Parcial 1 - PRÀCTIQUES - PRG - ETSInf. Curs 2015-16
11 d'abril de 2016. Duració: 1 hora

(Nota: L'examen s'avalúa sobre 10 punts, però el seu pes específic en la nota final de l'assignatura és de 0,8 punts)

NOM:	GRUP DE PRÀCTIQUES:

1. [2 punts] **Es demana:** respondre a les següents preguntes sobre la pràctica 1.

   a) Si sobre la torre `origen` hi ha 30 discos, què s'ha de fer amb els 29 discos que hi ha sobre el disc de major diàmetre abans de poder moure'l a la torre `destí`?

   Solució: Moure'ls a la torre `auxiliar`.

   b) En el mètode `hanoi`, per a un nombre de discos major que 1, la primera crida `moureDisc(origen, destí)` que s'executa, està associada al seu cas base o al general?

   Solució: Està associada al seu cas base.

2. [3 punts] El mètode `posFracSep(String)`, vist en la pràctica 2, determina recursivament la posició del separador de la part fraccionària d'un nombre en coma flotant ben format contingut en una `String`, açò és, la posició del caràcter punt `'.'` o coma `','` que aparega o -1 en cas que aquest no existisca.

   Es demana: completar amb les instruccions necessaries el mètode recursiu `posFracSep(String)` següent per tal que siga correcte, tenint en compte que s'ha optat per una descomposició recursiva descendent de la `String`, és a dir, s'ha considerat l'últim caràcter de la `String` i la substring inicial (des de la posició `0` fins la `s.length()-2`).

```
/** Torna la posició on es troba el separador de la part fraccionària
 * o -1 si no es troba.
 * @param s String que conté el valor en coma flotant.
 * @return int posició del separador o -1 si no es troba.
 * PRECONDICIÓ: s conté un nombre en coma flotant ben format. */
public static int posFracSep(String s) {
    int ult = s.length() - 1;
    if (s.charAt(ult) == '.' || s.charAt(ult) == ',') { return ult; }
    else { return posFracSep(s.substring(0, ult)); }
}
```

   Recorda que `s.substring(i, j)` torna un objecte `String` que representa la substring de `s` formada amb els caràcters compresos entre el `i` i el `j-1`.

Solució:

```
public static int posFracSep(String s) {
    if (s.length() == 0) { return -1; }
    else {
        int ult = s.length() - 1;
        if (s.charAt(ult) == '.' || s.charAt(ult) == ',') { return ult; }
        else { return posFracSep(s.substring(0, ult)); }
    }
}
```

3. [5 punts] Es vol mesurar el temps promedi d'execució d'un algorisme d'ordenació d'arrays d'enters, l'execució del qual, per a un array `a`, es fa mitjançant la crida:

```
AlgorismesMesurables.ordena(a);
```

A més, per millorar la precisió de la mesura, s'ha de repetir la mateixa un nombre `numReps` de repeticions.

Addicionalment, es disposa dels mètodes següents per inicialitzar un array segons el desitjat:

```
private static int[] crearArrayAleatori(int t)
private static int[] crearArrayOrdCreixent(int t)
private static int[] crearArrayOrdDecreixent(int t)
```

i, pot utilitzar-se la rutina de temporització del sistema, tal com s'ha vist en la pràctica 3:

```
long t = System.nanoTime();
```

Es demana: escriure un mètode amb la capçalera següent:

```
public static double tempsMilisPromedi(int t, int numReps)
```

que torne com a resultat el temps d'execució en el cas promedi, en mil·lisegons, de l'ordenació amb el métode `ordena(int[])` per a una talla `t` i amb un número de repeticions `numReps`.

Tingues en compte que s'ha de triar el mètode més convenient per a crear l'array segons el cas a mesurar i que no s'ha de tornar a ordenar un array ja ordenat.

Solució:

```
public static double tempsMilisPromedi(int t, int numReps) {
    final double NANOS_MILIS = 1e6;
    int[] a;
    long tIni, tFin;
    double acum = 0;
    for (int i = 0; i < numReps; i++) {
        a = crearArrayAleatori(t);
        tIni = System.nanoTime();
        AlgorismesMesurables.ordena(a);
        tFin = System.nanoTime();
        acum += tFin - tIni;
    }
    return acum / numReps / NANOS_MILIS;
}
```