Parcial 1 - Pràctiques - PRG - ETSInf - Curs 2013/14
23 de juny de 2014 - Durada: 50 minuts

1. [2 punts] Completar el següent mètode perquè resolga el problema de les Torres d'Hanoi:

```
public static void hanoi(int discos, String origen, String desti, String auxiliar) {
   if( /* COMPLETAR */ )           // Cas base
      moureDisc( origen, desti );
   else {                          // Cas general
      hanoi( /* COMPLETAR */ );
      moureDisc( /* COMPLETAR */ );
      hanoi( /* COMPLETAR */ );
   }
}
```

Solució:

```
public static void hanoi( int discos, String origen, String desti, String auxiliar ) {
   if( discos == 1 )           // Cas base
      moureDisc( origen, desti );
   else {                          // Cas general
      hanoi( discos-1, origen, auxiliar, desti );
      moureDisc( origen, desti );
      hanoi( discos-1, auxiliar, desti, origen );
   }
}
```

2. [3 punts] Implementar un mètode **RECURSIU** el perfil del qual ha de ser:

```
public static int aparicions(String a, String b)
```

que retorna la quantitat d'aparicions de la cadena `a` en la cadena `b`. Per exemple, si `a`=*coc* i `b`=*coca de cocochas cocinadas con coco*, `a` apareix en `b` cinc vegades. Es suposarà que la cadena `a` no és la cadena buida.

S'ha de resoldre utilitzant el mètode implementat en la *pràctica 2*, el perfil del qual és:

```
public static boolean esPrefixe(String a, String b)
```

Solució:

```
public static int aparicions(String a, String b) {
   if ( a.length() > b.length() ) return 0;
   else { int c = esPrefixe(a,b) ? 1 : 0;
          return c + aparicions(a, b.substring(1));}
}
```

3. [5 punts] Es disposa d'un mètode amb perfil `public static void algorisme(int n)` que implementa cert algorisme, el cost del qual té com a talla el paràmetre `n`. Es demana completar el mètode:

```
public static void mesuraAlgorisme(int tallaIni, int tallaFi, int tallaInc, int numRep) {
    System.out.printf("# Talla    Temps promedi (sg.)\n");
    System.out.printf("#---------------------------\n");

                  /* COMPLETAR */

}
```

perquè mesure de forma empírica (pràctica) el cost de `algorisme(n)` invocant-ho per a talles compreses entre `tallaIni` i `tallaFi` amb increments de `tallaInc`. Per a millorar l'estimació del temps mesurat, el mesurament es repetirà `numRep` vegades per a cada talla, mostrant per pantalla el valor de cadascuna de les talles i la mitjana dels temps mesurats en **segons**.

Cal usar el mètode `public static long nanoTime()`, de la classe `java.lang.System`, que retorna el valor actual del temporitzador del sistema en **nano-segons** (1 nano-segon = 10^{-9}segons).

NOTA: es considera que els mètodes `algorisme` i `mesuraAlgorisme` es troben en la mateixa classe.

Un exemple d'una eixida per a la invocació `mesuraAlgorisme(10000, 20000, 1000, 10)` és:

```
# Talla    Temps promedi (sg.)
#-----------------------------
  10000         5.32
  11000         6.28
  12000         8.61
   ...          ...
  20000         30.45
```

Solució:

```
public static void mesuraAlgorisme(int tallaIni, int tallaFi, int tallaInc, int numRep) {
    System.out.printf("# Talla    Temps promedi (sg.)\n");
    System.out.printf("#-----------------------------\n");
    long t1 = 0, t2 = 0, tt = 0; double tmedt = 0;   // Temps
    for (int t=tallaIni; t<=tallaFi; t+=tallaInc) {
      tt = 0;                         // Temps acumulat inicial a 0
      for (int r=0; r<numRep; r++) {
        t1 = System.nanoTime();       // Temps inicial
        algorisme(t);
        t2 = System.nanoTime();       // Temps final
        tt += (t2-t1);                // Actualitzar temps acumulat
      }
      tmedt = (double)tt/numRep;      // Temps promedi del cas mitjà
      System.out.printf("%8d  %8d\n", t, tmedt*1.0e-9);
    }
}
```