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<Punto2D.Cuadrante,Double> ejemplo1(List<Punto2D> l){
        return l.stream()
                .collect(Collectors.groupingBy(Punto2D::getCuadrante,
                Collectors.<Punto2D,Double>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 (<u>SÍ hay que incluirlos en la entrega</u>):

1.
```
public static Map<Integer,List<String>>  ejercicioA (Integer varA, String varB, Integer varC, String varD, Integer varE) {
        UnaryOperator<EnteroCadena> nx = elem ->
                {
                        return EnteroCadena.of(elem.a()+2,
                                        elem.a()%3==0?
                                        elem.s()+elem.a().toString():
                                        elem.s().substring(elem.a()%elem.s().length()));
                };

        return Stream.iterate(EnteroCadena.of(varA,varB), elem -> elem.a() < varC, nx)
                                .map(elem -> elem.s()+varD)
                                .filter(nom -> nom.length() < varE)
                                .collect(Collectors.groupingBy(String::length));
}
```

donde EnteroCadena es una clase con una propiedad entera *a* y otra de tipo cadena *s*, la cual debe implementar como un record.

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

$$f(a,b,s) = \begin{cases} a*a+b*b, & s.length = 0 \\ s.length + a + b, & a < 2 \;||\; b < 2 \\ a + b \; + f(a-1, \frac{b}{2}, s.substring(a\%s.length, b\%s.length)), & a\%s.length < b\%s.length \\ a * b \; + f(\frac{a}{2}, b-1, s.substring(b\%s.length, a\%s.length)), & en\ otro\ caso \end{cases}$$

3. A partir de 2 ficheros ordenados de objetos de tipo Punto2D, obtener una lista ordenada de puntos que incluya sólo los puntos del primer y tercer cuadrante. Para realizar la fusión debe hacer uso de iteradores directamente sobre los ficheros de entrada, no permitiéndose almacenar los puntos en listas y hacer fusión de listas.

4. Dada la siguiente definición recursiva de la función g (que toma como entrada 3 números enteros positivos y devuelve una cadena):

$$g(a,b,c) = \begin{cases} "(" + toString(a) + "+" + toString(b) + "+" + toString(c) + ")", & a < 2 \wedge b \leq 2 \vee c < 2 \\ "(" + toString(c) + "-" + toString(b) + "-" + toString(a) + ")", & a < 3 \vee b < 3 \wedge c \leq 3 \\ "(" + g(a-1, b/a, c-1) + "*" + g(a-2, b/2, c/2) + ")", & b\%a = 0 \wedge (a\%2 = 0 \vee b\%3 = 0) \\ "(" + g\left(\frac{a}{2}, \mathrm{b}-2, \mathrm{c}/2\right) + "/" + g(a/3, b-1, c/3) + ")" & en\ otro\ caso \end{cases}$$

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 resolver de forma eficiente:

- Ejercicio 1: Analice el código que se muestra y proporcione una solución iterativa y otra recursiva final equivalentes.
- Ejercicio 2: Proporcione una solución iterativa usando while, una recursiva no final, una recursiva final, y una en notación funcional.
- Ejercicio 3: Proporcione una solución iterativa usando while, una recursiva final, y una en notación funcional.
- Ejercicio 4: Proporcione una solución recursiva sin memoria, otra recursiva con memoria, y otra iterativa.

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.
- La solución tiene que ser acorde al material de la asignatura proporcionado.

DEBE REALIZAR SU ENTREGA EN 2 PARTES:

1. Proyecto en eclipse con las soluciones en Java.
2. Memoria de la práctica en un único archivo PDF, 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.