Midterm 2016 Meno a priezvisko: _____
obsahuje 5 príkladov, spolu 4+7+7+4+5 = 27 bodov skupina: _____

1) [4 body] Dobre uzátvorkovaný výraz je taký, že každá otváracia má neskôr zodpovedajúcu zatváraciu zátvorku, pričom ozátvorkované oblasti môžu byť vnorené, ale nesmú sa kríšovať. Koncept dobre poznáte zo školskej aritmetiky, ak potrebujete, pýtajte si rekurzívnu formalizáciu.
Príklady: (),(()),(()()), ((())),()()() sú dobre uzátvorkované, ale)(, ((, (()),))((, ()) nie sú.
[2 body] Definujte metódu `public static boolean test1(String s)`, ktorá zistí, či neprázdny reťazec len znakov (ľubovoľne) je dobre uzátvorkovaný.

Ak pridáme ďalší typ zátvoriek (ľubovoľne), tak je situácia komplikovanejšia. Príklady: ()[],([]),([]()), ([()]),()[]() sú dobre uzátvorkované, ale (], ([)],][, ([],)][(, [(] nie sú.
[2 body] Definujte metódu `public static boolean test2(String s)`, ktorá zistí, či neprázdny reťazec len znakov (ľubovoľne) je dobre uzátvorkovaný.
Hint: `java.util.Stack;` má metódy push, pop, peek=top

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

2) [7 bodov] Triedy popisujú implementáciu binárneho stromu s hodnotami typu Integer vo vnútorných vrcholoch:

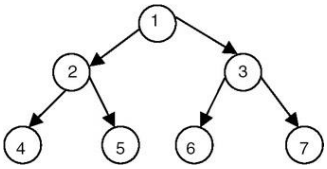
```
public class Node {
    Node left;      // ľavý podstrom
    Integer key;    // hodnota vrchola
    Node right;   } // pravý podstrom, konštruktor: public Node(Node left, Integer k, Node right)
public class Tree {
    Node root;    }
```

[2 body] Binárny strom je úplný, ak je prázdny (`null`), alebo výška ľavého a pravého podstromu sú rovnaké, a to platí vo všetkých vrcholoch stromu. V triede `Tree` definujte metódu `public boolean isPerfect()`, ktorá testuje úplnosť stromu. Výška stromu null je 0, ostatné stromy majú kladnú výsku. **Nelineárne riešenie nemôže získať maximum bodov.**

[1 bod] Úplný binárny strom výšky n má 2^n-1 vrcholov, dokážte (napr. mat.indukciou).

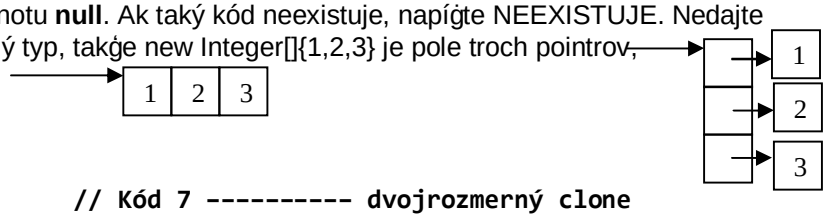
[2 body] V triede `Tree` definujte metódu `public boolean allDifs()`, ktorá testuje, či všetky hodnoty v strome sú rôzne.

[2 body] V Triede `Tree` definujte konštruktor `public Tree(int n)` , ktorý vytvorí úplný binárny strom výšky n obsahujúci len **rôzne hodnoty typu Integer**, teda **rôzne čísla** vo vrcholoch stromu. Ako rôzne a aké čísla, je úplne len na vás, napríklad 1..2^n-1. Nie je podstatné kopírovať obrázok, slúži len ako ilustratívny námet.

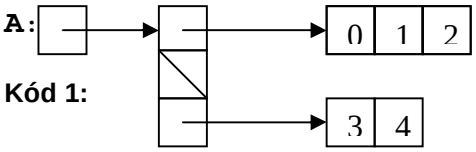


ï ï ï ï ï ï ï ï ï ï riešenia príkladu 2 píšte na tento list alebo si vypýtajte ďalší list

3) [7 bodov = 0.5 bodu za kaģdú správnu odpoveĻ] Ku kaģdému obrázku napíģte kód, ktorý vytvorí znázornené pole v pamäti. Preļ iarknutá bunka prestavuje hodnotu **null**. Ak taký kód neexistuje, napíģte NEEXISTUJE. Nedajte sa nachytaŠ! Uvedomte si, ģe Integer je referenļ ný typ, takģe new Integer[]{1,2,3} je pole troch pointrov, ale new int[]{1,2,3} je pole troch 32-bitových ļ ísel.



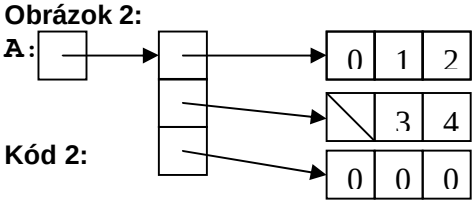
Obrázok 1:



Kód 1:

```
// Kód 7 ---------- dvojrozmerný clone
Integer[][] p ={{0,1},{2,3},null,{4,5}};
Integer[][] clon=(Integer[][])p.clone();
p[0][0] = 4;  //čo vypíše následujúci print ?
System.out.println(16/ clon[0][0]);
```

Obrázok 2:

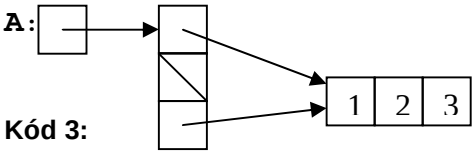


Kód 2:

nakreslite obrázky pamäte pre premenné A,B pre kódy:

```
// Kód 8 --------
String [][] A = new String[4][];
A[0] = new String[4];
A[2] = new String[2];
String[][] B = (String [][])A;
```

Obrázok 3:

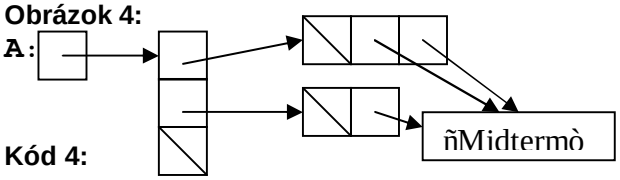


Kód 3:

```
// Kód 9 --------
String [][] A = new String[2][2];

// Kód 10 --------
  int [][] A = new int[4][3];
  int [] B = {1,2,3,4,5,6,7,8,9,10};
  System.arraycopy(B,0,A[0],0,3);
  System.arraycopy(B,1,A[1],0,3);
  System.arraycopy(B,2,A[2],0,3);
  System.arraycopy(B,3,A[3],0,3);
```

Obrázok 4:



Kód 4:

prepíģte kód bez pouģitia (&, &&, |, ||)

```
// Kód 11 --------
int a = ..., b = ...;  // nejaká hodnota
int c = b & 0xFF;
if ((a & 0) == (b & 0)) c++;
```

ńalej **preļ iarknite** riadky, ktoré sú:

- syntakticky nesprávne ï kompilátor by na nich zahlásil chybu (tie **preļ iarknite** a **oznaļ te S**),
- spôsobia chybu poļ as behu programu (**oznaļ te R**).

```
//Kód 5 -------- args
public class Args {
  public static void main(String[]
     args){
    System.out.print(args[1]);
  }
}
spustíme s argumentom > java Args 1
```

```
// Kód 6 ------- dvojrozmerné pole
Integer[][] p={{0,1},{2,3},null,{4,5}};
   for(int i=0; i<p.length; i++)
     for(int j=p.length; i>0; j--) {
       Integer[] temp = p[i];
        p[j] = p[i];
        p[i] = temp;
     }
```

prepíģte kód bez pouģitia (&, &&, |, ||, !)

```
// Kód 12 --------
boolean b = ...;
if (b || !b) System.out.
        println("that's the question!");
if (b & b == b && b == b) System.out.
        println("total Hamlet!");
```

prepíģte kód bez pouģitia (<<, >>)

```
// Kód 13 --------
int b = ...
int d = 1<<b+1;
int e = 1<<(b+1);
int f = b>>3;
```

prepíģte kód bez pouģitia (%, /)

```
// Kód 14 --------
int a = ......;
if (a % 2 == 0) a++;
int b = (a / 256) % 256;
```

Help: arraycopy(Object src, int srcPos, Object dest, int destPos, int length) kopíruje od srcPos po srcPos+length-1 z poĉa src na destPos aģ destPos+length-1 v poli dest.

rieģenia príkladu 3 píģte na tento list, aj zozadu, alebo si vypýtajte ĻalģíList

pomocný papier na rieġenia príkladu 3

ï ï ï ï ï ï ï ï rieġenia príkladu 3 píġte na tento list, aj zozadu, alebo si vypýtajte Ņalġí list

4) [4 body] Nasledujúci kód sa spustí s argumentami `java Words kto sa smeje naposledy ten sa smeje najlepšie`

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

Prvý `System.out.println(x)` podľa Vás vypíše **[označte správnu možnosť, 1 bod]**:

- `{2=[sa, sa], 3=[kto, ten], 5=[smeje, smeje], 9=[naposledy, najlepšie]}`
- `{0=[sa], 1=[kto], 2=[ten], 3=[smeje], 4=[smeje], 5=[naposledy], 6=[najlepšie]}`
- `{[sa], [kto, ten], [smeje], [naposledy, najlepšie]}`
- `{[sa], [kto, ten], [smeje], [najlepšie, naposledy]}`
- `{2=[sa, sa], 3=[kto, ten], 5=[smeje, smeje], 9=[najlepšie, naposledy]}`

V mieste označené enom **`//...`** dopíšte kus kódu, **ktorý zo štruktúry v premennej x** vytvorí TreeMap<String,Integer> y tak, aby pri výpise y sa vypísalo `{kto=3, najlepšie=9, naposledy=9, sa=2, smeje=5, ten=3}`. Kľúče mi sú výsledky utriedené reťazce zo vstupu bez opakovania a hodnotami sú ich dĺžky. V tejto časti kódu nepoužívajte parameter args. **[3 body]**

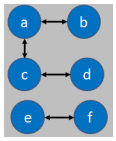
ïïïïïïï riešenia príkladu 4 píšte na tento list, aj zozadu, alebo si vypýtajte ďalší list

5) [5 bodov] Definujte triedu `FaceHook`, ktorá reprezentuje sieŠpriateÕov. Kaġdá osoba je identifikovaná jednoznaļ ným menom typu `String`. PriateĆstvo je (pre úļ ely tohoto zadania ;-) symetrická relácia. Trieda `FaceHook` musí implementovaŠtieto metódy, vnútorná reprezentácia je úplne na vás:

- **[2 bod]** `void bp(String meno1, String meno2)` ï `meno1!=meno2` sú blízki priatelia (b.p.), tak ich pridá do ġtruktúry.
- **[3 body]** `boolean vp(String meno1, String meno2)` vráti true, ak `meno1` a `meno2` sú vzdialení priatelia (v.p.). Vzdialený priateĆje alebo blízky priateĆalebo je to vzdialeným priateÕom nejakého môjho blízkeho priateĆa.

Hint: tu sa myslí tranzitívny uzáver relácie blizkyPriatel, alebo cesta v grafe,é Príklad:

```
FaceHook fk = new FaceHook();
fk.bp("a","b");
fk.bp("a","c");
fk.bp("c", "d");
fk.bp("e","f");
System.out.println(fk.vp("b","c"));          // true
System.out.println(fk.vp("c","b"));          // true
System.out.println(fk.vp("b","d"));          // true
System.out.println(fk.vp("a","e"));          // false
```



ï ï ï ï ï ï ï rieġenia príkladu 5 píġte na tento list, aj zozadu, alebo si vypýtajte Ņalġí list