Parcial 1 - PRÁCTICAS - PRG - ETSInf. Curso 2016-17
10 de abril de 2017. Duración: 1 hora

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de la asignatura es de **0,8 puntos**.

NOMBRE:	GRUPO DE PRÁCTICAS:

1. 4 puntos El método `esPrefijo(String, String)`, implementado en la práctica 2, devuelve `true` si el primer parámetro es prefijo del segundo y `false` en caso contrario.

   Se pide: completar el siguiente método recursivo `cuentaSubcadena(String, String)` para que, haciendo uso del método anterior, devuelva el número de veces que una cadena no vacía `a` es subcadena de otra cadena `b`. Por ejemplo, `cuentaSubcadena("ab", "aaba")` debe devolver `1`, `cuentaSubcadena("aa", "abac")` debe devolver `0` y `cuentaSubcadena("aa", "aaaa")` debe devolver `3`.

```
/** Precondición: a.length() > 0 */
public static int cuentaSubcadena(String a, String b) {
    if (a.length() <= b.length()) {
         if (esPrefijo(a, b)) { return /* COMPLETAR */; }
         else { return /* COMPLETAR */; }
    }
    else { return 0; }
}
```

   Recuerda que `s.substring(i)` es un método de la librería de Java que devuelve un objeto `String` que representa la substring de `s` formada por los caracteres comprendidos entre el `i` y el `s.length() - 1`.

Solución:

```
/** Precondición: a.length() > 0 */
public static int cuentaSubcadena(String a, String b) {
    if (a.length() <= b.length()) {
        if (esPrefijo(a, b)) { return 1 + cuentaSubcadena(a, b.substring(1)); }
        else { return cuentaSubcadena(a, b.substring(1)); }
    }
    else { return 0; }
}
```

2. 2 puntos **Se pide:** completar el método que sigue para que devuelva un array de enteros de talla `t` que contenga valores para el **caso peor** del método de ordenación por **inserción directa**, pero teniendo en cuenta que la diferencia entre dos valores consecutivos cualesquiera del array resultante debe ser como mínimo de 2.

```
private static int[] casoPeorInsercion(int t) {
    int[] a = new int[t];
    for ( /* COMPLETAR */ ) {
        a[i] = /* COMPLETAR */;
    }
    return a;
}
```

Solución:

El caso peor del método de ordenación por inserción directa se da cuando los elementos del array a ordenar están ordenados decrecientemente. A continuación, se presentan 2 formas distintas (de entre

muchas posibles soluciones) de completar el bucle `for` del método `casoPeorInsercion(int)` para que, siendo `a` el array resultante, se cumpla que $\forall$`i`, `0` $\leq$ `i` $<$ `a.length - 1`, `a[i] - a[i + 1]` $\geq$ `2`.

```
for (int i = 0; i < a.length; i++) {
   a[i] = t - (2 * i) - 1;
}

for (int i = 0, j = t - 1; i < a.length; i++, j -= 2) {
    a[i] = j;
}
```

3. [4 puntos] En la clase `AlgoritmosMedibles` está definido el método de ordenación por inserción directa con el siguiente perfil:

```
public static void insercion(int[] a)
```

En la clase `MedidaOrdenacion` están definidos los siguientes métodos para inicializar un array:

- `private static int[] crearArrayAleatorio(int t)` que devuelve un array de enteros de talla `t` con valores comprendidos entre `0` y `t - 1`.
- `private static int[] crearArrayOrdCreciente(int t)` que devuelve un array de enteros de talla `t` ordenado de forma creciente.
- `private static int[] crearArrayOrdDecreciente(int t)` que devuelve un array de enteros de talla `t` ordenado de forma decreciente.

Dado el siguiente fragmento de código en el que el método `medidaInsercionCasoPeor()`, definido también en la clase `MedidaOrdenacion`, está incompleto:

```
// Constantes que definen los parametros de medida
public static final int MAXTALLA = 10000, INITALLA = 1000;
public static final int INCRTALLA = 1000, REPETICIONES = 200;
public static final double NMS = 1e3;  // relacion micro - nanosegundos

public static void medidaInsercionCasoPeor() {
    System.out.printf("# Talla    Peor\n");
    System.out.printf("#-----------------\n");
    long tp1, tp2;
    int[] a;
    double tpeor;
    for (   /* COMPLETAR */   ) {

                /* COMPLETAR */

        tpeor /= REPETICIONES;
        System.out.printf(Locale.US, "%8d   %8.2f\n", t, tpeor / NMS);
    }
}
```

Se pide: completar el método `medidaInsercionCasoPeor` para obtener las medidas de tiempos (en microsegundos) del **caso peor** del método de ordenación por **inserción directa**.

Recuerda que el método `static long nanoTime()`, en `java.lang.System`, devuelve el valor actual del temporizador más preciso del sistema en nanosegundos.

Solución:

```
public static void medidaInsercionCasoPeor() {
    System.out.printf("# Talla    Peor\n");
    System.out.printf("#-----------------\n");
    long tp1, tp2;
    int[] a;
    double tpeor;
    for (int t = INITALLA; t <= MAXTALLA; t += INCRTALLA) {
        tpeor = 0;
        for (int r = 0; r < REPETICIONES; r++) {
            a = crearArrayOrdDecreciente(t);
            tp1 = System.nanoTime();
            AlgoritmosMedibles.insercion(a);
            tp2 = System.nanoTime();
            tpeor += tp2 - tp1;
        }
        tpeor /= REPETICIONES;
        System.out.printf(Locale.US, "%8d   %8.2f\n", t, tpeor / NMS);
    }
}
```