MEMORIA

PRACTICA INDIVIDUAL 1

Jaime Linares Barrera

2º Ingeniería Informática - Ingeniería del Software, Grupo 4

1. EJERCICIO 1

1.1. CÓDIGO EJERCICIO

```
package ejercicios;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.function.UnaryOperator;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public class Ejercicio1 {

    // FUNCIONAL
    public static Map<Integer, List<String>> fFuncional(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));
            }


    // ITERATIVA
    public static Map<Integer, List<String>> fIterativo(Integer varA, String varB,
            Integer varC, String varD, Integer varE) {
        Map<Integer, List<String>> ac = new HashMap<>();
        EnteroCadena e = EnteroCadena.of(varA, varB);
        while(e.a() < varC) {
            String valor = e.s() + varD;
            Integer clave = valor.length();
            if(clave < varE) {
                if(ac.containsKey(clave)) {
                    ac.get(clave).add(valor);
                } else {
                    List<String> ls = new ArrayList<>();
                    ls.add(valor);
                    ac.put(clave, ls);
                }
            }
            e = EnteroCadena.of(e.a()+2, e.a()%3==0?
                            e.s()+e.a().toString():
                            e.s().substring(e.a()%e.s().length()));
        }
        return ac;
    }

    // RECURSIVA FINAL
    public static Map<Integer, List<String>> fRecursivoFinal(Integer varA, String varB,
            Integer varC, String varD, Integer varE) {
        Map<Integer, List<String>> ac = new HashMap<>();
        EnteroCadena e = EnteroCadena.of(varA, varB);
        return fRecursivoFinal(e, ac, varA, varB, varC, varD, varE);
    }

    private static Map<Integer, List<String>> fRecursivoFinal(EnteroCadena e,
            Map<Integer, List<String>> ac, Integer varA, String varB, Integer varC,
            String varD, Integer varE) {
        if(e.a() < varC) {
            String valor = e.s() + varD;
            Integer clave = valor.length();
            if(clave < varE) {
                if(ac.containsKey(clave)) {
                    ac.get(clave).add(valor);
                } else {
                    List<String> ls = new ArrayList<>();
                    ls.add(valor);
                    ac.put(clave, ls);
                }
            }
            fRecursivoFinal(EnteroCadena.of(e.a()+2, e.a()%3==0?
                                e.s()+e.a().toString():
                                e.s().substring(e.a()%e.s().length())),
            ac, varA, varB, varC, varD, varE);
        }
        return ac;
    }
```

```java
package ejercicios;

public record EnteroCadena(Integer a, String s) {

    public static EnteroCadena of(Integer a, String s) {
        return new EnteroCadena(a, s);
    }

}
```

1.2. CÓDIGO TEST

```java
package tests;

import java.util.List;
import java.util.function.Function;

import ejercicios.Ejercicio1;
import us.lsi.common.Files2;

public class TestEjercicio1 {

    public static void main(String[] args) {

        // LECTURA DE FICHERO
        String rutaFichero = "ficheros/PI1Ej1DatosEntrada.txt";

        Function<String, TuplaTestEj1> parseTuplaEj1 = s -> {
            String[] ss = s.split(",");
            return TuplaTestEj1.of(Integer.valueOf(ss[0].trim()), ss[1].trim(),
                    Integer.valueOf(ss[2].trim()), ss[3].trim(),
                    Integer.valueOf(ss[4].trim()));
        };

        List<TuplaTestEj1> tuplas = Files2.streamFromFile(rutaFichero)
                .map(parseTuplaEj1)
                .toList();


        // TEST
        System.out.println("* TEST EJERCICIO 1 *");
        Integer i = 0;
        while(i<tuplas.size()) {
            System.out.println("------------------------");
            System.out.println(String.format("Test %d (funcional): %s", i+1, Ejercicio1.fFuncional(tuplas.get(i).a(), tuplas.get(i).b(),
                    tuplas.get(i).c(), tuplas.get(i).d(), tuplas.get(i).e())));
            System.out.println(String.format("Test %d (iterativa): %s", i+1, Ejercicio1.fIterativo(tuplas.get(i).a(), tuplas.get(i).b(),
                    tuplas.get(i).c(), tuplas.get(i).d(), tuplas.get(i).e())));
            System.out.println(String.format("Test %d (recursiva final): %s", i+1, Ejercicio1.fRecursivoFinal(tuplas.get(i).a(),
                    tuplas.get(i).b(), tuplas.get(i).c(), tuplas.get(i).d(), tuplas.get(i).e())));
            System.out.println("------------------------");
            i++;
        }
```

```java
package tests;

public record TuplaTestEj1(Integer a, String b, Integer c, String d, Integer e) {

    public static TuplaTestEj1 of(Integer a, String b, Integer c, String d, Integer e) {
        return new TuplaTestEj1(a, b, c, d, e);
    }

}
```

1.3. VOLCADO DE PANTALLA

```
* TEST EJERCICIO 1 *
-----------------------
Test 1 (funcional): {9=[vaeclipse], 10=[avaeclipse], 11=[javaeclipse]}
Test 1 (iterativa): {9=[vaeclipse], 10=[avaeclipse], 11=[javaeclipse]}
Test 1 (recursiva final): {9=[vaeclipse], 10=[avaeclipse], 11=[javaeclipse]}
-----------------------
-----------------------
Test 2 (funcional): {7=[12class], 11=[face12class], 13=[nterfaceclass], 14=[interfaceclass], 15=[nterface12class]}
Test 2 (iterativa): {7=[12class], 11=[face12class], 13=[nterfaceclass], 14=[interfaceclass], 15=[nterface12class]}
Test 2 (recursiva final): {7=[12class], 11=[face12class], 13=[nterfaceclass], 14=[interfaceclass], 15=[nterface12class]}
-----------------------
-----------------------
Test 3 (funcional): {10=[voidreturn, voidreturn]}
Test 3 (iterativa): {10=[voidreturn, voidreturn]}
Test 3 (recursiva final): {10=[voidreturn, voidreturn]}
-----------------------
-----------------------
Test 4 (funcional): {6=[rwhile, rwhile, 9while], 7=[r9while], 8=[forwhile]}
Test 4 (iterativa): {6=[rwhile, rwhile, 9while], 7=[r9while], 8=[forwhile]}
Test 4 (recursiva final): {6=[rwhile, rwhile, 9while], 7=[r9while], 8=[forwhile]}
-----------------------
-----------------------
Test 5 (funcional): {6=[ifelse, ifelse, ifelse, 24else], 8=[if24else]}
Test 5 (iterativa): {6=[ifelse, ifelse, ifelse, 24else], 8=[if24else]}
Test 5 (recursiva final): {6=[ifelse, ifelse, ifelse, 24else], 8=[if24else]}
-----------------------
-----------------------
Test 6 (funcional): {8=[15static], 10=[1521static], 12=[importstatic], 13=[mport15static], 14=[import15static]}
Test 6 (iterativa): {8=[15static], 10=[1521static], 12=[importstatic], 13=[mport15static], 14=[import15static]}
Test 6 (recursiva final): {8=[15static], 10=[1521static], 12=[importstatic], 13=[mport15static], 14=[import15static]}
-----------------------
```

2. EJERCICIO 2

2.1. CÓDIGO EJERCICIO

```java
package ejercicios;

import java.util.stream.Stream;

public class Ejercicio2 {

    // RECURSIVA NO FINAL
    public static Integer fRecursivoNoFinal(Integer a, Integer b, String s) {
        Integer res;
        if(s.length() == 0) {
            res = a*a + b*b;
        } else if(a<2 || b<2) {
            res = s.length() + a + b;
        } else if(a%s.length() < b%s.length()) {
            res = a + b + fRecursivoNoFinal(a-1, b/2, s.substring(a%s.length(), b%s.length()));
        } else {
            res = a * b + fRecursivoNoFinal(a/2, b-1, s.substring(b%s.length(), a%s.length()));
        }
        return res;
    }
```

```java
// RECURSIVA FINAL
public static Integer fRecursivoFinal(Integer a, Integer b, String s) {
    Integer res = 0;
    return fRecursivoFinal(res, a, b, s);
}

private static Integer fRecursivoFinal(Integer res, Integer a, Integer b, String s) {
    if(s.length() == 0) {
        res += a*a + b*b;
    } else if(a<2 || b<2) {
        res += s.length() + a + b;
    } else if(a%s.length() < b%s.length()) {
        res = fRecursivoFinal(res + a+b, a-1, b/2, s.substring(a%s.length(), b%s.length()));
    } else {
        res = fRecursivoFinal(res + a*b, a/2, b-1, s.substring(b%s.length(), a%s.length()));
    }
    return res;
}


// ITERATIVA
public static Integer fIterativo(Integer a, Integer b, String s) {
    Integer res = 0;
    while(!((s.length() == 0) || (a<2 || b<2))) {
        if(a%s.length() < b%s.length()) {
            res += a+b;
            s = s.substring(a%s.length(), b%s.length());
            a = a-1;
            b = b/2;
        } else {
            res += a*b;
            s = s.substring(b%s.length(), a%s.length());
            a = a/2;
            b = b-1;
        }
    }
    if(s.length() == 0) {
        res += a*a + b*b;
    } else if(a<2 || b<2) {
        res += s.length() + a+b;
    }
    return res;
}


// FUNCIONAL
public static Integer fFuncional(Integer a, Integer b, String s) {
    Integer res = 0;
    TuplaEj2 resTupla = Stream.iterate(TuplaEj2.semilla(a, b, s), x -> x.next())
            .filter(x -> x.esCasoBase())
            .findFirst()
            .get();
    if(resTupla.s().length() == 0) {
        res = resTupla.a()*resTupla.a() + resTupla.b()*resTupla.b() + resTupla.ac();
    } else if(a<2 || b<2) {
        res = resTupla.a()+resTupla.b()+resTupla.s().length() + resTupla.ac();
    }
    return res;
}
```

```java
package ejercicios;

public record TuplaEj2(Integer ac, Integer a, Integer b, String s) {

    public static TuplaEj2 of(Integer ac, Integer a, Integer b, String s) {
        return new TuplaEj2(ac, a, b, s);
    }

    public static TuplaEj2 semilla(Integer a, Integer b, String s) {
        return new TuplaEj2(0, a, b, s);
    }

    public TuplaEj2 next() {
        TuplaEj2 res;
        if(a%s.length() < b%s.length()) {
            res = of(ac + a+b, a-1, b/2,
                    s.substring(a%s.length(), b%s.length()));
        } else {
            res = of(ac + a*b, a/2, b-1,
                    s.substring(b%s.length(), a%s.length()));
        }
        return res;
    }

    public Boolean esCasoBase() {
        return s.length()==0 || (a<2 || b<2);
    }

}
```

2.2. CÓDIGO TEST

```java
package tests;

import java.util.List;
import java.util.function.Function;

import ejercicios.Ejercicio2;
import us.lsi.common.Files2;

public class TestEjercicio2 {

    public static void main(String[] args) {

        // LECTURA DE FICHERO
        String rutaFichero = "ficheros/PI1Ej2DatosEntrada.txt";

        Function<String, TuplaTestEj2> parseTuplaEj2 = s -> {
            String[] ss = s.split(",");
            return TuplaTestEj2.of(Integer.valueOf(ss[0].trim()),
                    Integer.valueOf(ss[1].trim()), ss[2].trim());
        };

        List<TuplaTestEj2> datosFichero = Files2.streamFromFile(rutaFichero)
                .map(parseTuplaEj2)
                .toList();

        // TEST
        System.out.println("* TEST EJERCICIO 2 *");
        Integer i= 0;
        while(i < datosFichero.size()) {
            TuplaTestEj2 tupla = datosFichero.get(i);
            System.out.println("------------------------");
            System.out.println(String.format("Test %d (recursivo no final): %s", i+1, Ejercicio2.fRecursivoNoFinal(
                    tupla.a(), tupla.b(), tupla.s())));
            System.out.println(String.format("Test %d (recursivo final): %s", i+1, Ejercicio2.fRecursivoFinal(
                    tupla.a(), tupla.b(), tupla.s())));
            System.out.println(String.format("Test %d (iterativo): %s", i+1, Ejercicio2.fIterativo(tupla.a(),
                    tupla.b(), tupla.s())));
            System.out.println(String.format("Test %d (funcional): %s", i+1, Ejercicio2.fFuncional(tupla.a(),
                    tupla.b(), tupla.s())));
            System.out.println("------------------------");
            i++;
        }
    }
```

```java
package tests;

public record TuplaTestEj2(Integer a, Integer b, String s) {

    public static TuplaTestEj2 of(Integer a, Integer b, String s) {
        return new TuplaTestEj2(a, b, s);
    }

}
```

2.3. VOLCADO DE PANTALLA

```
* TEST EJERCICIO 2 *
-----------------------
Test 1 (recursivo no final): 623
Test 1 (recursivo final): 623
Test 1 (iterativo): 623
Test 1 (funcional): 623
-----------------------
-----------------------
Test 2 (recursivo no final): 950
Test 2 (recursivo final): 950
Test 2 (iterativo): 950
Test 2 (funcional): 950
-----------------------
-----------------------
Test 3 (recursivo no final): 3278
Test 3 (recursivo final): 3278
Test 3 (iterativo): 3278
Test 3 (funcional): 3278
-----------------------
-----------------------
Test 4 (recursivo no final): 3135
Test 4 (recursivo final): 3135
Test 4 (iterativo): 3135
Test 4 (funcional): 3135
-----------------------
-----------------------
Test 5 (recursivo no final): 3810
Test 5 (recursivo final): 3810
Test 5 (iterativo): 3810
Test 5 (funcional): 3810
-----------------------
-----------------------
Test 6 (recursivo no final): 5553
Test 6 (recursivo final): 5553
Test 6 (iterativo): 5553
Test 6 (funcional): 5553
-----------------------
```

3. EJERCICIO 3

3.1. CÓDIGO EJERCICIO

```java
package ejercicios;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.stream.Stream;

import us.lsi.common.Files2;
import us.lsi.geometria.Punto2D;
import us.lsi.geometria.Punto2D.Cuadrante;

public class Ejercicio3 {

    // Función para pasar de String a Punto2D
    public static Punto2D parsePunto2D (String s) {
        String[] ss = s.split(",");
        return Punto2D.of(Double.valueOf(ss[0].trim()), Double.valueOf(ss[1].trim()));
    }


    // ITERATIVO
    public static List<Punto2D> fIterativo(String rutaFicheroA, String rutaFicheroB) {
        List<Punto2D> ac = new ArrayList<>();
        Iterator<String> it1 = Files2.streamFromFile(rutaFicheroA).iterator();
        Iterator<String> it2 = Files2.streamFromFile(rutaFicheroB).iterator();
        Punto2D p1 = parsePunto2D(it1.next());
        Punto2D p2 = parsePunto2D(it2.next());
        while(p1!=null || p2!=null) {
            if(p2==null || (p1!=null && p1.compareTo(p2)<0)) {
                if(p1.getCuadrante().equals(Cuadrante.PRIMER_CUADRANTE) ||
                        p1.getCuadrante().equals(Cuadrante.TERCER_CUADRANTE)) {
                    ac.add(p1);
                }
                p1 = it1.hasNext()? parsePunto2D(it1.next()):null;
            } else if(p1==null || (p2!=null && p2.compareTo(p1)<0)) {
                if(p2.getCuadrante().equals(Cuadrante.PRIMER_CUADRANTE) ||
                        p2.getCuadrante().equals(Cuadrante.TERCER_CUADRANTE)) {
                    ac.add(p2);
                }
                p2 = it2.hasNext()? parsePunto2D(it2.next()):null;
            }
        }
        return ac;
    }

    // RECURSIVO
    public static List<Punto2D> fRecursivo(String rutaFicheroA, String rutaFicheroB) {
        List<Punto2D> ac = new ArrayList<>();
        Iterator<String> it1 = Files2.streamFromFile(rutaFicheroA).iterator();
        Iterator<String> it2 = Files2.streamFromFile(rutaFicheroB).iterator();
        Punto2D p1 = parsePunto2D(it1.next());
        Punto2D p2 = parsePunto2D(it2.next());
        return fRecursivo(ac, it1, it2, p1, p2);
    }

    private static List<Punto2D> fRecursivo(List<Punto2D> ac, Iterator<String> it1,
            Iterator<String> it2, Punto2D p1, Punto2D p2) {
        if(p1!=null || p2!=null) {
            if(p2==null || (p1!=null && p1.compareTo(p2)<0)) {
                if(p1.getCuadrante().equals(Cuadrante.PRIMER_CUADRANTE) ||
                        p1.getCuadrante().equals(Cuadrante.TERCER_CUADRANTE)) {
                    ac.add(p1);
                }
                p1 = it1.hasNext()? parsePunto2D(it1.next()):null;
                fRecursivo(ac, it1, it2, p1, p2);
            } else if(p1==null || (p2!=null && p2.compareTo(p1)<0)) {
                if(p2.getCuadrante().equals(Cuadrante.PRIMER_CUADRANTE) ||
                        p2.getCuadrante().equals(Cuadrante.TERCER_CUADRANTE)) {
                    ac.add(p2);
                }
                p2 = it2.hasNext()? parsePunto2D(it2.next()):null;
                fRecursivo(ac, it1, it2, p1, p2);
            }
        }
        return ac;
    }
```

```java
    // FUNCIONAL
    public static List<Punto2D> fFuncional(String rutaFicheroA, String rutaFicheroB) {
        TuplaEj3 res = Stream.iterate(TuplaEj3.semilla(rutaFicheroA, rutaFicheroB), x -> x.next())
                .filter(x -> x.esCasoBase())
                .findFirst()
                .get();
        return res.ls();
    }
```

```java
package ejercicios;

import java.util.Iterator;
import java.util.List;

import us.lsi.common.Files2;
import us.lsi.common.List2;
import us.lsi.geometria.Punto2D;
import us.lsi.geometria.Punto2D.Cuadrante;

public record TuplaEj3(List<Punto2D> ls, Iterator<String> it1, Iterator<String> it2,
        Punto2D p1, Punto2D p2) {

    public static TuplaEj3 of(List<Punto2D> ls, Iterator<String> it1, Iterator<String> it2,
            Punto2D p1, Punto2D p2) {
        return new TuplaEj3(ls, it1, it2, p1, p2);
    }

    public static TuplaEj3 semilla(String rutaFicheroA, String rutaFocheroB) {
        Iterator<String> it1 = Files2.streamFromFile(rutaFicheroA).iterator();
        Iterator<String> it2 = Files2.streamFromFile(rutaFocheroB).iterator();
        Punto2D p1 = Ejercicio3.parsePunto2D(it1.next());
        Punto2D p2 = Ejercicio3.parsePunto2D(it2.next());
        return new TuplaEj3(List2.empty(), it1, it2, p1, p2);
    }
```

```java
    public static TuplaEj3 semilla(String rutaFicheroA, String rutaFocheroB) {
        Iterator<String> it1 = Files2.streamFromFile(rutaFicheroA).iterator();
        Iterator<String> it2 = Files2.streamFromFile(rutaFocheroB).iterator();
        Punto2D p1 = Ejercicio3.parsePunto2D(it1.next());
        Punto2D p2 = Ejercicio3.parsePunto2D(it2.next());
        return new TuplaEj3(List2.empty(), it1, it2, p1, p2);
    }

    public TuplaEj3 next() {
        Punto2D pt1 = p1;
        Punto2D pt2 = p2;
        if(p2==null || (p1!=null && p1.compareTo(p2)<0)) {
            if(p1.getCuadrante().equals(Cuadrante.PRIMER_CUADRANTE) ||
                    p1.getCuadrante().equals(Cuadrante.TERCER_CUADRANTE)) {
                ls.add(p1);
            }
            pt1 = it1.hasNext()? Ejercicio3.parsePunto2D(it1.next()):null;
        } else if(p1==null || (p2!=null && p2.compareTo(p1)<0)) {
            if(p2.getCuadrante().equals(Cuadrante.PRIMER_CUADRANTE) ||
                    p2.getCuadrante().equals(Cuadrante.TERCER_CUADRANTE)) {
                ls.add(p2);
            }
            pt2 = it2.hasNext()? Ejercicio3.parsePunto2D(it2.next()):null;
        }
        return of(ls, it1, it2, pt1, pt2);
    }

    public Boolean esCasoBase() {
        return p1==null && p2==null;
    }

}
```

3.2. CÓDIGO TEST

```
package tests;

import java.util.List;

import ejercicios.Ejercicio3;
import us.lsi.geometria.Punto2D;

public class TestEjercicio3 {

    // Función para mostrar los test por pantalla
    public static void fTest(List<Punto2D> ls) {
        Integer i = 0;
        while(i < ls.size()) {
            System.out.println(ls.get(i));
            i++;
        }
    }


    public static void main(String[] args) {
        // FICHEROS
        String rutaFichero1A = "ficheros/PI1Ej3DatosEntrada1A.txt";
        String rutaFichero1B = "ficheros/PI1Ej3DatosEntrada1B.txt";
        String rutaFichero2A = "ficheros/PI1Ej3DatosEntrada2A.txt";
        String rutaFichero2B = "ficheros/PI1Ej3DatosEntrada2B.txt";
        String rutaFichero3A = "ficheros/PI1Ej3DatosEntrada3A.txt";
        String rutaFichero3B = "ficheros/PI1Ej3DatosEntrada3B.txt";


        // TEST
        System.out.println("* TEST EJERCICIO 3 *");
        //
        System.out.println("-------------------------------");
        System.out.println("- Test 1 (iterativo): ");
        fTest(Ejercicio3.fIterativo(rutaFichero1A, rutaFichero1B));
        System.out.println("\n- Test 2 (iterativo): ");
        fTest(Ejercicio3.fIterativo(rutaFichero2A, rutaFichero2B));
        System.out.println("\n- Test 3 (iterativo): ");
        fTest(Ejercicio3.fIterativo(rutaFichero3A, rutaFichero3B));
        //
        System.out.println("-------------------------------");
        System.out.println("- Test 1 (recursivo): ");
        fTest(Ejercicio3.fRecursivo(rutaFichero1A, rutaFichero1B));
        System.out.println("\n- Test 2 (recursivo): ");
        fTest(Ejercicio3.fRecursivo(rutaFichero2A, rutaFichero2B));
        System.out.println("\n- Test 3 (recursivo): ");
        fTest(Ejercicio3.fRecursivo(rutaFichero3A, rutaFichero3B));

        System.out.println("-------------------------------");
        System.out.println("- Test 1 (funcional): ");
        fTest(Ejercicio3.fFuncional(rutaFichero1A, rutaFichero1B));
        System.out.println("\n- Test 2 (funcional): ");
        fTest(Ejercicio3.fFuncional(rutaFichero2A, rutaFichero2B));
        System.out.println("\n- Test 3 (funcional): ");
        fTest(Ejercicio3.fFuncional(rutaFichero3A, rutaFichero3B));
    }

}
```

3.3. VOLCADO DE PANTALLA

```
* TEST EJERCICIO 3 *
--------------------------------
- Test 1 (iterativo):
(-93.56,-33.78)
(-82.54,-58.64)
(-76.79,-30.38)
(-50.37,-54.07)
(-20.03,-99.54)
(-19.29,-25.9)
(-17.93,-20.26)
(24.02,68.2)
(39.87,48.37)
(45.29,97.59)

- Test 2 (iterativo):
(-82.35,-49.74)
(-74.69,-40.12)
(-72.94,-56.8)
(-65.53,-51.45)
(-48.56,-81.69)
(-47.56,-82.04)
(-37.99,-90.32)
(-36.56,-38.16)
(-8.3,-69.67)
(-6.82,-85.27)
(3.45,70.0)
(23.93,76.13)
(30.7,8.47)
(37.97,49.79)
(40.55,83.01)
(41.78,39.55)
(49.46,51.93)
(64.29,86.49)
(74.78,41.09)
(87.62,43.21)
```

```
- Test 3 (iterativo):
(-93.9,-6.76)
(-81.49,-23.61)
(-71.93,-51.44)
(-71.64,-24.87)
(-68.08,-8.76)
(-62.34,-38.53)
(-61.68,-1.78)
(-56.16,-41.49)
(-54.81,-26.67)
(-53.48,-50.98)
(-50.04,-96.54)
(-46.99,-83.11)
(-33.11,-92.17)
(-32.08,-66.57)
(-29.99,-72.32)
(-20.6,-8.85)
(-19.83,-5.01)
(-19.58,-94.75)
(-17.35,-76.96)
(-16.97,-96.8)
(-11.75,-13.63)
(0.42,13.94)
(9.07,33.36)
(10.69,95.3)
(14.7,82.66)
(15.68,26.66)
(16.33,54.0)
(16.78,55.2)
(28.38,81.47)
(28.91,91.34)
(35.75,38.79)
(45.23,56.37)
(45.41,82.21)
(47.42,41.06)
(53.42,66.34)
(55.06,57.38)
(58.08,11.18)
(60.16,59.96)
(60.68,8.38)
(65.54,70.44)
(68.32,23.46)
(78.6,69.48)
(79.09,80.75)
(79.3,62.79)
(79.76,69.36)
(84.74,31.62)
(86.21,86.12)
(87.89,49.68)
(90.47,25.64)
(96.34,83.99)
```

```
--------------------------------
- Test 1 (recursivo):
(-93.56,-33.78)
(-82.54,-58.64)
(-76.79,-30.38)
(-50.37,-54.07)
(-20.03,-99.54)
(-19.29,-25.9)
(-17.93,-20.26)
(24.02,68.2)
(39.87,48.37)
(45.29,97.59)

- Test 2 (recursivo):
(-82.35,-49.74)
(-74.69,-40.12)
(-72.94,-56.8)
(-65.53,-51.45)
(-48.56,-81.69)
(-47.56,-82.04)
(-37.99,-90.32)
(-36.56,-38.16)
(-8.3,-69.67)
(-6.82,-85.27)
(3.45,70.0)
(23.93,76.13)
(30.7,8.47)
(37.97,49.79)
(40.55,83.01)
(41.78,39.55)
(49.46,51.93)
(64.29,86.49)
(74.78,41.09)
(87.62,43.21)
```

```
- Test 3 (recursivo):
(-93.9,-6.76)
(-81.49,-23.61)
(-71.93,-51.44)
(-71.64,-24.87)
(-68.08,-8.76)
(-62.34,-38.53)
(-61.68,-1.78)
(-56.16,-41.49)
(-54.81,-26.67)
(-53.48,-50.98)
(-50.04,-96.54)
(-46.99,-83.11)
(-33.11,-92.17)
(-32.08,-66.57)
(-29.99,-72.32)
(-20.6,-8.85)
(-19.83,-5.01)
(-19.58,-94.75)
(-17.35,-76.96)
(-16.97,-96.8)
(-11.75,-13.63)
(0.42,13.94)
(9.07,33.36)
(10.69,95.3)
(14.7,82.66)
(15.68,26.66)
(16.33,54.0)
(16.78,55.2)
(28.38,81.47)
(28.91,91.34)
(35.75,38.79)
(45.23,56.37)
(45.41,82.21)
(47.42,41.06)
(53.42,66.34)
(55.06,57.38)
(58.08,11.18)
(60.16,59.96)
(60.68,8.38)
(65.54,70.44)
(68.32,23.46)
(78.6,69.48)
(79.09,80.75)
(79.3,62.79)
(79.76,69.36)
(84.74,31.62)
(86.21,86.12)
(87.89,49.68)
(90.47,25.64)
(96.34,83.99)
```

```
--------------------------------
- Test 1 (funcional):
(-93.56,-33.78)
(-82.54,-58.64)
(-76.79,-30.38)
(-50.37,-54.07)
(-20.03,-99.54)
(-19.29,-25.9)
(-17.93,-20.26)
(24.02,68.2)
(39.87,48.37)
(45.29,97.59)

- Test 2 (funcional):
(-82.35,-49.74)
(-74.69,-40.12)
(-72.94,-56.8)
(-65.53,-51.45)
(-48.56,-81.69)
(-47.56,-82.04)
(-37.99,-90.32)
(-36.56,-38.16)
(-8.3,-69.67)
(-6.82,-85.27)
(3.45,70.0)
(23.93,76.13)
(30.7,8.47)
(37.97,49.79)
(40.55,83.01)
(41.78,39.55)
(49.46,51.93)
(64.29,86.49)
(74.78,41.09)
(87.62,43.21)
```

```
- Test 3 (funcional):
(-93.9,-6.76)
(-81.49,-23.61)
(-71.93,-51.44)
(-71.64,-24.87)
(-68.08,-8.76)
(-62.34,-38.53)
(-61.68,-1.78)
(-56.16,-41.49)
(-54.81,-26.67)
(-53.48,-50.98)
(-50.04,-96.54)
(-46.99,-83.11)
(-33.11,-92.17)
(-32.08,-66.57)
(-29.99,-72.32)
(-20.6,-8.85)
(-19.83,-5.01)
(-19.58,-94.75)
(-17.35,-76.96)
(-16.97,-96.8)
(-11.75,-13.63)
(0.42,13.94)
(9.07,33.36)
(10.69,95.3)
(14.7,82.66)
(15.68,26.66)
(16.33,54.0)
(16.78,55.2)
(28.38,81.47)
(28.91,91.34)
(35.75,38.79)
(45.23,56.37)
(45.41,82.21)
(47.42,41.06)
(53.42,66.34)
(55.06,57.38)
(58.08,11.18)
(60.16,59.96)
(60.68,8.38)
(65.54,70.44)
(68.32,23.46)
(78.6,69.48)
(79.09,80.75)
(79.3,62.79)
(79.76,69.36)
(84.74,31.62)
(86.21,86.12)
(87.89,49.68)
(90.47,25.64)
(96.34,83.99)
```

4. EJERCICIO 4

4.1. CÓDIGO EJERCICIO

```java
package ejercicios;

import java.util.HashMap;
import java.util.Map;

import us.lsi.common.IntTrio;

public class Ejercicio4 {

    // RECURSIVO SIN MEMORIA
    public static String fRSM(Integer a, Integer b, Integer c) {
        String res;
        if(a<2 && b<=2 || c<2) {
            res = "(" + a.toString() + "+" + b.toString() + "+" + c.toString() + ")";
        } else if (a<3 || b<3 && c<=3) {
            res = "(" + c.toString() + "-" + b.toString() + "-" + a.toString() + ")";
        } else if(b%a == 0 && (a%2 == 0 || b%3 == 0)) {
            res = "(" + fRSM(a-1, b/a, c-1) + "*" + fRSM(a-2, b/2, c/2) + ")";
        } else {
            res = "(" + fRSM(a/2, b-2, c/2) + "/" + fRSM(a/3, b-1, c/3) + ")";
        }
        return res;
    }
```

```java
// RECURSIVO CON MEMORIA
public static String fRCM(Integer a, Integer b, Integer c) {
    Map<IntTrio, String> mp = new HashMap<>();
    return fRCM(a, b, c, mp);
}

private static String fRCM(Integer a, Integer b, Integer c, Map<IntTrio, String> mp) {
    String res = mp.get(IntTrio.of(a, b, c));
    if (res == null) {
        if(a<2 && b<=2 || c<2) {
            res = "(" + a.toString() + "+" + b.toString() + "+" + c.toString() + ")";
        } else if (a<3 || b<3 && c<=3) {
            res = "(" + c.toString() + "-" + b.toString() + "-" + a.toString() + ")";
        } else if(b%a == 0 && (a%2 == 0 || b%3 == 0)) {
            res = "(" + fRCM(a-1, b/a, c-1, mp) + "*" + fRCM(a-2, b/2, c/2, mp) + ")";
        } else {
            res = "(" + fRCM(a/2, b-2, c/2, mp) + "/" + fRCM(a/3, b-1, c/3, mp) + ")";
        }
        mp.put(IntTrio.of(a, b, c), res);
    }
    return res;
}

// ITERATIVO
public static String fIterativo(Integer a, Integer b, Integer c) {
    String res;
    Map<IntTrio, String> mp = new HashMap<>();
    for(int i=0; i<=a; i++) {
        for(int j=0; j<=b; j++) {
            for(int z=0; z<=c; z++) {
                if(i<2 && j<=2 || z<2) {
                    res = "(" + Integer.valueOf(i).toString() + "+" + Integer.valueOf(j).toString()
                            + "+" + Integer.valueOf(z).toString() + ")";
                } else if (i<3 || j<3 && z<=3) {
                    res = "(" + Integer.valueOf(z).toString() + "-" + Integer.valueOf(j).toString()
                            + "-" + Integer.valueOf(i).toString() + ")";
                } else if(j%i == 0 && (i%2 == 0 || j%3 == 0)) {
                    res = "(" + mp.get(IntTrio.of(i-1, j/i, z-1)) + "*" + mp.get(IntTrio.of(i-2, j/2, z/2)) + ")";
                } else {
                    res = "(" + mp.get(IntTrio.of(i/2, j-2, z/2)) + "/" + mp.get(IntTrio.of(i/3, j-1, z/3)) + ")";
                }
                mp.put(IntTrio.of(i, j, z), res);
            }
        }
    }
    return mp.get(IntTrio.of(a, b, c));
}
```

4.2. CÓDIGO TEST

```java
package tests;

import java.util.List;
import java.util.function.Function;

import ejercicios.Ejercicio4;
import us.lsi.common.Files2;
import us.lsi.common.IntTrio;

public class TestEjercicio4 {

    public static void main(String[] args) {

        // LECTURA FICHERO
        String rutaFichero = "ficheros/PI1Ej4DatosEntrada.txt";

        Function<String, IntTrio> parseTrioEnteros = s -> {
            String[] ss = s.split(",");
            return IntTrio.of(Integer.valueOf(ss[0].trim()),
                    Integer.valueOf(ss[1].trim()),
                    Integer.valueOf(ss[2].trim()));
        };

        List<IntTrio> triosEnteros = Files2.streamFromFile(rutaFichero)
                .map(parseTrioEnteros)
                .toList();
```

```java
        // TEST
        System.out.println("* TEST EJERCICIO 4 *");
        Integer i = 0;
        while(i < triosEnteros.size()) {
            System.out.println("------------------------");
            System.out.println(String.format("Test %d (Recursivo sin memoria): %s", i+1, Ejercicio4.fRSM(triosEnteros.get(i).first(),
                    triosEnteros.get(i).second(), triosEnteros.get(i).third())));
            System.out.println(String.format("Test %d (Recursivo con memoria): %s", i+1, Ejercicio4.fRCM(triosEnteros.get(i).first(),
                    triosEnteros.get(i).second(), triosEnteros.get(i).third())));
            System.out.println(String.format("Test %d (Iterativo): %s", i+1, Ejercicio4.fIterativo(triosEnteros.get(i).first(),
                    triosEnteros.get(i).second(), triosEnteros.get(i).third())));
            System.out.println("------------------------");
            i++;
        }

    }
```

4.3. VOLCADO DE PANTALLA

```
* TEST EJERCICIO 4 *
------------------------
Test 1 (Recursivo sin memoria): ((((3+14+1)/(2+15+0))/(5+17+1))/((5+17+1)/(3+18+1)))
Test 1 (Recursivo con memoria): ((((3+14+1)/(2+15+0))/(5+17+1))/((5+17+1)/(3+18+1)))
Test 1 (Iterativo): ((((3+14+1)/(2+15+0))/(5+17+1))/((5+17+1)/(3+18+1)))
------------------------
------------------------
Test 2 (Recursivo sin memoria): ((((2+24+1)/(1+25+0))/(3+27+1))/((3+27+1)/(2+28+1)))
Test 2 (Recursivo con memoria): ((((2+24+1)/(1+25+0))/(3+27+1))/((3+27+1)/(2+28+1)))
Test 2 (Iterativo): ((((2+24+1)/(1+25+0))/(3+27+1))/((3+27+1)/(2+28+1)))
------------------------
------------------------
Test 3 (Recursivo sin memoria): ((((3-4-2)/(2-5-1))/((2-5-1)/(1+6+1)))/(((2-5-1)/(1+6+1))/(3-8-2)))
Test 3 (Recursivo con memoria): ((((3-4-2)/(2-5-1))/((2-5-1)/(1+6+1)))/(((2-5-1)/(1+6+1))/(3-8-2)))
Test 3 (Iterativo): ((((3-4-2)/(2-5-1))/((2-5-1)/(1+6+1)))/(((2-5-1)/(1+6+1))/(3-8-2)))
------------------------
------------------------
Test 4 (Recursivo sin memoria): ((((2+9+1)/(1+10+0))/(3+12+1))/((3+12+1)/(2+13+1)))
Test 4 (Recursivo con memoria): ((((2+9+1)/(1+10+0))/(3+12+1))/((3+12+1)/(2+13+1)))
Test 4 (Iterativo): ((((2+9+1)/(1+10+0))/(3+12+1))/((3+12+1)/(2+13+1)))
------------------------
------------------------
Test 5 (Recursivo sin memoria): (((((2+22+1)/(1+23+0))/(3+25+1))/((3+25+1)/(2+26+1)))/(((3+25+1)/(2+26+1))/((3+7+1)*(2+14+1))))
Test 5 (Recursivo con memoria): (((((2+22+1)/(1+23+0))/(3+25+1))/((3+25+1)/(2+26+1)))/(((3+25+1)/(2+26+1))/((3+7+1)*(2+14+1))))
Test 5 (Iterativo): (((((2+22+1)/(1+23+0))/(3+25+1))/((3+25+1)/(2+26+1)))/(((3+25+1)/(2+26+1))/((3+7+1)*(2+14+1))))
------------------------
------------------------
Test 6 (Recursivo sin memoria): ((((((2+14+1)*(1+21+1))/(2+43+1))/(((2+7+1)/(1+8+0))*(3+22+1)))/((((2+7+1)/(1+8+0))*(3+22+1))/(((1+44+1)/(1+45+0))))/((((((2+7+1)/(1+8+0))*(3+22+1))/((1+44+
Test 6 (Recursivo con memoria): ((((((2+14+1)*(1+21+1))/(2+43+1))/(((2+7+1)/(1+8+0))*(3+22+1)))/((((2+7+1)/(1+8+0))*(3+22+1))/(((1+44+1)/(1+45+0))))/((((((2+7+1)/(1+8+0))*(3+22+1))/((1+44+
Test 6 (Iterativo): ((((((2+14+1)*(1+21+1))/(2+43+1))/(((2+7+1)/(1+8+0))*(3+22+1)))/((((2+7+1)/(1+8+0))*(3+22+1))/(((1+44+1)/(1+45+0))))/((((((2+7+1)/(1+8+0))*(3+22+1))/((1+44+1)/(1+45+0)))
------------------------

'((1+44+1)/(1+45+0)))/(((2+6+1)/(1+7+1))*((3+6+1)*(2+12+1)))))
'((1+44+1)/(1+45+0)))/(((2+6+1)/(1+7+1))*((3+6+1)*(2+12+1)))))
↓+45+0)))/(((2+6+1)/(1+7+1))*((3+6+1)*(2+12+1)))))
```