Memoria simulacro PI1 Juan Orellana Carretero (no están terminadas las funciones)

**Ejercicio 1**

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 {

public static record EnteroCadena(Integer a, String s) {

public static EnteroCadena of(Integer a, String s) {

return new EnteroCadena(a, s);

}

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>> ejercicio1Terativo(Integer varA, String varB, Integer varC, String

varD, Integer varE){

Map<Integer, List<String>> ac = new HashMap<>();

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

EnteroCadena nx = EnteroCadena.of(e.a+2,

e.a()%3==0?

e.s()+e.a().toString():

e.s().substring(e.a()%e.s().length()));

while(nx.a<varC) {

if(nx.s.length()<varE) {

List<String> ls;

Integer key = nx.s.length();

if(ac.containsKey(key)) {

ac.get(key).add(nx.s + varD);

}else {

ls = new ArrayList<>();

ac.put(key, ls);

ls.add(nx.s + varD);

}

}

}

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);

EnteroCadena nx = EnteroCadena.of(e.a+2,

e.a()%3==0?

e.s()+e.a().toString():

e.s().substring(e.a()%e.s().length()));

if(nx.a<varC) {

if(nx.s.length()<varE) {

List<String> ls;

Integer key = nx.s.length();

if(ac.containsKey(key)) {

ac.get(key).add(nx.s + varD);

}else {

ls = new ArrayList<>();

ac.put(key, ls);

ls.add(nx.s + varD);

}

}

}

return ac;

}

}

**Ejercicio 2**

package ejercicios;

public class Ejercicio2 {

public static Integer ejercicio2RecursivoNoFinal(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 + *ejercicio2RecursivoNoFinal*(a-1, b/2, s.substring(b%s.length(), a%s.length()));

}else {

r = a \* b + *ejercicio2RecursivoNoFinal*(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);

}else if(a<2||b<2) {

r = s.length()+a+b;

}else if(a%s.length() < b%s.length()) {

r = *recFinal*(a-1, b/2, s.substring(b%s.length(), a%s.length()), a+b);

}else {

r = *recFinal*(a/2, b-1 , s.substring(b%s.length(), a%s.length()), a\*b);

}

return r;

}

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

public static Integer ejercicio2Iterativo(Integer a, Integer b, String s) {

return null;

}

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() {

if(a%s.length()<b%s.length()) {

return of(ac + )

}

return of(ac + String.format("%d", a+b), a/2, b-2);

}

}

public static Integer ejercicio2Funcional(Integer a, Integer b, String s) {

Tupla t = Stream.iterate(Tupla.first(a,b), e->e.next())

.filter(e -> e.a()<5 || e.b()<5)

.findFirst()

.get();

return t.ac() + String.format("(%d)", t.a()\*t.b());

}

public static Integer ejercicio2Funcional(Integer a, Integer b, String s) {

return null;

}

\*/

}

}

}

}

**Ejercicio 3**

package ejercicios;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Iterator;

import java.util.List;

import us.lsi.geometria.Punto2D;

import us.lsi.geometria.Punto2D.Cuadrante;

public class Ejercicio3 {

public static List<Double> ejercicio3(Iterator<Double> it1,

Iterator<Double> it2, Comparator<Double> comp){

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

Double e1 = (it1.hasNext()?it1.next():null);

Double e2 = (it2.hasNext()?it2.next():null);

while(e1 != null && e2 != null) {

if(Punto2D.of(e1, e2).getCuadrante() == Cuadrante.PRIMER\_CUADRANTE ||

Punto2D.of(e1, e2).getCuadrante() == Cuadrante.TERCER\_CUADRANTE) {

if (comp.compare(e1, e2)<0) {

ls.add(e1);

e1 = (it1.hasNext()?it1.next():null);

}else {

ls.add(e2);

e2 = (it2.hasNext()?it2.next():null);

}

}else {

e1 = it1.next();

e2 = it2.next(); }

}

while (e1!=null) {

ls.add(e1);

e1 = (it1.hasNext()?it1.next():null);

}

while (e2!=null) {

ls.add(e2);

e2 = (it2.hasNext()?it2.next():null);

}

return ls;

}

}

**Ejercicio 4**

package ejercicios;

import java.util.HashMap;

import java.util.Map;

import us.lsi.common.IntTrio;

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-%a)", 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-%a)", 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 = null;

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-%a)", 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(a/2, b-2, c/2)) + "/" + m.get(IntTrio.of(a/3, b-1, c/3)) + ")";

}

m.put(IntTrio.of(i,j,k), r);

}

}

}

return m.get(IntTrio.of(a,b,c));

}

}