý podstrom
    public BNode(BNode<E> left, E key, BNode<E>right) {
      this.left=left; this.key=key; this.right = right;
    }
```

Definujte v nej metódu **`boolean isHeap(),`** ktorá platí, ak strom predstavuje binárnu haldu, t.j.

- binárny strom, ktorého všetky vetvy od koreňa majú dĺžku k alebo k-1 pre nejaké k **[2 body]**,
- hodnoty na všetkých vetvách od koreňa sú usporiadané zostupne (od najväčšej hodnoty (koreň je maximum) po najmenšiu) **[2 body]**,
- všetky vetvy dĺžky k sú vľavo od tých dĺžky k-1 **[2 body]**.

Ak poznáte inú definíciu haldy, napíšte ju, a programujte tú. V prípade čiastočného riešenia sa hodnotia vyznačené časti, ktoré ste naprogramovali správne. Pozn.: Prázdny strom je halda.

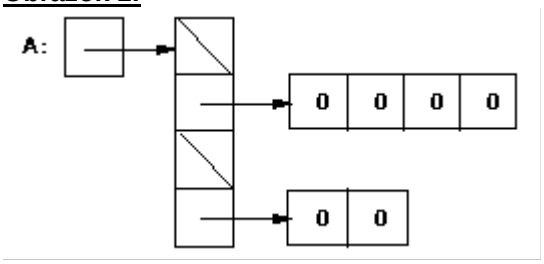



```
BNode<Integer> r =
       new BNode<Integer>(
              new BNode<Integer>(
                            new BNode<Integer>(null,0,null),
                            3,
                            new BNode<Integer>(null,2,null)),
                     5,
                     new BNode<Integer>(null, 4, null));
System.out.println(r.isHeap());
```

————— riešenia príkladu 2 píšte na tento list alebo si vypýtajte ďalší list

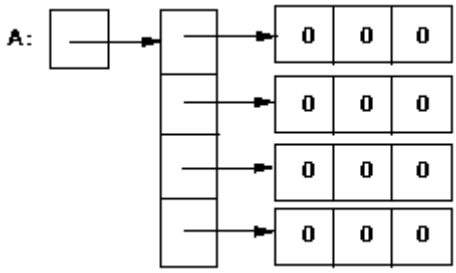
3) [5 bodov] Ku každému obrázku napíšte kód, ktorý vytvorí znázornené pole v pamäti. Prečiarknutá bunka prestavuje hodnotu **null**. Ak taký kód neexistuje, napíšte NEEXISTUJE. Nedajte sa nachytať ! K ďalším zadaným kódom (č. 6-10) napíšte, či program je syntakticky korektný, či zbehne. Ak nie, približne na akej chybe skončí. **[1/2 bodu za každú správnu odpoveď]**

Obrázok 1:



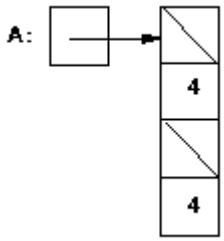
Kód 1:

Obrázok 2:



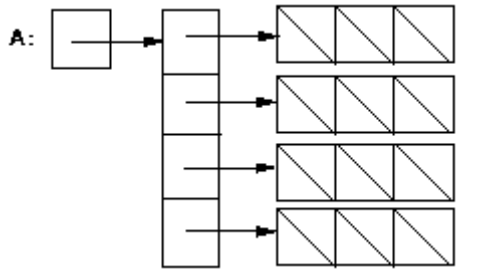
Kód 2:

Obrázok 3:



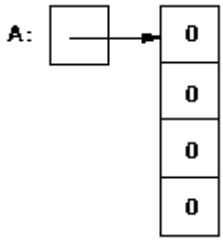
Kód 3:

Obrázok 4:



Kód 4:

Obrázok 5:



Kód 5:

Kód 6:
```
int A[5];
```
Odpoveď 6:
```
syntax: ok/chyba
sémantika: zbehne/padne s chybou:
```

Kód 7:
```
int[] A, B;
B = 0;
```
Odpoveď 7:
```
syntax: ok/chyba
sémantika: zbehne/padne s chybou:
```

Kód 8:
```
int[] A = {1,2,3};
int[] B;
B = A;
```
Odpoveď 8:
```
syntax: ok/chyba
sémantika: zbehne/padne s chybou:
```

Kód 9:
```
int[] A;
A[0] = 0;
```
Odpoveď 9:
```
syntax: ok/chyba
sémantika: zbehne/padne s chybou:
```

Kód 10:
```
int[] A = new int[20];
int[] B = new int[10];
A = B;
A[15] = 0;
```
Odpoveď 10:
```
syntax: ok/chyba
sémantika: zbehne/padne s chybou:
```

4) [5 bodov] Nasledujúci kód sa spustí s argumentami:
`java Words pec nam spadla pec nam spadla kto ze nam ju postavi`

```
import java.util.*;
public class Words {
  public static void main(String[] args){
    TreeMap<Integer,HashSet<String>> x=new TreeMap<Integer,HashSet<String>>();
      for(String arg : args) {
            HashSet<String> l = x.get(arg.length());
            if (l == null)
                  l = new HashSet<String>();
            l.add(arg);
            x.put(arg.length(),l);
      }
      System.out.println(x);
      // ----------
      TreeMap<String,Integer> y = new TreeMap<String,Integer>();
      // ... dopíšte sem









      // ...
      System.out.println(y);
} }
```

výpis premennej x vypíše podľa Vás **[označte správnu možnosť, 2 body]**:

- {2=[ju, ze], 3=[nam, nam, nam, pec, pec, kto], 6=[spadla, spadla], 7=[postavi]}
- {2=[ze, ju], 3=[pec, nam, pec, nam, kto, nam], 6=[spadla, spadla], 7=[postavi]}
- {2=[ze, ju], 3=[pec, nam, kto], 6=[spadla], 7=[postavi]}
- {2=[ju, ze], 3=[nam, pec, kto], 6=[spadla], 7=[postavi]}
- {[ju, ze]=2, [nam, pec, kto]=3, [spadla]=6, [postavi]=7}
- {[ju, ze], [nam, pec, kto], [spadla], [postavi]}
- {[ju, ze, nam, pec, kto, spadla, postavi]}
- niečo ine:

V mieste označenom `//...` dopíšte kus kódu, ktorý zo štruktúry v premennej x vytvorí `TreeMap<String,Integer>` y tak, aby pri výpise y sa vypísalo `{ju=2, kto=3, nam=3, pec=3, postavi=7, spadla=6, ze=2}`. Kľúčmi sú všetky utriedené reťazce zo štruktúry x bez opakovania a hodnotami sú dĺžky týchto reťazcov. **[3 body]**

——— riešenia príkladu 4 píšte na tento list, aj zozadu, alebo si vypýtajte ďalší list

5) [5 bodov] Cyklický zoznam je prázdny, alebo obsahuje prvky $x_1,...,x_n$. Cyklický je preto, že za prvkom x_n nasleduje x_1. Definujte **triedu CList<E>** s nasledujúcimi metódami:

- **`CList()`** – konštruktor vytvorí prázdny cyklický zoznam,
- **`E actual() throws Exception`** – vráti aktuálny prvok, na ktorom stojíme **[1 bod]**,
- **`void add(E elem)`** – pridá elem do zoznamu pred aktuálny prvok, na ktorom stojíme **[2 body]**,
- **`E next() throws Exception`** – vráti aktuálny a posunie sa na následujúci (cyklicky) **[2 body]**

Při definícii triedy **CList nesmiete použiť pole, ani žiadnu definovanú kolekciu.** Môžete predpokladať, že trieda **Node<E>** (z prednášky) je definovaná. V prípade nejasností na obmedzenia sa radšej spýtajte vopred.

```
public class Node<E> {
     public Node() {…}
     public Node(E e, Node<E> n) { … }
     public E getElement() { … }
     public Node<E> getNext() { … }
     public void setElement(E newElem) { … }
     public void setNext(Node<E> newNext) { … }
}
```

Test CList:

```
     CList cl = new CList(); // prázdny zoznam
     cl.add(1);              // [1] aktuálny je 1
     cl.add(2);              // [2,1] aktuálny je 2
     cl.add(3);              // [3,2,1] aktuálny je 3
     cl.actual());           // 3
     cl.next());             // 3
     cl.next());             // 2
     cl.next());             // 1
     cl.next());             // 3
     cl.next());             // 2
     cl.next());             // 1
```

——— riešenia príkladu 5 píšte na tento list, aj zozadu, alebo si vypýtajte ďalší list