Práctica Individual 1 – Ejercicios iterativos, recursivos y notación funcional

A resolver en clases de prácticas por el profesor/a (<u>NO hay que incluirlos en la entrega</u>):

1. Un punto es un tipo con las siguientes propiedades:
   - X, Double, básica, individual
   - Y, Double, básica, individual
   - Cuadrante, Cuadrante, derivada, individual. Enumerado {PRIMER_CUADRANTE, SEGUNDO_CUADRANTE, TERCER_CUADRANTE, CUARTO_CUADRANTE}.

```
public static Map<Cuadrante, Double> ejemplo1 (List<Punto2D> ls) {
        return ls.stream()
                .collect(Collectors.groupingBy(Punto2D::cuadrante,
                Collectors.reducing(0., x -> x.x(), (x, y) -> x + y)));
}
```

Analice el código que se muestra y proporcione una solución iterativa y otra recursiva final equivalentes.

2. Dada la siguiente definición recursiva de la función f (que toma como entrada 2 números enteros positivos y devuelve una cadena):

$$f(a,b) = \begin{cases} "(" + toString(a*b) + ")", & a < 5 \vee b < 5 \\ toString(a+b) + f(a/2, b-2), & en\ otro\ caso \end{cases}$$

siendo + un operador que representa la concatenación de cadenas, y toString(i) un método que devuelve una cadena a partir de un entero. Proporcione una solución iterativa usando while, una recursiva no final, una recursiva final, y una en notación funcional.

3. Dada la siguiente definición recursiva de la función g (que toma como entrada 2 números enteros positivos y devuelve un entero):

$$g(a,b) = \begin{cases} a^2 + b, & a < 2 \ \vee b < 2 \\ g\left(\frac{a}{2}, b-1\right) + g\left(\frac{a}{3}, b-2\right), & en\ otro\ caso \end{cases}$$

Proporcione una solución recursiva sin memoria, otra recursiva con memoria, y otra iterativa.

A resolver por los estudiantes (SÍ hay que incluirlos en la entrega):

1. Analice el código que se muestra a continuación, en el que EnteroCadena es una clase con una propiedad entera a y otra de tipo cadena s (la cual debe implementar como un record):

```
public static String ejercicio1 (Integer varA, Integer varB) {
      UnaryOperator<EnteroCadena> nx = elem ->
      {
            return EnteroCadena.of(elem.a()+3,
                        elem.a()%2==0?
                        elem.a()+"*":
                        elem.a()+"!");
      };

      return Stream
      .iterate(EnteroCadena.of(varA,"A"), elem -> elem.a() < varB, nx)
                  .filter(elem -> elem.a()%10 != 0)
                  .map(elem -> elem.s())
                  .collect(Collectors.joining("-"));
}
```

SE PIDE: Proporcione una solución iterativa y otra recursiva final equivalentes.

2. Dada la siguiente definición recursiva de la función f (que toma como entrada 2 números enteros positivos, y devuelve una lista de enteros):

$$f(a,b) = \begin{cases} \text{[a*b]}, & a < 2 \vee b < 2 \\ f(a\%b, b-1) + a, & a > b \\ f(a-2, b/2) + b, & eoc \end{cases}$$

donde:

- *[elem]*: representa una lista de un único elemento *elem*
- *list* + *elem*: representa la inserción del elemento *elem* al final de la lista *list*

SE PIDE: Proporcione una solución recursiva no final, una iterativa usando while, una recursiva final, y una en notación funcional.

3. Se tienen 2 ficheros A y B con elementos de tipo cadena. Se desea obtener una lista de cadenas en la que aparezcan los elementos de A y B intercalados de dos en dos, de forma que se seleccionan los 2 primeros elementos de A, después los 2 primeros de B, y así sucesivamente (en caso de que al final de alguno de los ficheros sólo quede 1 elemento se selecciona sólo 1 en lugar de 2). Tenga en cuenta que:

- una vez se llegue al final de un fichero, se deben incluir a continuación en la lista resultante todos los elementos del otro fichero que queden pendiente.
- no se permite el volcado completo de los ficheros de entrada en estructuras intermedias (como listas, conjuntos o arrays), es decir, debe hacer uso de iteradores directamente sobre los ficheros. La única estructura de este tipo que se permite crear es aquélla que se use para construir la lista resultante.

SE PIDE: Proporcione una solución iterativa usando while, una recursiva final, y una en notación funcional.

4. Dada la siguiente definición recursiva de la función g (que toma como entrada 2 números enteros positivos y devuelve una cadena):

$$g(a,b) = \begin{cases} toString(a) + "." + toString(b), & a \leq 4 \\ toString(b) + "-" + toString(a), & b \leq 4 \\ g(a/2, \mathrm{b} - 2) + "," + g(a-2, b/2) + "," + g(a-1, b-1) & en\ otro\ caso \end{cases}$$

siendo + un operador que representa la concatenación de cadenas, y toString(i) un método que devuelve una cadena a partir de un entero.

SE PIDE: Proporcione una solución recursiva sin memoria, otra recursiva con memoria, y otra iterativa imperativa.

SE PIDE resolver todos los ejercicios de forma <u>eficiente</u>. Tenga en cuenta que:

- Para cada ejercicio debe leer los datos de entrada de un fichero, y mostrar la salida por pantalla. Dicha lectura debe ser independiente del algoritmo concreto que resuelva el ejercicio, excepto para el ejercicio 3, en el que debe seguir las directrices descritas en el enunciado del mismo.
- La solución tiene que ser acorde al material de la asignatura proporcionado.

DEBE REALIZAR SU ENTREGA EN <u>2 PARTES</u>:

1. Proyecto en eclipse con las soluciones en Java.
2. Memoria de la práctica <u>en un único archivo PDF</u>, que debe contener:
    - Código realizado
    - Volcado de pantalla con los resultados obtenidos para las pruebas realizadas, incluyendo al menos los resultados obtenidos para los tests proporcionados.