**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**Высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Математической кибернетики и информационных технологий

**Отчет по лабораторной работе**

По дисциплине «СиАОД»

Выполнил: студент группы БВТ1801

Задоркин Максим Александрович

Руководитель:

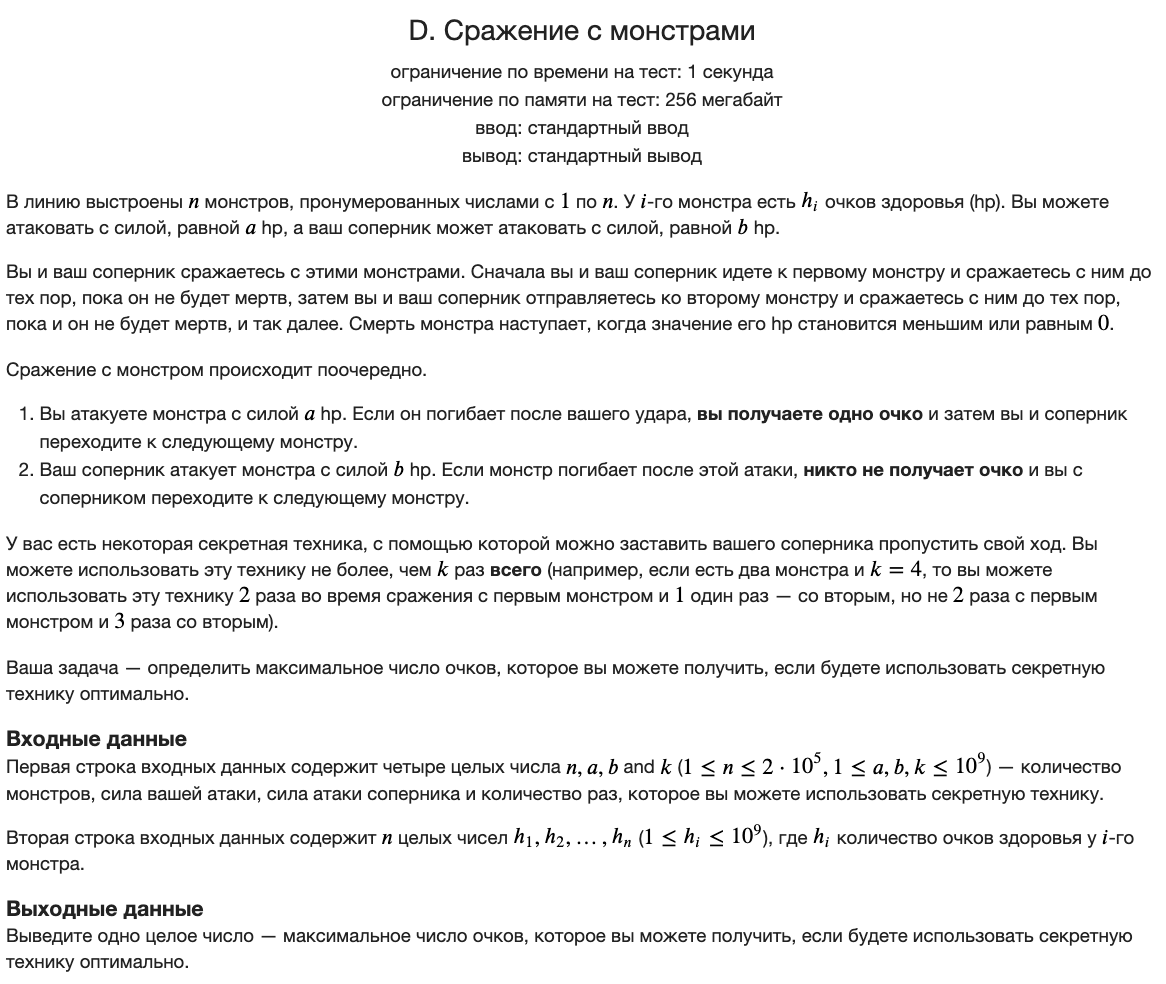
Кутейников Иван Алексеевич

Москва 2020

**Задача:**Решить несколько задач (не менее 2) уровнем не ниже 1300-1400 с сайта codeforces.com

**Ход работы:**

Код так же можно посмотреть на моем github: https://github.com/MaximZar/SaDPA/tree/master/codeForces

**Задача №1**  
Условие задачи: <https://codeforces.com/problemset/problem/1296/D>  
Сложность: 1500  


Код решения и пояснение к решению:

import **java.util.ArrayList**;

import **java.util.Collections**;

import **java.util.Scanner**;

*// https://codeforces.com/problemset/problem/1296/D*

**public** **class** Task1 {

**public** **static** **void** main(**String**[] args) {

**Scanner** input = **new** Scanner(System.in);

**int** n = input.nextInt(); *// количество монстров*

**double** a = input.nextInt(); *// моя сила удара*

**double** b = input.nextInt(); *// сила удара соперника*

**int** k = input.nextInt(); *// количество возможных использований секретной техники*

**double** sumDamage = a + b; *// общая сила удара*

**ArrayList**<**Integer**> monsters = **new** **ArrayList**<**Integer**>(); *// массив для монстров, которых не победили(дальше поподробнее)*

**int** countWin = 0; *// количество возможных выигрышных партий, очков*

**for** (**int** i = 1; i <= n; i += 1) {

**double** monster = input.nextInt(); *// очередной монстр*

**double** remainingDamage = monster % sumDamage; *// остаток урона после серии ударов*

*// remainingDamage = по сути ищем сколько останется хп перед итоговым боем*

*// или победил ли его наш соперник. Далее станет яснее*

*// последний удар сделали не мы, поэтому этот случай можно обработать*

*// для этого откатимся на 1 пару ходов. А потом будем использовать технику.*

**if** (remainingDamage == 0) {

*// количество проведенных секретных техник понадобится*

*// (на основе оставшегося hp у монстра при откате на 1 пару ходов):*

**int** amountOfSecretTechnology = (**int**) Math.ceil(b / a);

monsters.add(amountOfSecretTechnology);

**continue**;

}

**if** (remainingDamage <= a) { *// проверяем сможем ли именно МЫ его "добить" c 1 удара*

countWin += 1; *// прибавляем очко за победу*

**continue**;

}

*/\* в данном случае количество необходимых проведенных техник вычисляется несколько иначе*

*\* учитываем то, что монстр не был добит (remainingDamage). А значит нужно теперь пользоваться техниками*

*\* чтобы добить первее без отката на шаг назад*

*\*/*

**int** amountOfSecretTechnology = (**int**) Math.ceil((remainingDamage - a) / a);

monsters.add(amountOfSecretTechnology);

}

Collections.sort(monsters); *// сортируем чтобы выбрать первых двух с самым малым количеством*

*// использования секретных техник для выигрыша*

**int** index = 0; *// просто счетчик индексов*

**int** sizeOfMonsters = monsters.size();

**while** (k > 0) {

**if** (index >= sizeOfMonsters) **break**;

*// берем слабенького монстра*

**int** monster = monsters.get(index);

*/\* если запас техник остался то минусуем и добавляем очко выигрыша*

*\* если нет то прекращаем,*

*\* так как дальше монстры с еще большим количеством*

*\* необходимых техник для победы и смысла смотреть нет*

*\*/*

**if** (k - monster >= 0) {

k -= monster;

countWin += 1;

}

**else** **break**;

index += 1;

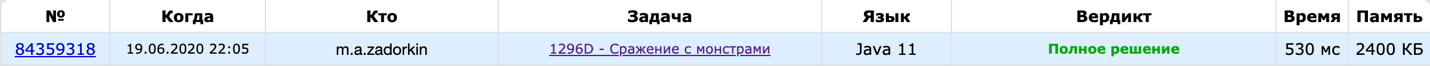
}

System.out.println(countWin);

}

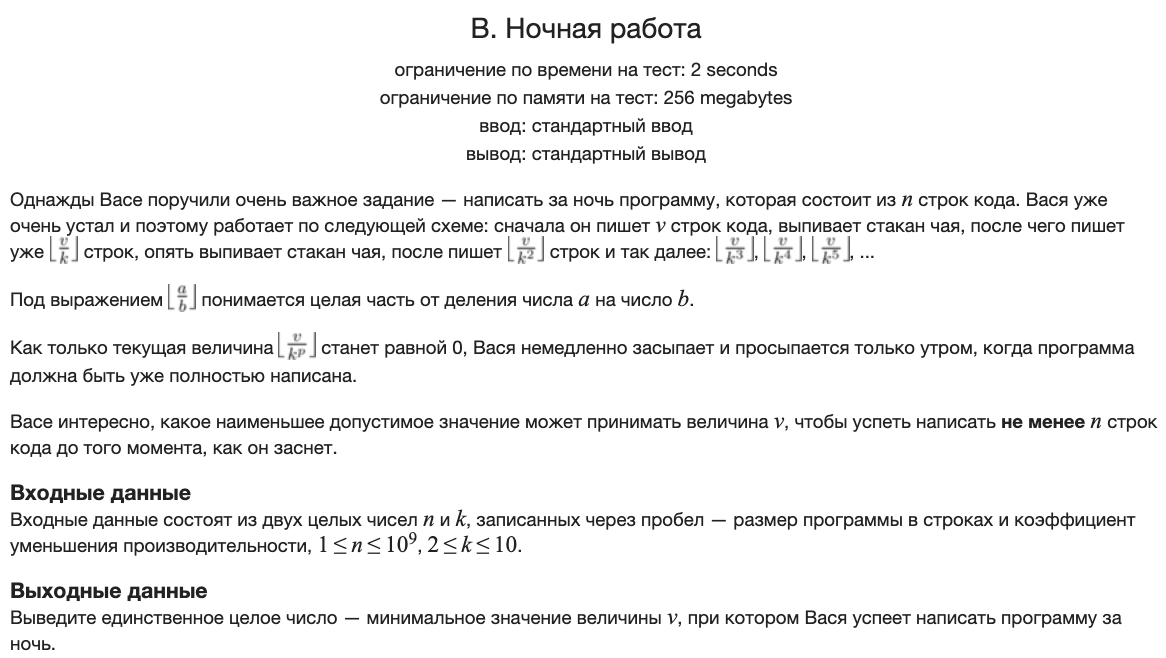
}

Результат тестирования:



**Задача 2:**

Условие задачи: <https://codeforces.com/problemset/problem/165/B>  
Сложность: 1500



Код задачи и пояснение к решению:

import **java.util.Scanner**;

*// https://codeforces.com/problemset/problem/165/B*

**public** **class** Task2 {

**public** **static** **void** main(**String**[] args) {

**Scanner** input = **new** Scanner(System.in);

**int** n = input.nextInt(); *// строк нужно написать*

**int** k = input.nextInt(); *// коэффициент уменьшения*

*/\* число v будет располагаться в пределах 1 .. n*

*\* а значит можем воспользоваться бинарным поиском*

*\* чтобы эффективнее найти нужную цифру*

*\* для реализации бинарного поиска создадим leftPoint и rightPoint*

*\* чтобы помечать наши отрезки*

*\*/*

**int** leftPoint = 1;

**int** rightPoint = n;

**int** v = n;

**while** (leftPoint < rightPoint) {

**int** pivot = leftPoint + (rightPoint - leftPoint) / 2; *// v на этой итерации,*

*// середина нашего отрезка текущего*

**int** countLinesOfCode = 0; *// высчитаем при данном v количество строк*

**int** iter = pivot;

**while** (iter > 0) {

countLinesOfCode += iter;

iter /= k; *// сделали формулу рекурентной*

}

**if** (countLinesOfCode >= n) { *// проверяем написали ли мы необходимое количество строк кода*

v = pivot; *// если написали то ставим новый результат как наименьший*

rightPoint = pivot; *// смещаем правую точку отрезка. сужаем вдвое область поиска*

}

**else** leftPoint = pivot + 1; *// если с этим числом не получилось то перемистимся на правую часть отрезка*

*// для это сдвинем левую точку на половину отрезка*

}

System.out.println(v);

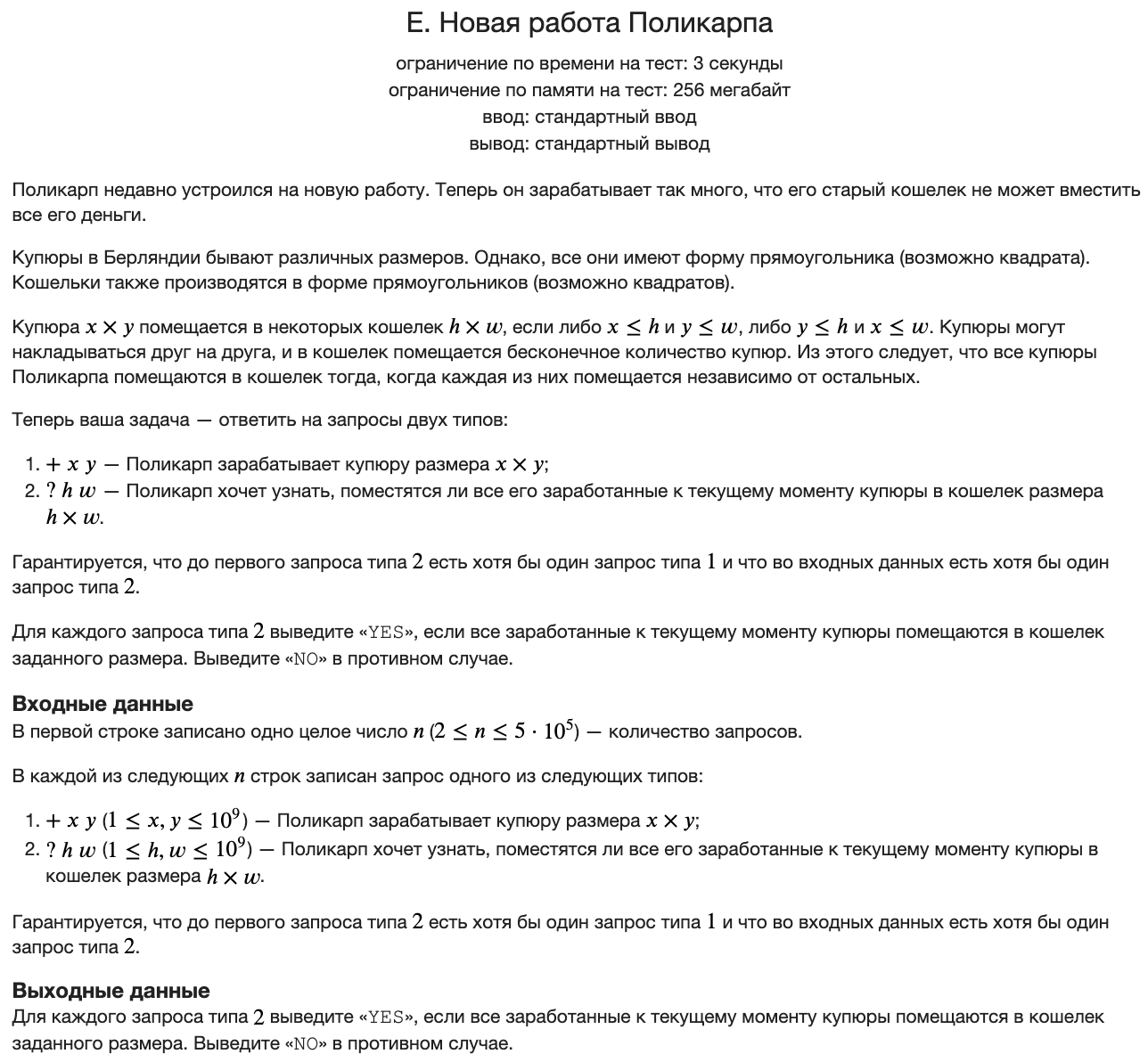
}

}

Результат тестирования:



**Задача 3:**

Условие задачи: https://codeforces.com/problemset/problem/1101/E  
Сложность: 1500

Код задачи и пояснение к решению:

import **java.util.\***;

import **java.io.\***;

*// https://codeforces.com/problemset/problem/1101/E*

**public** **class** Task3 {

**public** **static** **void** main(**String**[] args) {

**Scanner** input = **new** Scanner(System.in);

**PrintWriter** output = **new** PrintWriter(System.out);

**String** request;

**int** a, b, x, y, h=0, w=0;

**int** n = input.nextInt(); *// количество запросов*

**for** (**int** i = 0; i < n; i += 1) {

*// получаем очередной запрос*

request = String.valueOf(input.next().charAt(0));

a = input.nextInt();

b = input.nextInt();

*/\* чтобы не проверять дважды условия (так как мы можем положить купюру двояко)*

*\* отсортируем входящие параметры*

*\* так как они не влияют на итоговый результат*

*\*/*

x = Math.min(a, b);

y = Math.max(a, b);

**if** (request.equals("?")) { *// запрос вопросительный?*

*// проверяем наши параметры*

**if**(x >= h && y >= w) output.println("YES");

**else** output.println("NO");

}

**else** {

*/\* чтобы каждый раз не вычислять размеры кошелька*

*\* будем изменять его размеры при каждом + запросе*

*\* то есть будем узнать длинны его сторон каждый раз*

*\* и если в этом запросе сторона больше то изменяем старые размеры*

*\*/*

h = Math.max(h, x);

w = Math.max(w, y);

}

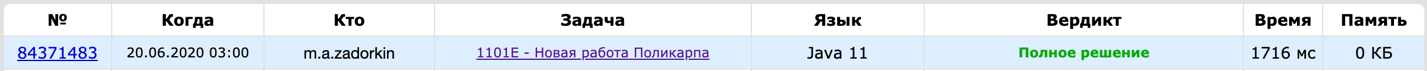
}

output.flush();

}

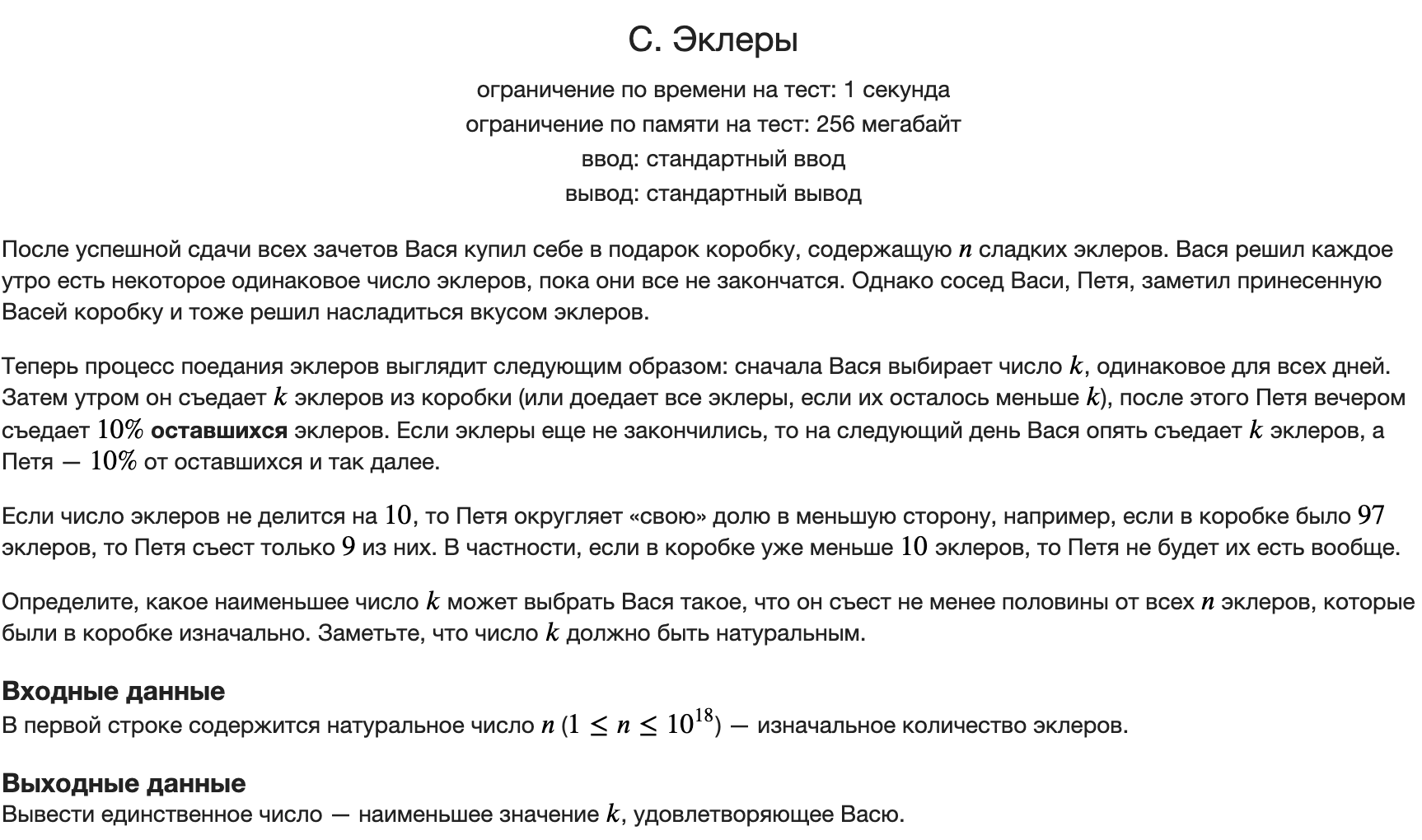
}

Результат тестирования:

****

**Задача 4:**

Условие задачи: <https://codeforces.com/problemset/problem/991/C>  
Сложность: 1500



Код задачи и пояснение к решению:

import **java.util.Scanner**;

*// https://codeforces.com/problemset/problem/991/C*

**public** **class** Task4 {

**public** **static** **void** main(**String** args[]) {

**long** eclairs = **new** Scanner(System.in).nextLong(); *// количество эклеров всего*

*/\* число k будет распологаться в промежутке*

*\* 1 .. eclairs, а это значит*

*\* что мы можем найти это число быстрее,*

*\* используя бинарный поиск*

*\*/*

**long** startPoint = 1;

**long** endPoint = eclairs;

**long** k = eclairs;

**while**(startPoint <= endPoint) {

**long** pivot = (startPoint + endPoint) / 2; *// выбираем середину нашего отрезка*

*// найдем число k для текущей pivot*

**long** countEclairs = eclairs;

**long** sum = 0;

**while**(countEclairs > 0) {

**long** day = Math.min(pivot, countEclairs); *// день*

sum += day; *// доля Васи (накапливаем, чтобы узнать сколько он съест всего)*

countEclairs -= day; *// доля Васи вычитаем*

countEclairs -= countEclairs / 10; *// Петину долю вычитаем*

}

**if** (sum \* 2 >= eclairs) { *// проверяем больше/равно половине от всех эклеров съел Вася или нет*

*// если да то pivot теперь новое k*

*// а так же сужаем отрезок вдвое*

*// и продолжаем идти к минимальному к*

k = pivot;

endPoint = pivot - 1;

}

*// к остается прежним, сужаем отрезок и переходим на правую сторону от середины прошлого отрезка*

**else** startPoint = pivot + 1;

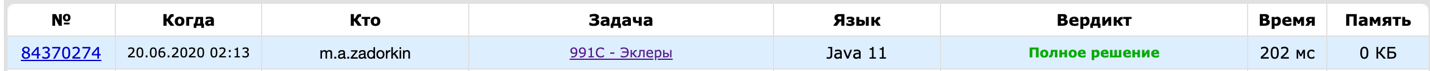
}

System.out.println(k);

}

}

Результат тестирования:



**Вывод:**  в процессе работы были изучены несколько примеров олимпиадных задач по алгоритмам, реализована работа этих программ и протестирована на платформе codeforces.