Ejercicio 1 – Recursividad

Dada la siguiente definición recursiva de la función f (donde los tres primeros parámetros son enteros y el cuarto es una cadena, devolviendo también una cadena):

$$f(a,b,c,d) = \begin{cases} c, & a < 3 \ \wedge \ b < 3 \\ "\sim" + \ f(a-1, b-1, c+4, d), & a < 3 \ \wedge \ b \leq 3 \\ "-" + c + "-" + f(a-2, b-1, c, d + "x"), & a \geq 3 \ \wedge b < 3 \\ "/" + d + "/" + f(a-1, b-2, c+1, d), & en\ otro\ caso \end{cases}$$

siendo + un operador que representa la concatenación de cadenas, o la suma de enteros en su caso.

SE PIDE:

a) Proporcione una solución recursiva no final en Java para f(a,b,c,d)
b) Proporcione una solución recursiva final en Java para f(a,b,c,d)
c) Proporcione una solución iterativa en Java para f(a,b,c,d)
d) Proporcione una solución funcional en Java para f(a,b,c,d)

Tiempo estimado: 50 min. **Puntuación: ADDA: 25%. EDA: 33,3%.**

SOLUCIÓN

a) Recursivo no final:

```
public static String f(Integer a, Integer b, Integer c, String
d) {
	String res;
	if (a < 3 && b < 3) {
		res = c.toString();
	} else if (a < 3) {
		res = "~" + f(a - 1, b - 1, c + 4, d);
	} else if (b < 3) {
		res = "-" + c + "-" + f(a - 2, b - 1, c, d + "x");
	} else {
		res = "/" + d + "/" + f(a - 1, b - 2, c + 1, d);
	}
	return res;
}
```

b) Recursivo final:

```
public static String f_rec_final(Integer a, Integer b, Integer
c, String d) {
	return f_aux(a, b, c, d, "");
}
public static String f_aux(Integer a, Integer b, Integer c,
String d, String acum) {
	String res;
	if (a < 3 && b < 3) {
		res = acum + c.toString();
	} else if (a < 3) {
		res = f_aux(a - 1, b - 1, c + 4, d, acum + "~");
	} else if (b < 3) {
		res = f_aux(a - 2, b - 1, c, d + "x", acum + "-" +
c + "-");
	} else {
		res = f_aux(a - 1, b - 2, c + 1, d, acum + "/" + d
+ "/");
	}
	return res;
}
```

c) Iterativo:

```
public static String f_iterativa(Integer a, Integer b, Integer
c, String d) {
	String res = "";
	while (!(a < 3 && b < 3)) {
		if (a < 3) {
			res += "~";
			a = a - 1;
			b = b - 1;
```

```
                c = c + 4;
            } else if (b < 3) {
                res += "-" + c + "-";
                a = a - 2;
                b = b - 1;
                d = d + "x";
            } else {
                res += "/" + d + "/";
                a = a - 1;
                b = b - 2;
                c = c + 1;
            }
        }
        res += c.toString();
        return res;
}
```

d) Funcional:

```
public record Tupla(Integer a, Integer b, Integer c, String d,
String res) {
public static Tupla of(Integer a, Integer b, Integer c, String
d, String res) {
        return new Tupla(a, b, c, d, res);
}

public Tupla next() {
        if (a < 3) {
                return Tupla.of(a - 1, b - 1, c + 4, d, res + "~");
        } else if (b < 3) {
                return Tupla.of(a - 2, b - 1, c, d + "x", res + "-"
+ c + "-");
        } else {
                return Tupla.of(a - 1, b - 2, c + 1, d, res + "/" +
d + "/");
        }
}

public boolean isCaseBase() {
        return a < 3 && b < 3;
}
}

public static String f_funcional(Integer a, Integer b, Integer
c, String d) {
        Tupla tupla = Stream.iterate(Tupla.of(a, b, c, d, ""), t
-> t.next()).filter(t -> t.isCaseBase()).findFirst().get();
        return tupla.res() + tupla.c().toString();
}
```