Ejercicio 1

```
	//iterativa
	public static boolean ejercicio1It(List<String> ls, Predicate<String> pS,
			Predicate<Integer> pI, Function<String,Integer> f) {
		boolean ac= false;
		int i = 0;
		while(i<ls.size()) {
			if(pS.test(ls.get(i)) == true) {
				ac = ac || pI.test(f.apply(ls.get(i)));
			}
					i++;
		}


		return ac;
	}
	//recursiva final
	public static boolean ejercicio1RF(List<String> ls, Predicate<String> pS,
			Predicate<Integer> pI, Function<String,Integer> f) {


		return ejercicio1RFAUX(ls,pS,pI,f, 0 ,false);
	}
	public static boolean ejercicio1RFAUX(List<String> ls, Predicate<String> pS,
Predicate<Integer> pI,
			Function<String, Integer> f,Integer i, boolean ac) {
		// TODO Auto-generated method stub
		if(i<ls.size()) {
		if(pS.test(ls.get(i)) == true && pI.test(ls.get(i).length()) == true)
{
			i++;
			return ejercicio1RFAUX(ls,pS,pI,f, i, true);
		}else {
			i++;
			return ejercicio1RFAUX(ls,pS,pI,f, i, ac);
			}
		}else {
			return ac;
		}

	}
	//funcional
	public static boolean ejercicio1F(List<String> ls, Predicate<String> pS,
			Predicate<Integer> pI, Function<String,Integer> f){
			return ls.stream()
			.filter(pS)
			.map(f)
			.anyMatch(pI);

		}
```

Ejercicio 2

```
public static Map<Integer,List<String>> ejercicio2 (List<List<String>> listas) {
		return listas.stream()
		.flatMap(lista ->
lista.stream()).collect(Collectors.groupingBy(String::length));

	}
	//iterativa
	public static Map<Integer,List<String>> ejercicio2It (List<List<String>>
listas) {
		Map<Integer,List<String>> ac = new HashMap<Integer,List<String>>();
		Integer i = 0;
		while(i<listas.size()) {
			List<String> l = new ArrayList<String>();
			l = listas.get(i);
			for(String cad : l) {
				Integer tam= cad.length();
				if (ac.containsKey(tam)) {
					List<String> mit = ac.get(tam);
					mit.add(cad);
					ac.put(tam, mit);
				}else {
					List<String> mit = new ArrayList<>();
					mit.add(cad);
					ac.put(tam, mit);
				}
			}

			i++;
		}

		return ac;
	}

	//recursiva final
	public static Map<Integer,List<String>> ejercicio2R (List<List<String>>
listas){

		return  ejercicio2RAux (listas,0, new
HashMap<Integer,List<String>>() );
}
	private static Map<Integer, List<String>> ejercicio2RAux(List<List<String>>
listas, Integer i, Map<Integer, List<String>> ac) {
		// TODO Auto-generated method stub
		if(i<listas.size()) {
			List<String> mit = listas.get(i);
			Integer tam = mit.get(i).length() ;

			for(String cad : mit) {
				if (ac.containsKey(tam)) {
```

```
                                ac.get(tam);
                                mit.add(cad);
                                ac.put(tam, mit);
                                return ejercicio2RAux(listas, i++, ac);
                        }else {

                                mit.add(cad);
                                ac.put(tam, mit);
                                return ac;
                        }

                }
                }
        return ejercicio2RAux(listas, i++, ac);
    }
```

Ejercicio3

```
    public record Par(Integer v1, Integer v2) {
      public Par(Integer v1, Integer v2) {
            this.v1=v1;
            this.v2=v2;
       }
            public static Par of(Integer v1, Integer v2) {
                  return new Par(v1,v2);
            }
            public Integer getV1() {
                  return v1;
            }
            public Integer getV2() {
                  return v2;
            }
            public static String String(Integer v1, Integer v2) {
                  String s = "[ v1 = " + String.valueOf(v1) + ", v2 = " +
String.valueOf(v2) + " ]";
                  return s;

            }


       }

    //funcional
    public static String ejercicio3(Integer a, Integer limit) {
            return Stream
            .iterate(Par.of(0, a),
            t -> t.v1 < limit,
            t -> Par.of(t.v1+1, t.v1 % 3 == 1 ? t.v2 : t.v1+t.v2))
            .collect(Collectors.toList())
            .toString();
            }
```

```java
	//iterativa
	public static String ejercicio3It(Integer a, Integer limit) {

		Integer i =0;
		Integer v2 = a;
		String ac = "";


		while(i<limit) {
			Integer v1=i;
			if(i%3 == 1) {
				v2=v2+0;
				ac = ac + ", " + Par.String(i,v2);

			}else {

				v2= v1+v2;

				ac = ac + ", " + Par.String(i,v2);
			}
			i++;


		}

		return ac;
	}
	//recursiva final
public static String ejercicio3R(Integer a, Integer limit) {

		return ejercicio3RAux (a, limit, " ", 0);
	}


	private static String ejercicio3RAux(Integer a, Integer limit, String ac,
Integer i) {
		// TODO Auto-generated method stub

		Integer v2 = a;
		if(i<limit) {
			Integer v1=i;
			if(i%3 == 1) {
				ac =ac + ", " + Par.String(v1, v2);
				return ejercicio3RAux(a,limit, ac, i++);

			}else {
				v2= v1+v2;
				ac = ac + ", " + Par.String(v1, v2);
			}
		}

		return ac;
	}
```

Ejercicio 4

```java
//iterativa (while)
public static Double ejercicio4It(Double n, Double e) {
	Double a = 0.0;
	Double b= n;
	Double c=b/2;


	while (a<n) {
		if(Math.pow(c, 3)-n< Math.pow(e,3)) {

		}

		a++;
	}
	return c;
}
//recursiva final

public static record infoF (Double a, Double b) {
	public static infoF of(Double num) {
		return new infoF (0.0,num);
	}

	public infoF next(Double n) {
		Double c = (a+ b)/2;
		Double newA= a;
		Double newB= b;
		if(c<n) {
			newA=c;
		}else {
			newB=c;
		}
		return new infoF(newA,newB);
	}

}
// funcional
public static Double ejercicio4F(Double n, Double e) {
	Double a = 0.0;
	Double b= n;
	Double c=b/2;
	infoF f= Stream.iterate(infoF.of(n), x ->x.next(n))
			.filter(p->Math.pow(c, 3)-n < Math.pow(e, 3))
			.findFirst().get();
	return f.b();
			}
```

Tipo test

```
public static void test1() {
		// El predicado sobre String devuelve cierto si dicho String contiene alguna vocal abierta (es decir, a, e ó o)
		// El predicado sobre Integer devuelve cierto si ese entero es par
		// La función String -> Integer devuelve la longitud de la cadena
		List<String> filas = Files2.linesFromFile("./ficheros/PI1E1_DatosEntrada.txt");
		Predicate<String> pS = x -> x.contains("a") || x.contains("e")|| x.contains("o");// El predicado sobre String devuelve cierto si dicho String contiene alguna vocal abierta (es decir, a, e ó o)
		Predicate<Integer> pI = x ->x%2==0; // El predicado sobre Integer devuelve cierto si ese entero es par
		Function<String, Integer> f = x -> x.length();// La función String -> Integer devuelve la longitud de la cadena


	System.out.println("#####################################################");
	System.out.println("#                   Ejercicio 1                     #");
	System.out.println("#         ficheros/PI1E1_DatosEntrada.txt           #");
	System.out.println("#####################################################");
	for (String linea : filas) {
		List<String> ls = new ArrayList<String>();
		if(!linea.startsWith("//") || linea.isEmpty()) {
		System.out.println("Entrada"+"["+ linea +"]");
		String[] partes = linea.split(", ");
			for (String parte : partes) {
			ls.add(parte);
			}
			System.out.println("Iterativo: " + Ejercicio1.ejercicio1It(ls, pS, pI, f));
			System.out.println("Recursivo: " + Ejercicio1.ejercicio1RF(ls, pS, pI, f));
			System.out.println("funcional: " + Ejercicio1.ejercicio1F(ls, pS, pI, f));
		}
	}
	System.out.println("#####################################################");
	}

	public static void test2() {
		List<String> filas = Files2.linesFromFile("./ficheros/PI1E2_DatosEntrada1.txt");
		//List<String> filas = Files2.linesFromFile("./ficheros/PI1E2_DatosEntrada2.txt");
		List<List<String>> ls = new ArrayList<List<String>>();
		for (String f: filas) {
			if(f.isEmpty()) {
				ls.add(new ArrayList<String>());
			}
			else {
				ls.add(separarLineas(f));
```

```
                }
                }



        System.out.println("#####################################################");
        System.out.println("#                   Ejercicio 2                     #");
        System.out.println("#         ficheros/PI1E2_DatosEntrada1.txt
#");
        System.out.println("#####################################################");

        System.out.println("Entrada" + ls);
        System.out.println("Iterativo (While): " + Ejercicio2.ejercicio2It(ls));
        System.out.println("funcional: " + Ejercicio2.ejercicio2It(ls));
        System.out.println("Recursivo: " + Ejercicio2.ejercicio2R(ls));// arreglar

        System.out.println("#####################################################");
        }

        private static List<String> separarLineas (String l){
                List<String> res =  new ArrayList<String>();
                for (String s:l.split(",")) {
                        res.add(s);
                }
                return res;

        }
        private static List<Integer> separarLineas2(String l){
                List<Integer> res =  new ArrayList<Integer>();
                for (String s:l.split(",")) {
                        res.add(Integer.parseInt(s));
                }
                return res;

        }

        public static void test3() {
                List<String> filas =
Files2.linesFromFile("./ficheros/PI1E3_DatosEntrada.txt");
                List<List<Integer>> ls = new ArrayList<List<Integer>>();
                Integer v1 = 0;
                Integer v2 =0 ;
                for(String fila : filas) {
                        if(fila.isEmpty()) {
                                ls.add(new ArrayList<Integer>());
                        }
                        else {
                                ls.add(separarLineas2(fila));
                        }

                }
```

```
		System.out.println("#####################################################");
		System.out.println("#                    Ejercicio 3                    #");
		System.out.println("#         ficheros/PI1E3_DatosEntrada.txt           #");
		System.out.println("#####################################################");
		for(List<Integer> l : ls) {
			v1=l.get(0);
			v2=l.get(1);
		System.out.println("Entrada" + Ejercicio3.Par.of(v1, v2));
		System.out.println("Iterativo (While): " + Ejercicio3.ejercicio3It(v1,v2));
		System.out.println("funcional: " + Ejercicio3.ejercicio3(v1,v2));
		System.out.println("Recursivo: " + Ejercicio3.ejercicio3R(v1,v2));

		}
		System.out.println("#####################################################");
		}
```

Salidas

```
#####################################################
#                   Ejercicio 1                     #
#         ficheros/PI1E1_DatosEntrada.txt           #
#####################################################
Entrada[Ingreso]
Iterativo: false
Recursivo: false
funcional: false
Entrada[Ingresos]
Iterativo: true
Recursivo: true
funcional: true
Entrada[Ingreso,Ingresos]
Iterativo: true
Recursivo: true
funcional: true
Entrada[ejercicios,practica,propuesta]
Iterativo: false
Recursivo: false
funcional: false
Entrada[pim,pam,pum]
Iterativo: false
Recursivo: false
funcional: false
Entrada[cadena,recomendar,definir]
Iterativo: false
Recursivo: false
funcional: false
Entrada[elemento,implementar,sol]
Iterativo: true
Recursivo: true
funcional: true
Entrada[ala,map,public,static]
Iterativo: false
```

Recursivo: false
funcional: false
Entrada[Aplicación, Rod, Palomitas de maíz, Machine, Pizza, Hormigas, Ingresos, Ayuda, Celebración, Hijo, Ejemplo, Tres, Significación, Gancho, Mujeres, Gracias, Aprobación, Extensión, Ropa, Rey, Ansiedad, Guante, Carne, Volumen, Pasta de dientes, Calendario, Llave, Unidad, Lectura, Locket, Necesidad, Tela, Reunión, Parche, Tanque, Entrada, Naranja, Lenguaje, Mono, Bote]
Iterativo: true
Recursivo: true
funcional: true

```
##################################################
#                  Ejercicio 2                   #
#         ficheros/PI1E2_DatosEntrada1.txt        #
##################################################
```

Entrada[[ejercicios, practica, propuesta], [cadena, recomendar, definir], [elemento, implementar, sol], [ala, map, public, static]]
Iterativo (While): {3=[sol, ala, map], 6=[cadena, public, static], 7=[definir], 8=[practica, elemento], 9=[propuesta], 10=[ejercicios, recomendar], 11=[implementar]}
funcional: {3=[sol, ala, map], 6=[cadena, public, static], 7=[definir], 8=[practica, elemento], 9=[propuesta], 10=[ejercicios, recomendar], 11=[implementar]}

```
##################################################
#                  Ejercicio 2                   #
#         ficheros/PI1E2_DatosEntrada2.txt       #
##################################################
```

Entrada[[lorem, ipsum, dolor, sit, amet, consectetuer, adipiscing, elit, sed, diam, nonummy, nibh, euismod, tincidunt, ut, laoreet, dolore, magna, aliquam, erat, volutpat, wisi, enim, ad, minim, veniam, quis, nostrud, exerci, tation, ullamcorper, suscipit, lobortis, nisl, aliquip, ex, ea, commodo, consequat, duis, autem, vel, eum, iriure, in, hendrerit, vulputate, velit, esse, molestie, illum, eu, feugiat, nulla, facilisis, at, vero, eros, et, accumsan, iusto, odio, dignissim, qui, blandit, praesent, luptatum, zzril, delenit, augue, te, feugait, facilisiexpetenda, partem, placerat, sea, eam, his, putant, aeterno, interesset, usu, mundi, omnium, virtute, aliquando, ius, aperiri, sententiae, duo, nullam, dolorum, quaestio, ei, vidit, per, ne, impedit, iracundia, neglegentur, consetetur, vis, animal, legimus, inimicus, id, audiam, deserunt, ubique, voluptatibus, reque, dicta, rebum, dissentiet, vim, omnis, deseruisse, ullum, deleniti, vituperata, quo, insolens, complectitur, eos, pri, dico, munere, propriae, ferri, facilis, paulo, ridens, possim, alterum, cu, accusamus, consulatu, decore, soleat, appareat, est, mucius, quaeque, quas, phaedrum, efficiantur, mediocritatem, hinc, oratio, gloriatur, brute, noluisse, verear, disputando, quem, legere, unum, soluta, ludus, vide, homero, pro, dicant, sensibus, conclusionemque, malis, evertitur, torquatos, has, solum, harum, reprimique, saperet, tractatos, antiopam, inani, postulant, mel, qualisque, forensibus, salutandi, mea, assentior, tamquam, sanctus, democritum, mei, electram, adversarium, vix, probo, iuvaret, posse, epicurei, suavitate, an, nam, menandri, accusata, petentium, meis, vocent, signiferumque, corpora, recusabo, eruditi, suscipiantur, mazim, sapientem, debet, commune, eius, falli, volumus, expetenda, sint, quando, delectus, constituto, mnesarchum, impetus, abhorreant, no, definiebas,

rationibus, vidisse, dolores, nominavi, quod, apeirian, concludaturque, timeam, iudicabit, erant, error, meliore, voluptatum, clita, persius, fabellas, labores, aliquyam, salutatus, persequeris, nemore, fierent, dissentiunt, scribentur, erroribus, temporibus, offendit, semper, invidunt, vituperatoribus, sadipscing, mollis, albucius, contentiones, epicuri, pertinacia, nec, utamur, lucilius, aeque, iudico, comprehensam, populo, delicatissimi, vitae, accusam, vivendum, dolorem, expetendis, mutat, consul, natum, numquam, fabulas, melius, diceret, mandamus, tempor, nostro, integre, bonorum, assum, solet, scriptorem, interpretaris, debitis, necessitatibus, graeco, choro, possit, corrumpit, noster, delicata, dicunt, percipit, audire, prompta, efficiendi, disputationi, veri, admodum, ceteros, nihil, oporteat, molestiae, oblique, oportere, urbanitas, labitur, moderatius, puto, scripserit, maiorum, habemus, detraxit, splendide, facilisi, alii, reprehendunt, kasd, senserit, mediocrem, gubergren, repudiare, postea, pertinax, adhuc, percipitur, eirmod, tritani, verterem, summo, tibique, fugit, quodsi, dicit, everti, deterruisset, definitiones, tollit, graecis, instructior, saepe, scaevola, takimata, errem, dictas, posidonium, modo, affert, incorrupte, equidem, congue, platonem, adolescens, imperdiet, aperiam, aliquid, malorum, usuest, provis]]
Iterativo (While): {2=[ut, ad, ex, ea, in, eu, at, et, te, ei, ne, id, cu, an, no], 3=[sit, sed, vel, eum, qui, sea, eam, his, usu, ius, duo, per, vis, vim, quo, eos, pri, est, pro, has, mel, mea, mei, vix, nam, nec], 4=[amet, elit, diam, nibh, erat, wisi, enim, quis, nisl, duis, esse, vero, eros, odio, dico, quas, hinc, quem, unum, vide, meis, eius, sint, quod, veri, puto, alii, kasd, modo], 5=[lorem, ipsum, dolor, magna, minim, autem, velit, illum, nulla, iusto, zzril, augue, mundi, vidit, reque, dicta, rebum, omnis, ullum, ferri, paulo, brute, ludus, malis, solum, harum, inani, probo, posse, mazim, debet, falli, erant, error, clita, aeque, vitae, mutat, natum, assum, solet, choro, nihil, adhuc, summo, fugit, dicit, saepe, errem], 6=[dolore, veniam, exerci, tation, iriure, partem, putant, omnium, nullam, animal, audiam, ubique, munere, ridens, possim, decore, soleat, mucius, oratio, verear, legere, soluta, homero, dicant, vocent, quando, timeam, nemore, semper, mollis, utamur, iudico, populo, consul, melius, tempor, nostro, graeco, possit, noster, dicunt, audire, postea, eirmod, quodsi, everti, tollit, dictas, affert, congue, usuest, provis], 7=[nonummy, euismod, laoreet, aliquam, nostrud, aliquip, commodo, feugiat, blandit, delenit, feugait, aeterno, virtute, aperiri, dolorum, impedit, legimus, facilis, alterum, quaeque, saperet, tamquam, sanctus, iuvaret, corpora, eruditi, commune, volumus, impetus, vidisse, dolores, meliore, persius, labores, fierent, epicuri, accusam, dolorem, numquam, fabulas, diceret, integre, bonorum, debitis, prompta, admodum, ceteros, oblique, labitur, maiorum, habemus, tritani, tibique, graecis, equidem, aperiam, aliquid, malorum], 8=[volutpat, suscipit, lobortis, molestie, accumsan, praesent, luptatum, placerat, quaestio, inimicus, deserunt, deleniti, insolens, propriae, appareat, phaedrum, noluisse, sensibus, antiopam, electram, epicurei, menandri, accusata, recusabo, delectus, nominavi, apeirian, fabellas, aliquyam, offendit, invidunt, albucius, lucilius, vivendum, mandamus, delicata, percipit, oporteat, oportere, detraxit, facilisi, senserit, pertinax, verterem, scaevola, takimata, platonem], 9=[tincidunt, consequat, hendrerit, vulputate, facilisis, dignissim, aliquando, iracundia, accusamus, consulatu, gloriatur, evertitur, torquatos, tractatos, postulant, qualisque, salutandi, assentior, suavitate, petentium, sapientem, expetenda, iudicabit, salutatus, erroribus, corrumpit, molestiae, urbanitas, splendide, mediocrem, gubergren, repudiare, imperdiet], 10=[adipiscing, interesset, sententiae, consetetur, dissentiet, deseruisse, vituperata, disputando, reprimique, forensibus, democritum, constituto, mnesarchum, abhorreant, definiebas, rationibus, voluptatum, scribentur, temporibus, sadipscing, pertinacia, expetendis, scriptorem, efficiendi, moderatius, scripserit, percipitur, posidonium, incorrupte, adolescens], 11=[ullamcorper, neglegentur, efficiantur, adversarium, persequeris, dissentiunt, instructior], 12=[consectetuer, voluptatibus, complectitur, suscipiantur, contentiones,

comprehensam, disputationi, reprehendunt, deterruisset, definitiones],
13=[mediocritatem, signiferumque, delicatissimi, interpretaris],
14=[concludaturque, necessitatibus], 15=[conclusionemque, vituperatoribus],
17=[facilisiexpetenda]}
funcional: {2=[ut, ad, ex, ea, in, eu, at, et, te, ei, ne, id, cu, an, no], 3=[sit,
sed, vel, eum, qui, sea, eam, his, usu, ius, duo, per, vis, vim, quo, eos, pri, est,
pro, has, mel, mea, mei, vix, nam, nec], 4=[amet, elit, diam, nibh, erat, wisi, enim,
quis, nisl, duis, esse, vero, eros, odio, dico, quas, hinc, quem, unum, vide, meis,
eius, sint, quod, veri, puto, alii, kasd, modo], 5=[lorem, ipsum, dolor, magna, minim,
autem, velit, illum, nulla, iusto, zzril, augue, mundi, vidit, reque, dicta, rebum,
omnis, ullum, ferri, paulo, brute, ludus, malis, solum, harum, inani, probo, posse,
mazim, debet, falli, erant, error, clita, aeque, vitae, mutat, natum, assum, solet,
choro, nihil, adhuc, summo, fugit, dicit, saepe, errem], 6=[dolore, veniam, exerci,
tation, iriure, partem, putant, omnium, nullam, animal, audiam, ubique, munere,
ridens, possim, decore, soleat, mucius, oratio, verear, legere, soluta, homero,
dicant, vocent, quando, timeam, nemore, semper, mollis, utamur, iudico, populo,
consul, melius, tempor, nostro, graeco, possit, noster, dicunt, audire, postea,
eirmod, quodsi, everti, tollit, dictas, affert, congue, usuest, provis], 7=[nonummy,
euismod, laoreet, aliquam, nostrud, aliquip, commodo, feugiat, blandit, delenit,
feugait, aeterno, virtute, aperiri, dolorum, impedit, legimus, facilis, alterum,
quaeque, saperet, tamquam, sanctus, iuvaret, corpora, eruditi, commune, volumus,
impetus, vidisse, dolores, meliore, persius, labores, fierent, epicuri, accusam,
dolorem, numquam, fabulas, diceret, integre, bonorum, debitis, prompta, admodum,
ceteros, oblique, labitur, maiorum, habemus, tritani, tibique, graecis, equidem,
aperiam, aliquid, malorum], 8=[volutpat, suscipit, lobortis, molestie, accumsan,
praesent, luptatum, placerat, quaestio, inimicus, deserunt, deleniti, insolens,
propriae, appareat, phaedrum, noluisse, sensibus, antiopam, electram, epicurei,
menandri, accusata, recusabo, delectus, nominavi, apeirian, fabellas, aliquyam,
offendit, invidunt, albucius, lucilius, vivendum, mandamus, delicata, percipit,
oporteat, oportere, detraxit, facilisi, senserit, pertinax, verterem, scaevola,
takimata, platonem], 9=[tincidunt, consequat, hendrerit, vulputate, facilisis,
dignissim, aliquando, iracundia, accusamus, consulatu, gloriatur, evertitur,
torquatos, tractatos, postulant, qualisque, salutandi, assentior, suavitate,
petentium, sapientem, expetenda, iudicabit, salutatus, erroribus, corrumpit,
molestiae, urbanitas, splendide, mediocrem, gubergren, repudiare, imperdiet],
10=[adipiscing, interesset, sententiae, consetetur, dissentiet, deseruisse,
vituperata, disputando, reprimique, forensibus, democritum, constituto, mnesarchum,
abhorreant, definiebas, rationibus, voluptatum, scribentur, temporibus, sadipscing,
pertinacia, expetendis, scriptorem, efficiendi, moderatius, scripserit, percipitur,
posidonium, incorrupte, adolescens], 11=[ullamcorper, neglegentur, efficiantur,
adversarium, persequeris, dissentiunt, instructior], 12=[consectetuer,
voluptatibus, complectitur, suscipiantur, contentiones, comprehensam,
disputationi, reprehendunt, deterruisset, definitiones], 13=[mediocritatem,
signiferumque, delicatissimi, interpretaris], 14=[concludaturque, necessitatibus],
15=[conclusionemque, vituperatoribus], 17=[facilisiexpetenda]}

##

##
Ejercicio 3
ficheros/PI1E3_DatosEntrada.txt
##
EntradaPar[v1=86, v2=87]

Iterativo (While): , [v1 = 0, v2 = 86], [v1 = 1, v2 = 86], [v1 = 2, v2 = 88], [v1 = 3, v2 = 91], [v1 = 4, v2 = 91], [v1 = 5, v2 = 96], [v1 = 6, v2 = 102], [v1 = 7, v2 = 102], [v1 = 8, v2 = 110], [v1 = 9, v2 = 119], [v1 = 10, v2 = 119], [v1 = 11, v2 = 130], [v1 = 12, v2 = 142], [v1 = 13, v2 = 142], [v1 = 14, v2 = 156], [v1 = 15, v2 = 171], [v1 = 16, v2 = 171], [v1 = 17, v2 = 188], [v1 = 18, v2 = 206], [v1 = 19, v2 = 206], [v1 = 20, v2 = 226], [v1 = 21, v2 = 247], [v1 = 22, v2 = 247], [v1 = 23, v2 = 270], [v1 = 24, v2 = 294], [v1 = 25, v2 = 294], [v1 = 26, v2 = 320], [v1 = 27, v2 = 347], [v1 = 28, v2 = 347], [v1 = 29, v2 = 376], [v1 = 30, v2 = 406], [v1 = 31, v2 = 406], [v1 = 32, v2 = 438], [v1 = 33, v2 = 471], [v1 = 34, v2 = 471], [v1 = 35, v2 = 506], [v1 = 36, v2 = 542], [v1 = 37, v2 = 542], [v1 = 38, v2 = 580], [v1 = 39, v2 = 619], [v1 = 40, v2 = 619], [v1 = 41, v2 = 660], [v1 = 42, v2 = 702], [v1 = 43, v2 = 702], [v1 = 44, v2 = 746], [v1 = 45, v2 = 791], [v1 = 46, v2 = 791], [v1 = 47, v2 = 838], [v1 = 48, v2 = 886], [v1 = 49, v2 = 886], [v1 = 50, v2 = 936], [v1 = 51, v2 = 987], [v1 = 52, v2 = 987], [v1 = 53, v2 = 1040], [v1 = 54, v2 = 1094], [v1 = 55, v2 = 1094], [v1 = 56, v2 = 1150], [v1 = 57, v2 = 1207], [v1 = 58, v2 = 1207], [v1 = 59, v2 = 1266], [v1 = 60, v2 = 1326], [v1 = 61, v2 = 1326], [v1 = 62, v2 = 1388], [v1 = 63, v2 = 1451], [v1 = 64, v2 = 1451], [v1 = 65, v2 = 1516], [v1 = 66, v2 = 1582], [v1 = 67, v2 = 1582], [v1 = 68, v2 = 1650], [v1 = 69, v2 = 1719], [v1 = 70, v2 = 1719], [v1 = 71, v2 = 1790], [v1 = 72, v2 = 1862], [v1 = 73, v2 = 1862], [v1 = 74, v2 = 1936], [v1 = 75, v2 = 2011], [v1 = 76, v2 = 2011], [v1 = 77, v2 = 2088], [v1 = 78, v2 = 2166], [v1 = 79, v2 = 2166], [v1 = 80, v2 = 2246], [v1 = 81, v2 = 2327], [v1 = 82, v2 = 2327], [v1 = 83, v2 = 2410], [v1 = 84, v2 = 2494], [v1 = 85, v2 = 2494], [v1 = 86, v2 = 2580]
funcional: [Par[v1=0, v2=86], Par[v1=1, v2=86], Par[v1=2, v2=86], Par[v1=3, v2=88], Par[v1=4, v2=91], Par[v1=5, v2=91], Par[v1=6, v2=96], Par[v1=7, v2=102], Par[v1=8, v2=102], Par[v1=9, v2=110], Par[v1=10, v2=119], Par[v1=11, v2=119], Par[v1=12, v2=130], Par[v1=13, v2=142], Par[v1=14, v2=142], Par[v1=15, v2=156], Par[v1=16, v2=171], Par[v1=17, v2=171], Par[v1=18, v2=188], Par[v1=19, v2=206], Par[v1=20, v2=206], Par[v1=21, v2=226], Par[v1=22, v2=247], Par[v1=23, v2=247], Par[v1=24, v2=270], Par[v1=25, v2=294], Par[v1=26, v2=294], Par[v1=27, v2=320], Par[v1=28, v2=347], Par[v1=29, v2=347], Par[v1=30, v2=376], Par[v1=31, v2=406], Par[v1=32, v2=406], Par[v1=33, v2=438], Par[v1=34, v2=471], Par[v1=35, v2=471], Par[v1=36, v2=506], Par[v1=37, v2=542], Par[v1=38, v2=542], Par[v1=39, v2=580], Par[v1=40, v2=619], Par[v1=41, v2=619], Par[v1=42, v2=660], Par[v1=43, v2=702], Par[v1=44, v2=702], Par[v1=45, v2=746], Par[v1=46, v2=791], Par[v1=47, v2=791], Par[v1=48, v2=838], Par[v1=49, v2=886], Par[v1=50, v2=886], Par[v1=51, v2=936], Par[v1=52, v2=987], Par[v1=53, v2=987], Par[v1=54, v2=1040], Par[v1=55, v2=1094], Par[v1=56, v2=1094], Par[v1=57, v2=1150], Par[v1=58, v2=1207], Par[v1=59, v2=1207], Par[v1=60, v2=1266], Par[v1=61, v2=1326], Par[v1=62, v2=1326], Par[v1=63, v2=1388], Par[v1=64, v2=1451], Par[v1=65, v2=1451], Par[v1=66, v2=1516], Par[v1=67, v2=1582], Par[v1=68, v2=1582], Par[v1=69, v2=1650], Par[v1=70, v2=1719], Par[v1=71, v2=1719], Par[v1=72, v2=1790], Par[v1=73, v2=1862], Par[v1=74, v2=1862], Par[v1=75, v2=1936], Par[v1=76, v2=2011], Par[v1=77, v2=2011], Par[v1=78, v2=2088], Par[v1=79, v2=2166], Par[v1=80, v2=2166], Par[v1=81, v2=2246], Par[v1=82, v2=2327], Par[v1=83, v2=2327], Par[v1=84, v2=2410], Par[v1=85, v2=2494], Par[v1=86, v2=2494]]
Recursivo: , [v1 = 0, v2 = 86]
EntradaPar[v1=39, v2=88]
Iterativo (While): , [v1 = 0, v2 = 39], [v1 = 1, v2 = 39], [v1 = 2, v2 = 41], [v1 = 3, v2 = 44], [v1 = 4, v2 = 44], [v1 = 5, v2 = 49], [v1 = 6, v2 = 55], [v1 = 7, v2 = 55], [v1 = 8, v2 = 63], [v1 = 9, v2 = 72], [v1 = 10, v2 = 72], [v1 = 11, v2 = 83], [v1 = 12, v2 = 95], [v1 = 13, v2 = 95], [v1 = 14, v2 = 109], [v1 = 15, v2 = 124], [v1 = 16, v2 = 124], [v1 = 17, v2 = 141], [v1 = 18, v2 = 159], [v1 = 19, v2 = 159], [v1 = 20, v2 = 179], [v1 = 21, v2 = 200

], [v1 = 22, v2 = 200], [v1 = 23, v2 = 223], [v1 = 24, v2 = 247], [v1 = 25, v2 = 247], [v1 = 26, v2 = 273], [v1 = 27, v2 = 300], [v1 = 28, v2 = 300], [v1 = 29, v2 = 329], [v1 = 30, v2 = 359], [v1 = 31, v2 = 359], [v1 = 32, v2 = 391], [v1 = 33, v2 = 424], [v1 = 34, v2 = 424], [v1 = 35, v2 = 459], [v1 = 36, v2 = 495], [v1 = 37, v2 = 495], [v1 = 38, v2 = 533], [v1 = 39, v2 = 572], [v1 = 40, v2 = 572], [v1 = 41, v2 = 613], [v1 = 42, v2 = 655], [v1 = 43, v2 = 655], [v1 = 44, v2 = 699], [v1 = 45, v2 = 744], [v1 = 46, v2 = 744], [v1 = 47, v2 = 791], [v1 = 48, v2 = 839], [v1 = 49, v2 = 839], [v1 = 50, v2 = 889], [v1 = 51, v2 = 940], [v1 = 52, v2 = 940], [v1 = 53, v2 = 993], [v1 = 54, v2 = 1047], [v1 = 55, v2 = 1047], [v1 = 56, v2 = 1103], [v1 = 57, v2 = 1160], [v1 = 58, v2 = 1160], [v1 = 59, v2 = 1219], [v1 = 60, v2 = 1279], [v1 = 61, v2 = 1279], [v1 = 62, v2 = 1341], [v1 = 63, v2 = 1404], [v1 = 64, v2 = 1404], [v1 = 65, v2 = 1469], [v1 = 66, v2 = 1535], [v1 = 67, v2 = 1535], [v1 = 68, v2 = 1603], [v1 = 69, v2 = 1672], [v1 = 70, v2 = 1672], [v1 = 71, v2 = 1743], [v1 = 72, v2 = 1815], [v1 = 73, v2 = 1815], [v1 = 74, v2 = 1889], [v1 = 75, v2 = 1964], [v1 = 76, v2 = 1964], [v1 = 77, v2 = 2041], [v1 = 78, v2 = 2119], [v1 = 79, v2 = 2119], [v1 = 80, v2 = 2199], [v1 = 81, v2 = 2280], [v1 = 82, v2 = 2280], [v1 = 83, v2 = 2363], [v1 = 84, v2 = 2447], [v1 = 85, v2 = 2447], [v1 = 86, v2 = 2533], [v1 = 87, v2 = 2620]
funcional: [Par[v1=0, v2=39], Par[v1=1, v2=39], Par[v1=2, v2=39], Par[v1=3, v2=41], Par[v1=4, v2=44], Par[v1=5, v2=44], Par[v1=6, v2=49], Par[v1=7, v2=55], Par[v1=8, v2=55], Par[v1=9, v2=63], Par[v1=10, v2=72], Par[v1=11, v2=72], Par[v1=12, v2=83], Par[v1=13, v2=95], Par[v1=14, v2=95], Par[v1=15, v2=109], Par[v1=16, v2=124], Par[v1=17, v2=124], Par[v1=18, v2=141], Par[v1=19, v2=159], Par[v1=20, v2=159], Par[v1=21, v2=179], Par[v1=22, v2=200], Par[v1=23, v2=200], Par[v1=24, v2=223], Par[v1=25, v2=247], Par[v1=26, v2=247], Par[v1=27, v2=273], Par[v1=28, v2=300], Par[v1=29, v2=300], Par[v1=30, v2=329], Par[v1=31, v2=359], Par[v1=32, v2=359], Par[v1=33, v2=391], Par[v1=34, v2=424], Par[v1=35, v2=424], Par[v1=36, v2=459], Par[v1=37, v2=495], Par[v1=38, v2=495], Par[v1=39, v2=533], Par[v1=40, v2=572], Par[v1=41, v2=572], Par[v1=42, v2=613], Par[v1=43, v2=655], Par[v1=44, v2=655], Par[v1=45, v2=699], Par[v1=46, v2=744], Par[v1=47, v2=744], Par[v1=48, v2=791], Par[v1=49, v2=839], Par[v1=50, v2=839], Par[v1=51, v2=889], Par[v1=52, v2=940], Par[v1=53, v2=940], Par[v1=54, v2=993], Par[v1=55, v2=1047], Par[v1=56, v2=1047], Par[v1=57, v2=1103], Par[v1=58, v2=1160], Par[v1=59, v2=1160], Par[v1=60, v2=1219], Par[v1=61, v2=1279], Par[v1=62, v2=1279], Par[v1=63, v2=1341], Par[v1=64, v2=1404], Par[v1=65, v2=1404], Par[v1=66, v2=1469], Par[v1=67, v2=1535], Par[v1=68, v2=1535], Par[v1=69, v2=1603], Par[v1=70, v2=1672], Par[v1=71, v2=1672], Par[v1=72, v2=1743], Par[v1=73, v2=1815], Par[v1=74, v2=1815], Par[v1=75, v2=1889], Par[v1=76, v2=1964], Par[v1=77, v2=1964], Par[v1=78, v2=2041], Par[v1=79, v2=2119], Par[v1=80, v2=2119], Par[v1=81, v2=2199], Par[v1=82, v2=2280], Par[v1=83, v2=2280], Par[v1=84, v2=2363], Par[v1=85, v2=2447], Par[v1=86, v2=2447], Par[v1=87, v2=2533]]
Recursivo: , [v1 = 0, v2 = 39]
EntradaPar[v1=-78, v2=50]
Iterativo (While): , [v1 = 0, v2 = -78], [v1 = 1, v2 = -78], [v1 = 2, v2 = -76], [v1 = 3, v2 = -73], [v1 = 4, v2 = -73], [v1 = 5, v2 = -68], [v1 = 6, v2 = -62], [v1 = 7, v2 = -62], [v1 = 8, v2 = -54], [v1 = 9, v2 = -45], [v1 = 10, v2 = -45], [v1 = 11, v2 = -34], [v1 = 12, v2 = -22], [v1 = 13, v2 = -22], [v1 = 14, v2 = -8], [v1 = 15, v2 = 7], [v1 = 16, v2 = 7], [v1 = 17, v2 = 24], [v1 = 18, v2 = 42], [v1 = 19, v2 = 42], [v1 = 20, v2 = 62], [v1 = 21, v2 = 83], [v1 = 22, v2 = 83], [v1 = 23, v2 = 106], [v1 = 24, v2 = 130], [v1 = 25, v2 = 130], [v1 = 26, v2 = 156], [v1 = 27, v2 = 183], [v1 = 28, v2 = 183], [v1 = 29, v2 = 212], [v1 = 30, v2 = 242], [v1 = 31, v2 = 242], [v1 = 32, v2 = 274], [v1 = 33, v2 = 307], [v1 = 34, v2 = 307], [v1 = 35, v2 = 342], [v1 = 36, v2 = 378], [v1 = 37, v2 = 378], [v1 = 38, v2 = 416], [v1 = 39, v2 = 455], [v1 = 40, v2 = 455], [v1 = 41, v2 = 496], [v1 = 42, v2 = 538], [v1 =

43, v2 = 538], [v1 = 44, v2 = 582], [v1 = 45, v2 = 627], [v1 = 46, v2 = 627], [v1 = 47, v2 = 674], [v1 = 48, v2 = 722], [v1 = 49, v2 = 722]
funcional: [Par[v1=0, v2=-78], Par[v1=1, v2=-78], Par[v1=2, v2=-78], Par[v1=3, v2=-76], Par[v1=4, v2=-73], Par[v1=5, v2=-73], Par[v1=6, v2=-68], Par[v1=7, v2=-62], Par[v1=8, v2=-62], Par[v1=9, v2=-54], Par[v1=10, v2=-45], Par[v1=11, v2=-45], Par[v1=12, v2=-34], Par[v1=13, v2=-22], Par[v1=14, v2=-22], Par[v1=15, v2=-8], Par[v1=16, v2=7], Par[v1=17, v2=7], Par[v1=18, v2=24], Par[v1=19, v2=42], Par[v1=20, v2=42], Par[v1=21, v2=62], Par[v1=22, v2=83], Par[v1=23, v2=83], Par[v1=24, v2=106], Par[v1=25, v2=130], Par[v1=26, v2=130], Par[v1=27, v2=156], Par[v1=28, v2=183], Par[v1=29, v2=183], Par[v1=30, v2=212], Par[v1=31, v2=242], Par[v1=32, v2=242], Par[v1=33, v2=274], Par[v1=34, v2=307], Par[v1=35, v2=307], Par[v1=36, v2=342], Par[v1=37, v2=378], Par[v1=38, v2=378], Par[v1=39, v2=416], Par[v1=40, v2=455], Par[v1=41, v2=455], Par[v1=42, v2=496], Par[v1=43, v2=538], Par[v1=44, v2=538], Par[v1=45, v2=582], Par[v1=46, v2=627], Par[v1=47, v2=627], Par[v1=48, v2=674], Par[v1=49, v2=722]]
Recursivo: , [v1 = 0, v2 = -78]
EntradaPar[v1=58, v2=25]
Iterativo (While): , [v1 = 0, v2 = 58], [v1 = 1, v2 = 58], [v1 = 2, v2 = 60], [v1 = 3, v2 = 63], [v1 = 4, v2 = 63], [v1 = 5, v2 = 68], [v1 = 6, v2 = 74], [v1 = 7, v2 = 74], [v1 = 8, v2 = 82], [v1 = 9, v2 = 91], [v1 = 10, v2 = 91], [v1 = 11, v2 = 102], [v1 = 12, v2 = 114], [v1 = 13, v2 = 114], [v1 = 14, v2 = 128], [v1 = 15, v2 = 143], [v1 = 16, v2 = 143], [v1 = 17, v2 = 160], [v1 = 18, v2 = 178], [v1 = 19, v2 = 178], [v1 = 20, v2 = 198], [v1 = 21, v2 = 219], [v1 = 22, v2 = 219], [v1 = 23, v2 = 242], [v1 = 24, v2 = 266]
funcional: [Par[v1=0, v2=58], Par[v1=1, v2=58], Par[v1=2, v2=58], Par[v1=3, v2=60], Par[v1=4, v2=63], Par[v1=5, v2=63], Par[v1=6, v2=68], Par[v1=7, v2=74], Par[v1=8, v2=74], Par[v1=9, v2=82], Par[v1=10, v2=91], Par[v1=11, v2=91], Par[v1=12, v2=102], Par[v1=13, v2=114], Par[v1=14, v2=114], Par[v1=15, v2=128], Par[v1=16, v2=143], Par[v1=17, v2=143], Par[v1=18, v2=160], Par[v1=19, v2=178], Par[v1=20, v2=178], Par[v1=21, v2=198], Par[v1=22, v2=219], Par[v1=23, v2=219], Par[v1=24, v2=242]]
Recursivo: , [v1 = 0, v2 = 58]
EntradaPar[v1=84, v2=46]
Iterativo (While): , [v1 = 0, v2 = 84], [v1 = 1, v2 = 84], [v1 = 2, v2 = 86], [v1 = 3, v2 = 89], [v1 = 4, v2 = 89], [v1 = 5, v2 = 94], [v1 = 6, v2 = 100], [v1 = 7, v2 = 100], [v1 = 8, v2 = 108], [v1 = 9, v2 = 117], [v1 = 10, v2 = 117], [v1 = 11, v2 = 128], [v1 = 12, v2 = 140], [v1 = 13, v2 = 140], [v1 = 14, v2 = 154], [v1 = 15, v2 = 169], [v1 = 16, v2 = 169], [v1 = 17, v2 = 186], [v1 = 18, v2 = 204], [v1 = 19, v2 = 204], [v1 = 20, v2 = 224], [v1 = 21, v2 = 245], [v1 = 22, v2 = 245], [v1 = 23, v2 = 268], [v1 = 24, v2 = 292], [v1 = 25, v2 = 292], [v1 = 26, v2 = 318], [v1 = 27, v2 = 345], [v1 = 28, v2 = 345], [v1 = 29, v2 = 374], [v1 = 30, v2 = 404], [v1 = 31, v2 = 404], [v1 = 32, v2 = 436], [v1 = 33, v2 = 469], [v1 = 34, v2 = 469], [v1 = 35, v2 = 504], [v1 = 36, v2 = 540], [v1 = 37, v2 = 540], [v1 = 38, v2 = 578], [v1 = 39, v2 = 617], [v1 = 40, v2 = 617], [v1 = 41, v2 = 658], [v1 = 42, v2 = 700], [v1 = 43, v2 = 700], [v1 = 44, v2 = 744], [v1 = 45, v2 = 789]
funcional: [Par[v1=0, v2=84], Par[v1=1, v2=84], Par[v1=2, v2=84], Par[v1=3, v2=86], Par[v1=4, v2=89], Par[v1=5, v2=89], Par[v1=6, v2=94], Par[v1=7, v2=100], Par[v1=8, v2=100], Par[v1=9, v2=108], Par[v1=10, v2=117], Par[v1=11, v2=117], Par[v1=12, v2=128], Par[v1=13, v2=140], Par[v1=14, v2=140], Par[v1=15, v2=154], Par[v1=16, v2=169], Par[v1=17, v2=169], Par[v1=18, v2=186], Par[v1=19, v2=204], Par[v1=20, v2=204], Par[v1=21, v2=224], Par[v1=22, v2=245], Par[v1=23, v2=245], Par[v1=24, v2=268], Par[v1=25, v2=292], Par[v1=26, v2=292], Par[v1=27, v2=318], Par[v1=28, v2=345], Par[v1=29, v2=345], Par[v1=30, v2=374], Par[v1=31, v2=404], Par[v1=32, v2=404], Par[v1=33, v2=436], Par[v1=34, v2=469], Par[v1=35, v2=469], Par[v1=36, v2=504], Par[v1=37, v2=540], Par[v1=38, v2=540], Par[v1=39, v2=578], Par[v1=40,

v2=617], Par[v1=41, v2=617], Par[v1=42, v2=658], Par[v1=43, v2=700], Par[v1=44, v2=700], Par[v1=45, v2=744]]
Recursivo: , [v1 = 0, v2 = 84]
EntradaPar[v1=-20, v2=97]
Iterativo (While): , [v1 = 0, v2 = -20], [v1 = 1, v2 = -20], [v1 = 2, v2 = -18], [v1 = 3, v2 = -15], [v1 = 4, v2 = -15], [v1 = 5, v2 = -10], [v1 = 6, v2 = -4], [v1 = 7, v2 = -4], [v1 = 8, v2 = 4], [v1 = 9, v2 = 13], [v1 = 10, v2 = 13], [v1 = 11, v2 = 24], [v1 = 12, v2 = 36], [v1 = 13, v2 = 36], [v1 = 14, v2 = 50], [v1 = 15, v2 = 65], [v1 = 16, v2 = 65], [v1 = 17, v2 = 82], [v1 = 18, v2 = 100], [v1 = 19, v2 = 100], [v1 = 20, v2 = 120], [v1 = 21, v2 = 141], [v1 = 22, v2 = 141], [v1 = 23, v2 = 164], [v1 = 24, v2 = 188], [v1 = 25, v2 = 188], [v1 = 26, v2 = 214], [v1 = 27, v2 = 241], [v1 = 28, v2 = 241], [v1 = 29, v2 = 270], [v1 = 30, v2 = 300], [v1 = 31, v2 = 300], [v1 = 32, v2 = 332], [v1 = 33, v2 = 365], [v1 = 34, v2 = 365], [v1 = 35, v2 = 400], [v1 = 36, v2 = 436], [v1 = 37, v2 = 436], [v1 = 38, v2 = 474], [v1 = 39, v2 = 513], [v1 = 40, v2 = 513], [v1 = 41, v2 = 554], [v1 = 42, v2 = 596], [v1 = 43, v2 = 596], [v1 = 44, v2 = 640], [v1 = 45, v2 = 685], [v1 = 46, v2 = 685], [v1 = 47, v2 = 732], [v1 = 48, v2 = 780], [v1 = 49, v2 = 780], [v1 = 50, v2 = 830], [v1 = 51, v2 = 881], [v1 = 52, v2 = 881], [v1 = 53, v2 = 934], [v1 = 54, v2 = 988], [v1 = 55, v2 = 988], [v1 = 56, v2 = 1044], [v1 = 57, v2 = 1101], [v1 = 58, v2 = 1101], [v1 = 59, v2 = 1160], [v1 = 60, v2 = 1220], [v1 = 61, v2 = 1220], [v1 = 62, v2 = 1282], [v1 = 63, v2 = 1345], [v1 = 64, v2 = 1345], [v1 = 65, v2 = 1410], [v1 = 66, v2 = 1476], [v1 = 67, v2 = 1476], [v1 = 68, v2 = 1544], [v1 = 69, v2 = 1613], [v1 = 70, v2 = 1613], [v1 = 71, v2 = 1684], [v1 = 72, v2 = 1756], [v1 = 73, v2 = 1756], [v1 = 74, v2 = 1830], [v1 = 75, v2 = 1905], [v1 = 76, v2 = 1905], [v1 = 77, v2 = 1982], [v1 = 78, v2 = 2060], [v1 = 79, v2 = 2060], [v1 = 80, v2 = 2140], [v1 = 81, v2 = 2221], [v1 = 82, v2 = 2221], [v1 = 83, v2 = 2304], [v1 = 84, v2 = 2388], [v1 = 85, v2 = 2388], [v1 = 86, v2 = 2474], [v1 = 87, v2 = 2561], [v1 = 88, v2 = 2561], [v1 = 89, v2 = 2650], [v1 = 90, v2 = 2740], [v1 = 91, v2 = 2740], [v1 = 92, v2 = 2832], [v1 = 93, v2 = 2925], [v1 = 94, v2 = 2925], [v1 = 95, v2 = 3020], [v1 = 96, v2 = 3116]
funcional: [Par[v1=0, v2=-20], Par[v1=1, v2=-20], Par[v1=2, v2=-20], Par[v1=3, v2=-18], Par[v1=4, v2=-15], Par[v1=5, v2=-15], Par[v1=6, v2=-10], Par[v1=7, v2=-4], Par[v1=8, v2=-4], Par[v1=9, v2=4], Par[v1=10, v2=13], Par[v1=11, v2=13], Par[v1=12, v2=24], Par[v1=13, v2=36], Par[v1=14, v2=36], Par[v1=15, v2=50], Par[v1=16, v2=65], Par[v1=17, v2=65], Par[v1=18, v2=82], Par[v1=19, v2=100], Par[v1=20, v2=100], Par[v1=21, v2=120], Par[v1=22, v2=141], Par[v1=23, v2=141], Par[v1=24, v2=164], Par[v1=25, v2=188], Par[v1=26, v2=188], Par[v1=27, v2=214], Par[v1=28, v2=241], Par[v1=29, v2=241], Par[v1=30, v2=270], Par[v1=31, v2=300], Par[v1=32, v2=300], Par[v1=33, v2=332], Par[v1=34, v2=365], Par[v1=35, v2=365], Par[v1=36, v2=400], Par[v1=37, v2=436], Par[v1=38, v2=436], Par[v1=39, v2=474], Par[v1=40, v2=513], Par[v1=41, v2=513], Par[v1=42, v2=554], Par[v1=43, v2=596], Par[v1=44, v2=596], Par[v1=45, v2=640], Par[v1=46, v2=685], Par[v1=47, v2=685], Par[v1=48, v2=732], Par[v1=49, v2=780], Par[v1=50, v2=780], Par[v1=51, v2=830], Par[v1=52, v2=881], Par[v1=53, v2=881], Par[v1=54, v2=934], Par[v1=55, v2=988], Par[v1=56, v2=988], Par[v1=57, v2=1044], Par[v1=58, v2=1101], Par[v1=59, v2=1101], Par[v1=60, v2=1160], Par[v1=61, v2=1220], Par[v1=62, v2=1220], Par[v1=63, v2=1282], Par[v1=64, v2=1345], Par[v1=65, v2=1345], Par[v1=66, v2=1410], Par[v1=67, v2=1476], Par[v1=68, v2=1476], Par[v1=69, v2=1544], Par[v1=70, v2=1613], Par[v1=71, v2=1613], Par[v1=72, v2=1684], Par[v1=73, v2=1756], Par[v1=74, v2=1756], Par[v1=75, v2=1830], Par[v1=76, v2=1905], Par[v1=77, v2=1905], Par[v1=78, v2=1982], Par[v1=79, v2=2060], Par[v1=80, v2=2060], Par[v1=81, v2=2140], Par[v1=82, v2=2221], Par[v1=83, v2=2221], Par[v1=84, v2=2304], Par[v1=85, v2=2388], Par[v1=86, v2=2388], Par[v1=87, v2=2474], Par[v1=88, v2=2561],

Par[v1=89, v2=2561], Par[v1=90, v2=2650], Par[v1=91, v2=2740], Par[v1=92, v2=2740], Par[v1=93, v2=2832], Par[v1=94, v2=2925], Par[v1=95, v2=2925], Par[v1=96, v2=3020]]
Recursivo: , [v1 = 0, v2 = -20]
EntradaPar[v1=44, v2=89]
Iterativo (While): , [v1 = 0, v2 = 44], [v1 = 1, v2 = 44], [v1 = 2, v2 = 46], [v1 = 3, v2 = 49], [v1 = 4, v2 = 49], [v1 = 5, v2 = 54], [v1 = 6, v2 = 60], [v1 = 7, v2 = 60], [v1 = 8, v2 = 68], [v1 = 9, v2 = 77], [v1 = 10, v2 = 77], [v1 = 11, v2 = 88], [v1 = 12, v2 = 100], [v1 = 13, v2 = 100], [v1 = 14, v2 = 114], [v1 = 15, v2 = 129], [v1 = 16, v2 = 129], [v1 = 17, v2 = 146], [v1 = 18, v2 = 164], [v1 = 19, v2 = 164], [v1 = 20, v2 = 184], [v1 = 21, v2 = 205], [v1 = 22, v2 = 205], [v1 = 23, v2 = 228], [v1 = 24, v2 = 252], [v1 = 25, v2 = 252], [v1 = 26, v2 = 278], [v1 = 27, v2 = 305], [v1 = 28, v2 = 305], [v1 = 29, v2 = 334], [v1 = 30, v2 = 364], [v1 = 31, v2 = 364], [v1 = 32, v2 = 396], [v1 = 33, v2 = 429], [v1 = 34, v2 = 429], [v1 = 35, v2 = 464], [v1 = 36, v2 = 500], [v1 = 37, v2 = 500], [v1 = 38, v2 = 538], [v1 = 39, v2 = 577], [v1 = 40, v2 = 577], [v1 = 41, v2 = 618], [v1 = 42, v2 = 660], [v1 = 43, v2 = 660], [v1 = 44, v2 = 704], [v1 = 45, v2 = 749], [v1 = 46, v2 = 749], [v1 = 47, v2 = 796], [v1 = 48, v2 = 844], [v1 = 49, v2 = 844], [v1 = 50, v2 = 894], [v1 = 51, v2 = 945], [v1 = 52, v2 = 945], [v1 = 53, v2 = 998], [v1 = 54, v2 = 1052], [v1 = 55, v2 = 1052], [v1 = 56, v2 = 1108], [v1 = 57, v2 = 1165], [v1 = 58, v2 = 1165], [v1 = 59, v2 = 1224], [v1 = 60, v2 = 1284], [v1 = 61, v2 = 1284], [v1 = 62, v2 = 1346], [v1 = 63, v2 = 1409], [v1 = 64, v2 = 1409], [v1 = 65, v2 = 1474], [v1 = 66, v2 = 1540], [v1 = 67, v2 = 1540], [v1 = 68, v2 = 1608], [v1 = 69, v2 = 1677], [v1 = 70, v2 = 1677], [v1 = 71, v2 = 1748], [v1 = 72, v2 = 1820], [v1 = 73, v2 = 1820], [v1 = 74, v2 = 1894], [v1 = 75, v2 = 1969], [v1 = 76, v2 = 1969], [v1 = 77, v2 = 2046], [v1 = 78, v2 = 2124], [v1 = 79, v2 = 2124], [v1 = 80, v2 = 2204], [v1 = 81, v2 = 2285], [v1 = 82, v2 = 2285], [v1 = 83, v2 = 2368], [v1 = 84, v2 = 2452], [v1 = 85, v2 = 2452], [v1 = 86, v2 = 2538], [v1 = 87, v2 = 2625], [v1 = 88, v2 = 2625]
funcional: [Par[v1=0, v2=44], Par[v1=1, v2=44], Par[v1=2, v2=44], Par[v1=3, v2=46], Par[v1=4, v2=49], Par[v1=5, v2=49], Par[v1=6, v2=54], Par[v1=7, v2=60], Par[v1=8, v2=60], Par[v1=9, v2=68], Par[v1=10, v2=77], Par[v1=11, v2=77], Par[v1=12, v2=88], Par[v1=13, v2=100], Par[v1=14, v2=100], Par[v1=15, v2=114], Par[v1=16, v2=129], Par[v1=17, v2=129], Par[v1=18, v2=146], Par[v1=19, v2=164], Par[v1=20, v2=164], Par[v1=21, v2=184], Par[v1=22, v2=205], Par[v1=23, v2=205], Par[v1=24, v2=228], Par[v1=25, v2=252], Par[v1=26, v2=252], Par[v1=27, v2=278], Par[v1=28, v2=305], Par[v1=29, v2=305], Par[v1=30, v2=334], Par[v1=31, v2=364], Par[v1=32, v2=364], Par[v1=33, v2=396], Par[v1=34, v2=429], Par[v1=35, v2=429], Par[v1=36, v2=464], Par[v1=37, v2=500], Par[v1=38, v2=500], Par[v1=39, v2=538], Par[v1=40, v2=577], Par[v1=41, v2=577], Par[v1=42, v2=618], Par[v1=43, v2=660], Par[v1=44, v2=660], Par[v1=45, v2=704], Par[v1=46, v2=749], Par[v1=47, v2=749], Par[v1=48, v2=796], Par[v1=49, v2=844], Par[v1=50, v2=844], Par[v1=51, v2=894], Par[v1=52, v2=945], Par[v1=53, v2=945], Par[v1=54, v2=998], Par[v1=55, v2=1052], Par[v1=56, v2=1052], Par[v1=57, v2=1108], Par[v1=58, v2=1165], Par[v1=59, v2=1165], Par[v1=60, v2=1224], Par[v1=61, v2=1284], Par[v1=62, v2=1284], Par[v1=63, v2=1346], Par[v1=64, v2=1409], Par[v1=65, v2=1409], Par[v1=66, v2=1474], Par[v1=67, v2=1540], Par[v1=68, v2=1540], Par[v1=69, v2=1608], Par[v1=70, v2=1677], Par[v1=71, v2=1677], Par[v1=72, v2=1748], Par[v1=73, v2=1820], Par[v1=74, v2=1820], Par[v1=75, v2=1894], Par[v1=76, v2=1969], Par[v1=77, v2=1969], Par[v1=78, v2=2046], Par[v1=79, v2=2124], Par[v1=80, v2=2124], Par[v1=81, v2=2204], Par[v1=82, v2=2285], Par[v1=83, v2=2285], Par[v1=84, v2=2368], Par[v1=85, v2=2452], Par[v1=86, v2=2452], Par[v1=87, v2=2538], Par[v1=88, v2=2625]]
Recursivo: , [v1 = 0, v2 = 44]
EntradaPar[v1=36, v2=45]

Iterativo (While): , [v1 = 0, v2 = 36], [v1 = 1, v2 = 36], [v1 = 2, v2 = 38], [v1 = 3, v2 = 41], [v1 = 4, v2 = 41], [v1 = 5, v2 = 46], [v1 = 6, v2 = 52], [v1 = 7, v2 = 52], [v1 = 8, v2 = 60], [v1 = 9, v2 = 69], [v1 = 10, v2 = 69], [v1 = 11, v2 = 80], [v1 = 12, v2 = 92], [v1 = 13, v2 = 92], [v1 = 14, v2 = 106], [v1 = 15, v2 = 121], [v1 = 16, v2 = 121], [v1 = 17, v2 = 138], [v1 = 18, v2 = 156], [v1 = 19, v2 = 156], [v1 = 20, v2 = 176], [v1 = 21, v2 = 197], [v1 = 22, v2 = 197], [v1 = 23, v2 = 220], [v1 = 24, v2 = 244], [v1 = 25, v2 = 244], [v1 = 26, v2 = 270], [v1 = 27, v2 = 297], [v1 = 28, v2 = 297], [v1 = 29, v2 = 326], [v1 = 30, v2 = 356], [v1 = 31, v2 = 356], [v1 = 32, v2 = 388], [v1 = 33, v2 = 421], [v1 = 34, v2 = 421], [v1 = 35, v2 = 456], [v1 = 36, v2 = 492], [v1 = 37, v2 = 492], [v1 = 38, v2 = 530], [v1 = 39, v2 = 569], [v1 = 40, v2 = 569], [v1 = 41, v2 = 610], [v1 = 42, v2 = 652], [v1 = 43, v2 = 652], [v1 = 44, v2 = 696]
funcional: [Par[v1=0, v2=36], Par[v1=1, v2=36], Par[v1=2, v2=36], Par[v1=3, v2=38], Par[v1=4, v2=41], Par[v1=5, v2=41], Par[v1=6, v2=46], Par[v1=7, v2=52], Par[v1=8, v2=52], Par[v1=9, v2=60], Par[v1=10, v2=69], Par[v1=11, v2=69], Par[v1=12, v2=80], Par[v1=13, v2=92], Par[v1=14, v2=92], Par[v1=15, v2=106], Par[v1=16, v2=121], Par[v1=17, v2=121], Par[v1=18, v2=138], Par[v1=19, v2=156], Par[v1=20, v2=156], Par[v1=21, v2=176], Par[v1=22, v2=197], Par[v1=23, v2=197], Par[v1=24, v2=220], Par[v1=25, v2=244], Par[v1=26, v2=244], Par[v1=27, v2=270], Par[v1=28, v2=297], Par[v1=29, v2=297], Par[v1=30, v2=326], Par[v1=31, v2=356], Par[v1=32, v2=356], Par[v1=33, v2=388], Par[v1=34, v2=421], Par[v1=35, v2=421], Par[v1=36, v2=456], Par[v1=37, v2=492], Par[v1=38, v2=492], Par[v1=39, v2=530], Par[v1=40, v2=569], Par[v1=41, v2=569], Par[v1=42, v2=610], Par[v1=43, v2=652], Par[v1=44, v2=652]]
Recursivo: , [v1 = 0, v2 = 36]
EntradaPar[v1=-78, v2=9]
Iterativo (While): , [v1 = 0, v2 = -78], [v1 = 1, v2 = -78], [v1 = 2, v2 = -76], [v1 = 3, v2 = -73], [v1 = 4, v2 = -73], [v1 = 5, v2 = -68], [v1 = 6, v2 = -62], [v1 = 7, v2 = -62], [v1 = 8, v2 = -54]
funcional: [Par[v1=0, v2=-78], Par[v1=1, v2=-78], Par[v1=2, v2=-78], Par[v1=3, v2=-76], Par[v1=4, v2=-73], Par[v1=5, v2=-73], Par[v1=6, v2=-68], Par[v1=7, v2=-62], Par[v1=8, v2=-62]]
Recursivo: , [v1 = 0, v2 = -78]
EntradaPar[v1=89, v2=66]
Iterativo (While): , [v1 = 0, v2 = 89], [v1 = 1, v2 = 89], [v1 = 2, v2 = 91], [v1 = 3, v2 = 94], [v1 = 4, v2 = 94], [v1 = 5, v2 = 99], [v1 = 6, v2 = 105], [v1 = 7, v2 = 105], [v1 = 8, v2 = 113], [v1 = 9, v2 = 122], [v1 = 10, v2 = 122], [v1 = 11, v2 = 133], [v1 = 12, v2 = 145], [v1 = 13, v2 = 145], [v1 = 14, v2 = 159], [v1 = 15, v2 = 174], [v1 = 16, v2 = 174], [v1 = 17, v2 = 191], [v1 = 18, v2 = 209], [v1 = 19, v2 = 209], [v1 = 20, v2 = 229], [v1 = 21, v2 = 250], [v1 = 22, v2 = 250], [v1 = 23, v2 = 273], [v1 = 24, v2 = 297], [v1 = 25, v2 = 297], [v1 = 26, v2 = 323], [v1 = 27, v2 = 350], [v1 = 28, v2 = 350], [v1 = 29, v2 = 379], [v1 = 30, v2 = 409], [v1 = 31, v2 = 409], [v1 = 32, v2 = 441], [v1 = 33, v2 = 474], [v1 = 34, v2 = 474], [v1 = 35, v2 = 509], [v1 = 36, v2 = 545], [v1 = 37, v2 = 545], [v1 = 38, v2 = 583], [v1 = 39, v2 = 622], [v1 = 40, v2 = 622], [v1 = 41, v2 = 663], [v1 = 42, v2 = 705], [v1 = 43, v2 = 705], [v1 = 44, v2 = 749], [v1 = 45, v2 = 794], [v1 = 46, v2 = 794], [v1 = 47, v2 = 841], [v1 = 48, v2 = 889], [v1 = 49, v2 = 889], [v1 = 50, v2 = 939], [v1 = 51, v2 = 990], [v1 = 52, v2 = 990], [v1 = 53, v2 = 1043], [v1 = 54, v2 = 1097], [v1 = 55, v2 = 1097], [v1 = 56, v2 = 1153], [v1 = 57, v2 = 1210], [v1 = 58, v2 = 1210], [v1 = 59, v2 = 1269], [v1 = 60, v2 = 1329], [v1 = 61, v2 = 1329], [v1 = 62, v2 = 1391], [v1 = 63, v2 = 1454], [v1 = 64, v2 = 1454], [v1 = 65, v2 = 1519]
funcional: [Par[v1=0, v2=89], Par[v1=1, v2=89], Par[v1=2, v2=89], Par[v1=3, v2=91], Par[v1=4, v2=94], Par[v1=5, v2=94], Par[v1=6, v2=99], Par[v1=7, v2=105], Par[v1=8,

v2=105], Par[v1=9, v2=113], Par[v1=10, v2=122], Par[v1=11, v2=122], Par[v1=12,
v2=133], Par[v1=13, v2=145], Par[v1=14, v2=145], Par[v1=15, v2=159], Par[v1=16,
v2=174], Par[v1=17, v2=174], Par[v1=18, v2=191], Par[v1=19, v2=209], Par[v1=20,
v2=209], Par[v1=21, v2=229], Par[v1=22, v2=250], Par[v1=23, v2=250], Par[v1=24,
v2=273], Par[v1=25, v2=297], Par[v1=26, v2=297], Par[v1=27, v2=323], Par[v1=28,
v2=350], Par[v1=29, v2=350], Par[v1=30, v2=379], Par[v1=31, v2=409], Par[v1=32,
v2=409], Par[v1=33, v2=441], Par[v1=34, v2=474], Par[v1=35, v2=474], Par[v1=36,
v2=509], Par[v1=37, v2=545], Par[v1=38, v2=545], Par[v1=39, v2=583], Par[v1=40,
v2=622], Par[v1=41, v2=622], Par[v1=42, v2=663], Par[v1=43, v2=705], Par[v1=44,
v2=705], Par[v1=45, v2=749], Par[v1=46, v2=794], Par[v1=47, v2=794], Par[v1=48,
v2=841], Par[v1=49, v2=889], Par[v1=50, v2=889], Par[v1=51, v2=939], Par[v1=52,
v2=990], Par[v1=53, v2=990], Par[v1=54, v2=1043], Par[v1=55, v2=1097], Par[v1=56,
v2=1097], Par[v1=57, v2=1153], Par[v1=58, v2=1210], Par[v1=59, v2=1210], Par[v1=60,
v2=1269], Par[v1=61, v2=1329], Par[v1=62, v2=1329], Par[v1=63, v2=1391], Par[v1=64,
v2=1454], Par[v1=65, v2=1454]]
Recursivo: , [v1 = 0, v2 = 89]
##