# PI1 – JUAN ORELLANA CARRETERO

## EJERCICIO 1

```
public class Ejercicio1 {

	public record EnteroCadena(Integer a, String s) {
		public static EnteroCadena of(Integer a, String s) {
			return new EnteroCadena(a, s);
		}
		public EnteroCadena nx() {
			EnteroCadena tupla = null;

			if(a()%3==0) {
				tupla = of(a()+2, s()+a().toString());
			}else {
				tupla = of(a()+2, 
s().substring(a()%s().length()));
			}
			return tupla;
		}
	}

	public static Map<Integer,List<String>> ejercicio1 (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));
			}



	public static Map<Integer, List<String>> 
ejercicio1Iterativo(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 p = e.s()+varD;
			if(p.length()<varE) {

				Integer key = p.length();
				if(ac.containsKey(key)) {
					List<String> ls = ac.get(key);
					ls.add(p);
					ac.put(key, ls);
```

```
                        }else {
                              List<String> ls = new ArrayList<>();
                              ls.add(p);
                              ac.put(key, ls);
                        }
                  }
                  e = e.nx();
            }
            return ac;
      }


      public static Map<Integer, List<String>>
ejercicio1Recursivo(Integer varA, String varB, Integer varC, String
                  varD, Integer varE){
            return ejercicio1RecursivoAux(varA, varB, varC, varD,
varE, new HashMap<>() );
      }

      private static Map<Integer, List<String>>
ejercicio1RecursivoAux(Integer varA, String varB, Integer varC,
                  String varD, Integer varE, Map<Integer, List<String>>
ac) {

            EnteroCadena e = EnteroCadena.of(varA, varB);

            if(e.a()<varC) {
                  String p = e.s()+varD;
                  if(p.length()<varE) {
                        Integer key = p.length();
                        if(ac.containsKey(key)) {
                              List<String> ls = ac.get(key);
                              ls.add(p);
                              ac.put(key, ls);
                              e = e.nx();
                              return ejercicio1RecursivoAux(e.a(),
e.s(), varC, varD, varE, ac);
                        }else {
                              List<String> ls = new ArrayList<>();
                              ls.add(p);
                              ac.put(key, ls);
                              e = e.nx();
                              return ejercicio1RecursivoAux(e.a(),
e.s(), varC, varD, varE, ac);
                        }
                  }

            }
            return ac;
      }
}
```

## EJERCICIO 2

```
public class Ejercicio2 {

	public static Integer ejercicio2RecursivoNoFinal(Integer a,
Integer b, String s) {
		return ejercicio2RecursivoNoFinalAux(a, b, s);
	}

	public static Integer ejercicio2RecursivoNoFinalAux(Integer a,
Integer b, String s) {
		Integer r = null;
		if(s.length()==0) {
			r = (a*a) + (b*b);
		}else if(a<2||b<2) {
			r = s.length()+a+b;
		}else if(a%s.length() < b%s.length()) {
			r = a + b + ejercicio2RecursivoNoFinalAux(a-1, b/2,
s.substring(a%s.length(), b%s.length()));
		}else {
			r = a * b + ejercicio2RecursivoNoFinalAux(a/2, b-1 ,
s.substring(b%s.length(), a%s.length()));
		}
		return r;
	}

	public static Integer ejercicio2RecursivoFinal(Integer a,
Integer b, String s) {
		return recFinal(a, b, s, 0);
	}

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


	public static Integer ejercicio2Iterativo(Integer a, Integer b,
String s) {
		Integer r = 0;
		while(!(s.length()==0) ||(!(a<2||b<2)) ) {
		if(a%s.length() < b%s.length()) {
			//r = recFinal(a-1, b/2, s.substring(a%s.length(),
b%s.length()), a+b);
			r = r + (a+b);
			a = a-1;
			b = b/2;
			s = s + s.substring(a%s.length(), b%s.length());
		}else {
			//r = recFinal(a/2, b-1 , s.substring(b%s.length(), 
a%s.length()), a*b);
			r = r + a*b;
			a /= 2;
			b -=1 ;
			s = s.substring(b%s.length(), a%s.length());

			}
		}
		if(s.length()==0) {
			r = r + (a*a) + (b*b);
		}else {
			r = r+s.length()+a+b;
		}
		return r;
	}

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

	public static Tupla first(Integer a, Integer b, String s) {
		return of(0, a, b, s);
	}

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

	public static Integer ejercicio2Funcional(Integer a, Integer b, 
String s) {
		Integer r = 0;

		Tupla t = Stream.iterate(Tupla.first(a,b,s), e->e.next())
				.filter(e -> e.s().length()==0 || 
(e.a()<2||e.b()<2))
				.findFirst()
				.get();

		if(t.s().length()==0) {
			r = t.ac() + (t.a()*t.a()) + (t.b()*t.b());
		}else {
			r = t.ac() + (t.s().length()+t.a()+t.b());
		}
		return r;
	}

	}
```

**EJERCICIO 3**

```
public class Ejercicio3 {


      private static Punto2D parsePunto(String s) {
            String[] v = s.split(",");
            Double x = Double.valueOf(v[0]);
            Double y = Double.valueOf(v[1]);
            return Punto2D.of(x,y);
      }

      public static List<Punto2D> ejercicio3Iterativo(String file1,
                  String file2, Comparator<Punto2D> comp){

                  List<Punto2D> ls = new ArrayList<>();

                  Iterator<String> it1 = new IteratorFile(file1);
                  Iterator<String> it2 = new IteratorFile(file2);

                  Punto2D e1 =
(it1.hasNext()?parsePunto(it1.next()):null);
                  Punto2D e2 =
(it2.hasNext()?parsePunto(it2.next()):null);

                  while(it1.hasNext()||it2.hasNext()) {
                        if(e1 != null && e2 != null) {
                              if(e1.compareTo(e2) < 0) {
                                    if(e1.getCuadrante() ==
Cuadrante.PRIMER_CUADRANTE || e1.getCuadrante() ==
Cuadrante.TERCER_CUADRANTE) {
                                          ls.add(e1);
                                    }
                                    e1 =
it1.hasNext()?parsePunto(it1.next()):null;

                              } else {
                                    if(e2.getCuadrante() ==
Cuadrante.PRIMER_CUADRANTE || e2.getCuadrante() ==
Cuadrante.TERCER_CUADRANTE) {
                                          ls.add(e2);
                                    }
                                    e2 =
it2.hasNext()?parsePunto(it2.next()):null;

                              }

                  }else if(e1==null) {

      if((e2.getCuadrante()==Cuadrante.PRIMER_CUADRANTE||

      e2.getCuadrante()==Cuadrante.TERCER_CUADRANTE)) {
                                    ls.add(e2);
                              }
                              e2 =
it2.hasNext()?parsePunto(it2.next()):null;

                  }else {

      if((e1.getCuadrante()==Cuadrante.PRIMER_CUADRANTE||
      e1.getCuadrante()==Cuadrante.TERCER_CUADRANTE)) {
                        ls.add(e1);
                        }
                        e1 = 
(it1.hasNext()?parsePunto(it1.next()):null);

                  }

            }
                  return ls;
      }



      public static List<Punto2D> ejercicio3RecursivoFinal(String 
file1,
                  String file2, Comparator<Punto2D> comp){

            List<Punto2D> ls = new ArrayList<>();
            Iterator<String> it1 = new IteratorFile(file1);
            Iterator<String> it2 = new IteratorFile(file2);

            Punto2D e1 = (it1.hasNext()?parsePunto(it1.next()):null);
            Punto2D e2 = (it2.hasNext()?parsePunto(it2.next()):null);

            return ejercicio3RecursivoFinalAux(it1, it2, comp, ls, e1
,e2);
      }


      private static List<Punto2D> 
ejercicio3RecursivoFinalAux(Iterator<String> it1, Iterator<String> 
it2, Comparator<Punto2D> comp,
                  List<Punto2D> ls, Punto2D e1, Punto2D e2) {
            if(it1.hasNext()||it2.hasNext()) {
                  if(e1 != null && e2 != null) {
                        if(e1.compareTo(e2) < 0) {
                              if(e1.getCuadrante() == 
Cuadrante.PRIMER_CUADRANTE || e1.getCuadrante() == 
Cuadrante.TERCER_CUADRANTE) {
                                    ls.add(e1);
                              }
                              e1 = 
it1.hasNext()?parsePunto(it1.next()):null;
                              ls = ejercicio3RecursivoFinalAux(it1, 
it2, comp, ls, e1, e2);

                        } else {
                              if(e2.getCuadrante() == 
Cuadrante.PRIMER_CUADRANTE || e2.getCuadrante() == 
Cuadrante.TERCER_CUADRANTE) {
                                    ls.add(e2);
                              }
                              e2 = 
it2.hasNext()?parsePunto(it2.next()):null;
                              ls = ejercicio3RecursivoFinalAux(it1, 
it2, comp, ls, e1, e2);

                        }
                        }
                        else if(e1==null) {

        if((e2.getCuadrante()==Cuadrante.PRIMER_CUADRANTE||

        e2.getCuadrante()==Cuadrante.TERCER_CUADRANTE)) {
                                        ls.add(e2);
                                }
                                e2 = it2.hasNext()?parsePunto(it2.next()):null;
                                ls = ejercicio3RecursivoFinalAux(it1, it2,
comp, ls, e1, e2);

                        }else {

        if((e1.getCuadrante()==Cuadrante.PRIMER_CUADRANTE||

        e1.getCuadrante()==Cuadrante.TERCER_CUADRANTE)) {
                                ls.add(e1);
                                }
                                e1 =
(it1.hasNext()?parsePunto(it1.next()):null);
                                ls = ejercicio3RecursivoFinalAux(it1, it2,
comp, ls, e1, e2);
                        }
                }


                return ls;

        }

        public static record Tupla(List<Punto2D> ac,Iterator<String>
it1,
                        Iterator<String> it2, Comparator<Punto2D> comp,
Punto2D e1, Punto2D e2) {

                public static Tupla of(List<Punto2D> ac,Iterator<String>
it1,
                                Iterator<String> it2, Comparator<Punto2D> comp,
Punto2D e1, Punto2D e2) {
                        return new Tupla(ac, it1, it2, comp, e1, e2);
                }


                public Tupla next() {
                        Tupla res = null;
                        Punto2D e = null;
                        if(e1 != null && e2 != null) {
                                if(e1.compareTo(e2) < 0) {
                                        if(e1.getCuadrante() ==
Cuadrante.PRIMER_CUADRANTE || e1.getCuadrante() ==
Cuadrante.TERCER_CUADRANTE) {
                                                ac.add(e1);
                                        }
                                        e =
it1.hasNext()?parsePunto(it1.next()):null;
                                        res = of(ac, it1, it2, comp, e, e2);

                                } else {
```

```
                                if(e2.getCuadrante() ==
Cuadrante.PRIMER_CUADRANTE || e2.getCuadrante() ==
Cuadrante.TERCER_CUADRANTE) {
                                      ac.add(e2);
                                }
                                e =
it2.hasNext()?parsePunto(it2.next()):null;
                                res = of(ac, it1, it2, comp, e1, e);

                          }

                   }else if(e1==null) {

      if((e2.getCuadrante()==Cuadrante.PRIMER_CUADRANTE||

      e2.getCuadrante()==Cuadrante.TERCER_CUADRANTE)) {
                                ac.add(e2);
                          }else {
                          e = it2.hasNext()?parsePunto(it2.next()):null;
                          }
                          res = of(ac, it1, it2, comp, e1, e);
                   }else {
                         if((e1.getCuadrante()==Cuadrante.PRIMER_CUADRANTE||

      e1.getCuadrante()==Cuadrante.TERCER_CUADRANTE)) {
                         ac.add(e1);
                         }else {
                         e = (it1.hasNext()?parsePunto(it1.next()):null);
                   }
                   res = of(ac, it1, it2, comp, e, e2);
                   }
                         return res;
                   }
      }


      public static List<Punto2D> ejercicio3Funcional(String file1,
String file2, Comparator<Punto2D> comp) {

            Iterator<String> it1 = new IteratorFile(file1);
            Iterator<String> it2 = new IteratorFile(file2);

            Punto2D e1 = (it1.hasNext()?parsePunto(it1.next()):null);
            Punto2D e2 = (it2.hasNext()?parsePunto(it2.next()):null);

            List<Punto2D> ls = Stream.iterate(Tupla.of(null, it1, it2,
comp, e1, e2), e->e.next())
                        .filter(e -> (e.e1() == null && e.e2() ==
null))
                        .findFirst()
                        .get().ac();
            return ls;

      }
}
```

## EJERCICIO 4

```
public class Ejercicio4 {

      public static String ejercicio4RecSinMemoria(Integer a, Integer
b, Integer c) {
            String r = null;
            if(a<2 && b<=2 || c<2) {
                  r = String.format("(%d+%d+%d)",a,b,c );
            }else if(a<3 || b<3 && c<=3) {
                  r = String.format("(%d-%d-%d)", c,b,a);
            }else if(b%a == 0 && (a%2==0 || b%3==0)) {
                  r = "(" + ejercicio4RecSinMemoria(a-1, b/a, c-1) +
"*" + ejercicio4RecSinMemoria(a/3, b/2, c/2) + ")";
            }else {
                  r = "(" + ejercicio4RecSinMemoria(a/2, b-2, c/2) +
"/" + ejercicio4RecSinMemoria(a/3, b-1, c/3) + ")";
            }
            return r;
      }



      public static String ejercicio4ConMemoria(Integer a, Integer b,
Integer c) {
            return rec4Mem(a, b, c, new HashMap<>());
      }

      private static String rec4Mem(Integer a, Integer b, Integer c,
Map<IntTrio, String> m) {
            String r = null;
            IntTrio key = IntTrio.of(a, b, c);
            if(m.containsKey(key)) {
                  r = m.get(key);
            }else {
                  if(a<2 && b<=2 || c<2) {
                        r = String.format("(%d+%d+%d)",a,b,c );
                  }else if(a<3 || b<3 && c<=3) {
                        r = String.format("(%d-%d-%d)", c,b,a);
                  }else if(b%a == 0 && (a%2==0 || b%3==0)) {
                        r = "(" + ejercicio4RecSinMemoria(a-1, b/a, c-
1) + "*" + ejercicio4RecSinMemoria(a/3, b/2, c/2) + ")";
                  }else {
                        r = "(" + ejercicio4RecSinMemoria(a/2, b-2,
c/2) + "/" + ejercicio4RecSinMemoria(a/3, b-1, c/3) + ")";
                  }
                  m.put(key, r);
            }
            return r;
      }


      public static String ejercicio4Iterativo(Integer a, Integer b,
Integer c) {
            Map<IntTrio, String> m = new HashMap<>();
            String r = "";
            for(int i=0; i<=a; i++) {
                  for(int j=0; j<=b; j++) {
                        for(int k=0; k<=c; k++) {
                              if(i<2 && j<=2 || k<2) {
```

```
                                                r =
String.format("(%d+%d+%d)",i,j,k );
                                        }else if(i<3 || j<3 && k<=3) {
                                                r = String.format("(%d-%d-%d)",
k,j,i);
                                        }else if(j%i == 0 && (i%2==0 || j%3==0))
{
                                                r = "(" + m.get(IntTrio.of(i-1,
j/i, k-1)) + "*" + m.get(IntTrio.of(i/3, j/2, k/2)) + ")";
                                        }else {
                                                r = "(" + m.get(IntTrio.of(i/2, j-
2, k/2)) + "/" + m.get(IntTrio.of(i/3, j-1, k/3)) + ")";
                                        }
                                        m.put(IntTrio.of(i,j,k), r);
                                }
                        }
                }
                return m.get(IntTrio.of(a,b,c));

        }

}
```

## TEST DE LOS EJERCICIOS

```
public class TestEjercicios {

      public static void main(String[] args) {


            testEjercicio1();


            testEjercicio2();


            testEjercicio3();


            testEjercicio4();
      }




//////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////
      public static record entradaFicheroEj1(Integer varA, String
varB, Integer varC, String
                  varD, Integer varE) {
            public static entradaFicheroEj1 of(Integer varA, String
varB, Integer varC, String
                        varD, Integer varE) {
                  return new entradaFicheroEj1(varA, varB, varC, varD,
varE);
            }

            private static entradaFicheroEj1 parseLinea(String linea) {
                  String [] splitted = linea.split(",");
                  return
of(Integer.valueOf(splitted[0]),splitted[1].toString(),
                              Integer.valueOf(splitted[2]),
splitted[3].toString(), Integer.valueOf(splitted[4]));

      }

}
      public static void testEjercicio1() {
            String file = "ficheros/PI1Ej1DatosEntrada.txt";

            List<String> lineas = Files2.linesFromFile(file);
            List<entradaFicheroEj1> l = lineas.stream().map(linea ->
entradaFicheroEj1.parseLinea(linea)).toList();
            System.out.println("\n EJERCICIO 1");

      System.out.println("\n###########################################
######");
            System.out.println("\n Ejercicio 1 Iterativo");
            l.forEach(d->{

      System.out.println(Ejercicio1.ejercicio1Iterativo(d.varA(),
d.varB(), d.varC(), d.varD(), d.varE()));

            });
```

```
	System.out.println("\n#############################################
######");
		System.out.println("\n Ejercicio 1 Recursivo");
		l.forEach(d->{
	
	System.out.println(Ejercicio1.ejercicio1Recursivo(d.varA(), 
d.varB(), d.varC(), d.varD(), d.varE()));
	
		});
	
	}


///////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////



	public static record entradaFicheroEj2(Integer a, Integer b, 
String c) {
		public static entradaFicheroEj2 of(Integer a, Integer b, 
String c) {
			return new entradaFicheroEj2(a, b, c);
		}
		
		private static entradaFicheroEj2 parseLinea(String linea) {
			String [] splitted = linea.split(",");
			return of(Integer.valueOf(splitted[0]), 
Integer.valueOf(splitted[1]), splitted[2].toString());
	
	}

}

	public static void testEjercicio2() {
		String file = "ficheros/PI1Ej2DatosEntrada.txt";
		List<String> lineas = Files2.linesFromFile(file);
		List<entradaFicheroEj2> l = lineas.stream().map(linea -> 
entradaFicheroEj2.parseLinea(linea)).toList();
		System.out.println("\n EJERCICIO 2");
	
	System.out.println("\n#############################################
######");
		System.out.println("\n Ejercicio 2 Recursivo No final");
		l.forEach(d->{
	
	System.out.println(Ejercicio2.ejercicio2RecursivoNoFinal(d.a(), 
d.b(), d.c()));
		});
	
	System.out.println("\n#############################################
######");
		System.out.println("\n Ejercicio 2 Recursivo Final");
		l.forEach(d->{
	
	System.out.println(Ejercicio2.ejercicio2RecursivoFinal(d.a(), 
d.b(), d.c()));
		});
```

```
      System.out.println("\n##########################################
######");
            System.out.println("\n Ejercicio 2 Iterativo");
            l.forEach(d->{

      System.out.println(Ejercicio2.ejercicio2RecursivoFinal(d.a(),
d.b(), d.c()));
            });

      System.out.println("\n##########################################
######");
            System.out.println("\n Ejercicio 2 Funcional");
            l.forEach(d->{

      System.out.println(Ejercicio2.ejercicio2Funcional(d.a(), d.b(),
d.c()));
            });
      }


      public static void testEjercicio3() {
            String file1 = "ficheros/PI1Ej3DatosEntrada1A.txt";
            String file2 = "ficheros/PI1Ej3DatosEntrada1B.txt";
            String file3 = "ficheros/PI1Ej3DatosEntrada2A.txt";
            String file4 = "ficheros/PI1Ej3DatosEntrada2B.txt";
            String file5 = "ficheros/PI1Ej3DatosEntrada3A.txt";
            String file6 = "ficheros/PI1Ej3DatosEntrada3B.txt";
            System.out.println("\n EJERCICIO 3");

      System.out.println("\n##########################################
######");
            System.out.println("\n Solucion iterativa");
            System.out.println("1) Iterativo Ficheros 1 y 2: \n" +
Ejercicio3.ejercicio3Iterativo(file1, file2,
Comparator.naturalOrder()));
            System.out.println("2) Iterativo Ficheros 3 y 4: \n" +
Ejercicio3.ejercicio3Iterativo(file3, file4,
Comparator.naturalOrder()));
            System.out.println("3) Iterativo Ficheros 5 y 6: \n" +
Ejercicio3.ejercicio3Iterativo(file5, file6,
Comparator.naturalOrder()));

      System.out.println("\n##########################################
######");
            System.out.println("\n Solucion recursiva final");
            System.out.println("1) Recursiva Final Ficheros 1 y 2: \n"
+ Ejercicio3.ejercicio3RecursivoFinal(file1, file2,
Comparator.naturalOrder()));
            System.out.println("2) Recursiva Final Ficheros 3 y 4: \n"
+ Ejercicio3.ejercicio3RecursivoFinal(file3, file4,
Comparator.naturalOrder()));
            System.out.println("3) Recursiva Final Ficheros 5 y 6: \n"
+ Ejercicio3.ejercicio3RecursivoFinal(file5, file6,
Comparator.naturalOrder()));

      System.out.println("\n##########################################
######");
            System.out.println("\n Solucion funcional");
```

```
		System.out.println("1) Funcional Ficheros 1 y 2: \n" + 
Ejercicio3.ejercicio3RecursivoFinal(file1, file2, 
Comparator.naturalOrder()));
		System.out.println("2) Funcional Ficheros 3 y 4: \n" + 
Ejercicio3.ejercicio3RecursivoFinal(file3, file4, 
Comparator.naturalOrder()));
		System.out.println("3) Funcional Ficheros 5 y 6: \n" + 
Ejercicio3.ejercicio3RecursivoFinal(file5, file6, 
Comparator.naturalOrder()));
	}


	public static record entradaFicheroEj4(Integer a, Integer b, 
Integer c) {
		public static entradaFicheroEj4 of(Integer a, Integer b, 
Integer c) {
			return new entradaFicheroEj4(a, b, c);
		}

		private static entradaFicheroEj4 parseLinea(String linea) {
			String [] splitted = linea.split(",");
			return of(Integer.valueOf(splitted[0]), 
Integer.valueOf(splitted[1]), Integer.valueOf(splitted[2]));

	}
}
	public static void testEjercicio4() {
		String file = "ficheros/PI1Ej4DatosEntrada.txt";
		List<String> lineas = Files2.linesFromFile(file);
		List<entradaFicheroEj4> l = lineas.stream().map(linea -> 
entradaFicheroEj4.parseLinea(linea)).toList();
		System.out.println("\n EJERCICIO 4");
		
	System.out.println("\n############################################
######");
		System.out.println("\n Ejercicio Iterativo");
		l.forEach(d->{
			
	System.out.println(Ejercicio4.ejercicio4Iterativo(d.a(), d.b(), 
d.c()));
		});
		
	System.out.println("\n############################################
######");
		System.out.println("\n Ejercicio Recursivo Sin Memoria");
		l.forEach(d->{
			
	System.out.println(Ejercicio4.ejercicio4RecSinMemoria(d.a(), 
d.b(), d.c()));
			});
		
	System.out.println("\n############################################
######");
		System.out.println("\n Ejercicio Recursivo Sin Memoria");
		l.forEach(d->{
			
	System.out.println(Ejercicio4.ejercicio4ConMemoria(d.a(), d.b(), 
d.c()));
			});
	}
}
```

## VOLCADOS EN PANTALLA

```
 EJERCICIO 1

################################################

 Ejercicio 1 Iterativo
{9=[vaeclipse], 10=[avaeclipse], 11=[javaeclipse]}
{7=[12class], 11=[face12class], 13=[nterfaceclass], 14=[interfaceclass], 15=[nterface12class]}
{10=[voidreturn, voidreturn]}
{6=[rwhile, rwhile, 9while], 7=[r9while], 8=[forwhile]}
{6=[ifelse, ifelse, ifelse, 24else], 8=[if24else]}
{8=[15static], 10=[1521static], 12=[importstatic], 13=[mport15static], 14=[import15static]}

################################################

 Ejercicio 1 Recursivo
{9=[vaeclipse], 10=[avaeclipse], 11=[javaeclipse]}
{7=[12class], 11=[face12class], 13=[nterfaceclass], 14=[interfaceclass], 15=[nterface12class]}
{10=[voidreturn, voidreturn]}
{6=[rwhile, rwhile, 9while], 7=[r9while], 8=[forwhile]}
{6=[ifelse, ifelse, ifelse, 24else], 8=[if24else]}
{8=[15static], 10=[1521static], 12=[importstatic], 13=[mport15static], 14=[import15static]}
```

```
 EJERCICIO 2

################################################

 Ejercicio 2 Recursivo No final
623
950
3278
3135
3810
5553

################################################

 Ejercicio 2 Recursivo Final
623
950
3278
3135
3810
5553

################################################

 Ejercicio 2 Iterativo
623
950
3278
3135
3810
5553

################################################

 Ejercicio 2 Funcional
623
950
3278
3135
3810
5553
```

```
 EJERCICIO 3

################################################

 Solucion iterativa
1) Iterativo Ficheros 1 y 2:
[(-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)]
2) Iterativo Ficheros 3 y 4:
[(-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)]
3) Iterativo Ficheros 5 y 6:
[(-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)]

################################################

 Solucion recursiva final
1) Recursiva Final Ficheros 1 y 2:
[(-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)]
2) Recursiva Final Ficheros 3 y 4:
[(-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)]
3) Recursiva Final Ficheros 5 y 6:
[(-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)]

################################################

 Solucion funcional
```

```
1) Funcional Ficheros 1 y 2:
[(-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)]
2) Funcional Ficheros 3 y 4:
[(-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)]
3) Funcional Ficheros 5 y 6:
[(-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)]


EJERCICIO 4

##################################################

 Ejercicio Iterativo
((((3+14+1)/(2+15+0))/(5+17+1))/((5+17+1)/(3+18+1)))
((((2+24+1)/(1+25+0))/(3+27+1))/((3+27+1)/(2+28+1)))
((((3-4-2)/(2-5-1))/((2-5-1)/(1+6+1)))/(((2-5-1)/(1+6+1))/(3-8-2)))
((((2+9+1)/(1+10+0))/(3+12+1))/((3+12+1)/(2+13+1)))
(((((2+22+1)/(1+23+0))/(3+25+1))/((3+25+1)/(2+26+1)))/(((3+25+1)/(2+26
+1))/((3+7+1)*(1+14+1))))
((((((2+14+1)*(1+21+1))/(2+43+1))/(((2+7+1)/(1+8+0))*(1+22+1)))/((((2+
7+1)/(1+8+0))*(1+22+1))/((1+44+1)/(1+45+0))))/(((((2+7+1)/(1+8+0))*(1+
22+1))/((1+44+1)/(1+45+0)))/(((2+6+1)/(1+7+1))*(2-24-2))))

##################################################

 Ejercicio Recursivo Sin Memoria
((((3+14+1)/(2+15+0))/(5+17+1))/((5+17+1)/(3+18+1)))
((((2+24+1)/(1+25+0))/(3+27+1))/((3+27+1)/(2+28+1)))
((((3-4-2)/(2-5-1))/((2-5-1)/(1+6+1)))/(((2-5-1)/(1+6+1))/(3-8-2)))
((((2+9+1)/(1+10+0))/(3+12+1))/((3+12+1)/(2+13+1)))
(((((2+22+1)/(1+23+0))/(3+25+1))/((3+25+1)/(2+26+1)))/(((3+25+1)/(2+26
+1))/((3+7+1)*(1+14+1))))
((((((2+14+1)*(1+21+1))/(2+43+1))/(((2+7+1)/(1+8+0))*(1+22+1)))/((((2+
7+1)/(1+8+0))*(1+22+1))/((1+44+1)/(1+45+0))))/(((((2+7+1)/(1+8+0))*(1+
22+1))/((1+44+1)/(1+45+0)))/(((2+6+1)/(1+7+1))*(2-24-2))))

##################################################

 Ejercicio Recursivo Sin Memoria
((((3+14+1)/(2+15+0))/(5+17+1))/((5+17+1)/(3+18+1)))
((((2+24+1)/(1+25+0))/(3+27+1))/((3+27+1)/(2+28+1)))
((((3-4-2)/(2-5-1))/((2-5-1)/(1+6+1)))/(((2-5-1)/(1+6+1))/(3-8-2)))
((((2+9+1)/(1+10+0))/(3+12+1))/((3+12+1)/(2+13+1)))
```

```
(((((2+22+1)/(1+23+0))/(3+25+1))/((3+25+1)/(2+26+1)))/(((3+25+1)/(2+26
+1))/((3+7+1)*(1+14+1))))
((((((2+14+1)*(1+21+1))/(2+43+1))/(((2+7+1)/(1+8+0))*(1+22+1)))/((((2+
7+1)/(1+8+0))*(1+22+1))/((1+44+1)/(1+45+0))))/(((((2+7+1)/(1+8+0))*(1+
22+1))/((1+44+1)/(1+45+0)))/(((2+6+1)/(1+7+1))*(2-24-2))))
```