

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)



Отчет по индивидуальному учебному плану

---

## Фундаментальные алгоритмы

---

Студенты:

Макаров Никита, 8о-406Б  
Якименко Антон, 8о-406Б

Руководитель:

Зайцев В.Е.

Москва  
2016

# Содержание

<b>1</b>	<b>Личные отчеты</b>	<b>2</b>
<b>2</b>	<b>Журнал по командным контестам</b>	<b>4</b>
2.1	OpenCup GrandPrix of Ukraine . . . . .	4
2.2	OpenCup GrandPrix of Japan . . . . .	10
2.3	OpenCup GrandPrix of Eurasia) . . . . .	19
2.4	OpenCup GrandPrix of SPb . . . . .	32
2.5	ACM-ICPC Московский четвертьфинал . . . . .	38
2.6	OpenCup GrandPrix of Yekaterinburg . . . . .	39
2.7	OpenCup GrandPrix of Siberia . . . . .	47
2.8	OpenCup GrandPrix of Europe . . . . .	58
2.9	OpenCup GrandPrix of Peterhof . . . . .	66
2.10	OpenCup Grand Prix of Asia . . . . .	74
2.11	OpenCup GrandPrix of Saratov . . . . .	80
2.12	OpenCup GrandPrix of China . . . . .	86
2.13	OpenCup GrandPrix of Bashkortostan . . . . .	95
2.14	OpenCup GrandPrix of Tatarstan . . . . .	110
2.15	OpenCup GrandPrix of Baltics . . . . .	114
2.16	OpenCup GrandPrix of Moscow . . . . .	122
2.17	Vekua Cup 2016 . . . . .	126
<b>3</b>	<b>Журнал по личным контестам Макарова Н.А.</b>	<b>127</b>
3.1	Codeforces Round 320 Div 2 . . . . .	127
3.2	Codeforces Round 323 Div 2 . . . . .	131
3.3	Codeforces Round 324 Div 2 . . . . .	135
3.4	Codeforces Beta Round 40 Div 2 . . . . .	137
3.5	Codeforces Round 258 Div 2 . . . . .	139
3.6	Codeforces Round 196 Div 2 . . . . .	142
3.7	Codeforces Round 304 Div 2 . . . . .	144
3.8	Codeforces Round 277 Div 2 . . . . .	146
3.9	Codeforces Round 103 Div 2 . . . . .	148
3.10	VK Cup 2016 - Квалификация 2 . . . . .	151
3.11	VK Cup 2016 - Квалификация 1 . . . . .	156
3.12	VK Cup 2016 - Раунд 1 . . . . .	158
3.13	VK Cup 2016 - Уайлд-кард раунд 1 . . . . .	162
3.14	Vekua Cup 2016 Личный этап . . . . .	164
<b>4</b>	<b>Журнал по личным контестам Якименко А.В.</b>	<b>169</b>
4.1	Vekua Cup 2016 Личный этап . . . . .	169

# 1 Личные отчеты

Отчет о работе студента Макарова Н.А. по индивидуальному учебному плану в VII-VIII семестрах 2015-2016 учебного года.

№	Дата	Конкурс	Место проведения	Кол-во участников	Решено задач	Задач на участника
1	13.09.2015	OpenCup GP of Ukraine Div 2	МАИ	3	4	1.33
2	16.09.2015	Codeforces Round 320 Div 2	Дом	1	2	2
3	27.09.2015	OpenCup GP of Japan Div 2	МАИ	3	5	1.66
4	03.10.2015	Codeforces Round 323 Div 2	Дом	1	2	2
5	04.10.2015	OpenCup GP of Eurasia Div 2	МАИ	3	4	1.33
6	06.10.2015	Codeforces Round 324 Div 2	Дом	1	1	1
7	11.10.2015	OpenCup GP of SPb	МАИ	3	2	0.66
8	13.10.2015	Codeforces Round 40 Div 2	Дом	1	1	1
9	13.10.2015	Codeforces Round 258 Div 2	Дом	1	1	1
10	13.10.2015	Codeforces Round 196 Div 2	Дом	1	1	1
11	13.10.2015	Codeforces Round 304 Div 2	Дом	1	1	1
12	13.10.2015	Codeforces Round 277 Div 2	Дом	1	1	1
13	14.10.2015	Codeforces Round 103 Div 2	Дом	1	1	1
14	18.10.2015	ACM ICPC 1/4 Final	НИУ ВШЭ	3	2	0.66
15	01.11.2015	OpenCup GP of YKb Div 2	МАИ	3	4	1.33
16	08.11.2015	OpenCup GP of Siberia Div 2	МАИ	3	6	2
17	22.11.2015	OpenCup GP of Europe Div 2	МАИ	3	3	1
18	20.12.2015	OpenCup GP of Peterhof Div 2	МАИ	3	4	1.33
19	31.01.2016	OpenCup GP of Asia Div 2	МАИ	3	3	1
20	07.02.2016	OpenCup GP of Saratov Div 2	МАИ	3	3	1
21	14.02.2016	OpenCup GP of China Div 2	МАИ	3	4	1.33
22	28.02.2016	OpenCup GP of BSHK Div 2	МАИ	3	7	2.33
23	13.03.2016	OpenCup GP of Tatarstan Div 2	МАИ	3	2	0.66
24	20.03.2016	OpenCup GP of Baltics Div 2	МАИ	3	3	1
25	21.03.2016	VK Cup 2016 Квалификация 1	Дом	1	1	1
26	21.03.2016	VK Cup 2016 Квалификация 2	Дом	1	2	2
27	28.03.2016	VK Cup 2015 Раунд 1	Дом	1	1	1
28	03.04.2016	OpenCup GP of Moscow Div 2	МАИ	3	2	0.66
29	10.04.2016	VK Cup 2016 Уайлд-карт 1	Дом	1	1	1
30	23.04.2016	Vekua Cup 2016 Личный	МФТИ	1	2	2
31	24.04.2016	Vekua Cup 2016 Командный	МФТИ	3	1	0.33

Итого: 31 конкурс, 77 решенных задач. Личный вклад  $\approx 36.61$  задач (командные + личные).

Студент:\_\_\_\_\_ Макаров Н.А.

Руководитель:\_\_\_\_\_ Зайцев В.Е.

Отчет о работе студента Якименко А.В. по индивидуальному учебному плану в VII-VIII семестрах 2015-2016 учебного года.

№	Дата	Контест	Место проведения	Кол-во участников	Решено задач	Задач на участника
1	13.09.2015	OpenCup GP of Ukraine Div 2	МАИ	3	4	1.33
2	27.09.2015	OpenCup GP of Japan Div 2	МАИ	3	5	1.66
3	04.10.2015	OpenCup GP of Eurasia Div 2	МАИ	3	4	1.33
4	11.10.2015	OpenCup GP of SPb	МАИ	3	2	0.66
5	18.10.2015	ACM ICPC 1/4 Final	НИУ ВШЭ	3	2	0.66
6	01.11.2015	OpenCup GP of YKb Div 2	МАИ	3	4	1.33
7	08.11.2015	OpenCup GP of Siberia Div 2	МАИ	3	6	2
8	22.11.2015	OpenCup GP of Europe Div 2	МАИ	3	3	1
9	20.12.2015	OpenCup GP of Peterhof Div 2	МАИ	3	4	1.33
10	31.01.2016	OpenCup GP of Asia Div 2	МАИ	3	3	1
11	07.02.2016	OpenCup GP of Saratov Div 2	МАИ	3	3	1
12	14.02.2016	OpenCup GP of China Div 2	МАИ	3	4	1.33
13	28.02.2016	OpenCup GP of BSHK Div 2	МАИ	3	7	2.33
14	13.03.2016	OpenCup GP of Tatarstan Div 2	МАИ	3	2	0.66
15	20.03.2016	OpenCup GP of Baltics Div 2	МАИ	3	3	1
16	03.04.2016	OpenCup GP of Moscow Div 2	МАИ	3	2	0.66
17	23.04.2016	Vekua Cup 2016 Личный	МФТИ	1	2	2
18	24.04.2016	Vekua Cup 2016 Командный	МФТИ	3	1	0.33

Итого: 18 контестов, 61 решенная задача. Личный вклад  $\approx 21.61$  (командные + личные).

Студент: \_\_\_\_\_ Якименко А.В.

Руководитель: \_\_\_\_\_ Зайцев В.Е.

## 2 Журнал по командным контестам

### 2.1 OpenCup GrandPrix of Ukraine Div 2

#### Результаты

46.	MAI #6: Makarov, Rik, Yakimenko	-	+1 0:56	+ 1:12	-5 4:51	-	+ 0:24	+1 3:07	-	-	-	3	303	25%	0.11
-----	---------------------------------	---	------------	-----------	------------	---	-----------	------------	---	---	---	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10321](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10321)

## Задача C - Cool Numbers

Степан называет целое положительное число  $p$  весёлым, если  $p$  и число  $p_1$ , полученное прочтением его десятичной записи справа налево, являются различными простыми числами.

Напоминаем, что целое положительное число является простым, если оно не имеет целых делителей кроме единицы и самого себя.

По заданному  $K$  найдите  $K$ -е весёлое число.

### Input

Первая строка входа содержит одно целое число  $K$  ( $1 \leq K \leq 1000$ ).

### Output

Если  $K$ -е весёлое число не превосходит  $10^6$ , выведите его. Иначе выведите  $-1$ .

### Example

standard input	standard output
1	13

## Алгоритм

Для решения данной задачи можно написать генератор весёлых чисел, который посчитает первые 1000 весёлых чисел, а затем уже выводить ответ используя сгенерированные числа. Сложность решения  $O(1)$ .

## Исходный код

```
1 #include <iostream>
2 int main(int argc, const char * argv[]) {
3     int nums[] = {13, 17, .. (generated numbers) .., 70529, 70573};
4     int k;
5     std::cin >> k;
6     std::cout << nums[k - 1];
7     return 0;
8 }
```

## Задача D - Diagram

Дамблдор только что завершил сложный магический ритуал, для которого, помимо прочего, использовалась магическая диаграмма с  $N$  попарно различными символами, каждый из которых находится в определённой точке окружности.

Пока Дамблдор спал, Гарри Поттер нашёл диаграмму и хочет использовать её для того, чтобы выполнить домашнее задание по заклинаниям. Но для этого ему нужен правильный  $K$ -угольник.

Гарри знает, что длина окружности — целое число, делящееся на  $K$ . Также он знает расстояния между соседними по окружности символами — также целые числа. Он хочет выбрать  $K$  символов так, чтобы они были вершинами правильного  $K$ -угольника.

Напишите программу, которая определит, сможет ли Гарри это сделать.

### Input

Первая строка входе содержит целые числа  $N$  и  $K$  ( $3 \leq N \leq 10^5$ ,  $3 \leq K \leq 10^5$ ), за которыми следует монотонно возрастающая последовательность из  $N + 1$  целых чисел  $X_i$ , задающих расстояния по часовой стрелке от нулевого символа до  $i$ -го по дуге окружности ( $X_0 = 0$ , а  $X_N$  — дуга окружности;  $0 \leq X_i \leq 10^9$ ). Гарантируется, что  $X_N \bmod K = 0$ . Соседние числа во вводе разделены пробелами.

### Output

Если Гарри сможет выбрать  $K$  символов, находящихся в вершинах правильного  $K$ -угольника, выведите 1. Иначе выведите 0.

### Example

standard input	standard output
5 3 0 1 2 4 5 6	1

### Алгоритм

В этой задаче можно воспользоваться хитростью. Так как нужное количество чисел чтобы построить окружность не поместится в памяти, будем использовать битовый массив, показывающий наличие точки на окружности в заданных координатах. Затем просто пройдем по массиву и попытаемся собрать нужную последовательность. Сложность алгоритма  $O(N * K)$ .

## Исходный код

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #define ULL unsigned long long
5 int main() {
6     ULL *array = (ULL *)malloc(15500000 * sizeof(ULL));
7     ULL p;
8     for (p = 0; p < 15000000; p++) {
9         array[p] = 0;
10    }
11    ULL N, K;
12    scanf("%lld %lld ", &N, &K);
13    ULL i;
14    int r = 64;
15    for (i = 0; i < N; i++) {
16        ULL dist;
17        scanf("%lld ", &dist);
18        array[dist/r] |= (1LL << (dist % r));
19    }
20    ULL max_length;
21    scanf("%lld ", &max_length);
22    const ULL part_size = max_length / K;
23    for (i = 0; i < part_size; i++) {
24        ULL index = i / r;
25        int shift = i % r;
26        bool found = true;
27        if (array[index] & (1LL << shift)) {
28            ULL j;
29            for (j = i + part_size; j < max_length; j += part_size) {
30                ULL j_index = j / r;
31                int j_shift = j % r;
32                if (!(array[j_index] & (1LL << j_shift))) {
33                    found = false;
34                    break;
35                }
36            }
37            if (found) {
38                printf("1\n");
39                return 0;
40            }
41        }
42    }
43    printf("0\n");
44    return 0;
45 }
```

## Задача F - First And Last

По заданной ненулевой десятичной цифре  $a$  и десятичной цифре  $b$  найдите, существует ли такое неотрицательное  $n$ , что  $a$  является первой цифрой числа  $2^n$ , а  $b$  — последней. Если существует, выведите наименьшее  $n$  с таким свойством.

### Input

Вход состоит из двух целых чисел  $a$  и  $b$  ( $1 \leq a \leq 9$ ,  $0 \leq b \leq 9$ ) — заданных цифр.

### Output

Если существует такое неотрицательное целое  $n$ , что первая цифра  $2^n$  равна  $a$ , а вторая —  $b$ , выведите минимальное значение  $n$ , при котором это выполняется. Иначе выведите  $-1$ .

### Example

standard input	standard output
2 2	1
5 5	-1

## Алгоритм

Задача решается простым перебором. Перебираются все степени двойки до 1000(больше числа быть не может), если попадается подходящие под условие число, то выводим порядок степени и выводим. Если не попадается, то выводим -1.

## Исходный код

```
1 #!/usr/bin/perl
2 use bigint;
3 my($a, $b) = split " ", <>;
4 my $p = 1;
5 for my $n (0..1000) {
6     if (($a eq $b && $p eq $a) || $p =~ m/^$a.*?$b$/) {
7         print $n, "\n";
8         exit;
9     }
10    $p <= 1;
11 }
12 print "-1\n";
```

## Задача G - Game of Solitaire

Рассмотрим следующую разновидность пасьянса. Задана колода из  $N$  карт, пронумерованных последовательными целыми числами от 1 до  $N$ . Числа написаны на лицевой стороне карт. Карты лежат в ряд лицом вверх, карта 1 левее всех, карта  $N$  — правее всех.

Игрок выбирает целое положительное число  $K < N$ . После этого он перемещает первые  $K$  карт в конец ряда.

Например, если  $N = 6$  и игрок выбрал число 4 ( $K = 4$ ), расстановка после перемещения будет следующей: 5 6 1 2 3 4.

Далее игрок делает следующее. Он берёт самую левую карту, переворачивает её лицом вниз; пусть на этой карте написано число  $M$ ; тогда он переходит к  $M$ -й слева карте (включая и те, которые уже перевёрнуты) и повторяет ту же самую процедуру — переворачивает и переходит к карте, заданной номером на только что перевёрнутой. Как только игрок дошёл до карты, которая уже была перевёрнута, он заканчивает первый раунд. После этого игрок начинает следующий раунд: берёт самую левую неперевёрнутую карту, переворачивает её, переходит... и так далее, пока все карты не будут перевёрнуты.

По заданным  $N$  и  $M$  определите количество раундов, требуемое для завершения пасьянса.

### Input

Вход содержит два целых числа  $N$  и  $K$ ,  $1 \leq K < N \leq 10^9$ .

### Output

Выведите количество раундов, требуемое для завершения пасьянса.

### Example

standard input	standard output
6 4	2

## Алгоритм

В этой задаче нужно заметить, что ответом будет наибольший общий делитель входных чисел. НОД находим с помощью алгоритма Евклида. Сложность решения  $O(\log * \min(a, b))$

## Исходный код

```
1 #include <iostream>
2 using namespace std;
3 long gcd (long a, long b) {
4     while (b) {
5         a %= b;
6         swap(a, b);
7     }
8     return a;
9 }
10 int main() {
11     long n, k;
12     cin >> n >> k;
13     cout << gcd(n, k);
14     return 0;
15 }
```

## 2.2 OpenCup GrandPrix of Japan Div 2

### Результаты

36.	MAI #6: Makarov, Rik, Yakimenko	+	-	-	-	+	-	-	+	+	+	5	333	0%	0.11
-----	---------------------------------	---	---	---	---	---	---	---	---	---	---	---	-----	----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10322](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10322)

## Задача A - Where is the Boundary

Островное государство JAGония на некоторой планете вытянуто с запада на восток. В JAGонии присутствуют два вида культуры — западная и восточная. Регионы на востоке имеют в основном признаки восточной культуры, регионы на западе — западной; впрочем, провести чёткую границу довольно сложно.

Вам поручено оценить эту границу по имеющимся данным.

Более точно,

1. JAGония разделена на  $n$  префектур, расположенных с запада на восток. Можно считать, что префектуры соответствуют точкам  $1, 2, \dots, n$ , причём точка 1 — самая западная.
2. Собранные данные содержат по  $m$  признаков, каждый из которых определён для каждой префектуры и может принимать значения ‘E’ (восточная) или ‘W’ (западная) в зависимости от того, к какой культуре можно отнести население соответствующей префектуры по данному признаку.
3. Вы должны провести границу, которая минимизирует суммарную ошибку. То есть Вы должны минимизировать количество ‘W’ на территории восточнее границы и количество ‘E’ — на территории, расположенной западнее.
4. Границу можно проводить только по границе между префектурами.

В случае, если все префектуры относятся к восточной культуре, считать, что граница проходит между префектурами 0 и 1, если все относятся к западной — между  $n$  и  $n + 1$ . Если оптимальных границ несколько, выберите наиболее западную (то есть минимальную по значению выводимых чисел) из них.

### Input

Первая строка входа содержит два целых числа  $n$  ( $1 \leq n \leq 10^4$ ) и  $m$  ( $1 \leq m \leq 100$ ) — количество префектур и количество признаков. Каждая из последующих  $m$  строк содержит по  $n$  символов ‘E’ или ‘W’;  $j$ -й символ в  $i$ -й строке обозначает, что население в  $j$ -й префектуре относится по  $i$ -му признаку к восточной или западной культуре соответственно.

### Output

Выполните два целых числа — западную и восточную префектуры, между которыми проходит граница (в случае однородности выполните “виртуальную” нулевую или  $n + 1$ -ю префектуру первой или второй соответственно).

### Examples

standard input	standard output
2 1 WE	1 2
3 2 WWE WEE	1 2
3 1 WWW	3 4
3 1 WEW	1 2

### Алгоритм

В этой задаче нужно просто считать все данные и перебрать все значения ошибок, запоминая индексы минимальных значений. Сложность  $O(N * M)$ .

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <map>
4 using namespace std;
5 typedef struct {
6     int left;
7     int rigth;
8     int sum;
9 } error;
10 int main(int argc, const char * argv[]) {
11     int n, m;
12     cin >> n >> m;
13     vector< map<string, int>> jag(n);
14     for (int i = 0; i < n; i++) {
15         jag[i][ "w" ] = 0;
16         jag[i][ "e" ] = 0;
17     }
18     vector<error> errors (n + 1);
19     errors[0].left = 0;
20     errors[0].rigth = 0;
21     errors[0].sum = 0;
22     for (int i = 0; i < m; i++) {
23         string features;
24         cin >> features;
25         for (size_t j = 0; j < features.size(); j++) {
26             if (features[j] == 'W') {
27                 jag[j][ "w" ]++;
28                 errors[0].rigth++; // !!!!!!
29             } else {
30                 jag[j][ "e" ]++;
31             }
32         }
33     }
34     errors[0].sum = errors[0].left + errors[0].rigth;
35     int min_error = errors[0].rigth;
36     int min_error_index = -1;
37     for (int i = 1; i < n + 1; i++) {
38         errors[i].left = 0;
39         errors[i].rigth = 0;
40         errors[i].sum = 0;
41         errors[i].left = errors[i - 1].left + jag[i - 1][ "e" ];
42         errors[i].rigth = errors[i - 1].rigth - jag[i - 1][ "w" ];
43         errors[i].sum = errors[i].left + errors[i].rigth;
44         if (errors[i].sum < min_error) {
45             min_error = errors[i].sum;
46             min_error_index = i;
47         }
48     }
49     if (min_error_index == -1) {
50         cout << "0 1" << endl;
51     } else {
52         cout << min_error_index << " " << min_error_index + 1 << endl;
53     }
54     return 0;
55 }
```

## Задача G - Surface Area of Cubes

Таро играет в игру “Surface Area of Cubes”.

В этой игре дан параллелепипед  $A \times B \times C$ , составленный из  $A \times B \times C$  единичных кубиков. Центр каждого единичного кубика находится в целочисленной точке  $(x, y, z)$  ( $0 \leq x \leq A - 1$ ,  $0 \leq y \leq B - 1$ ,  $0 \leq z \leq C - 1$ ). После чего мастер убирает  $N$  различных единичных кубиков. Игрок после этого должен сообщить общую площадь поверхности получившегося объекта.

Операция удаления не меняет положения кубиков, которые не были удалены; удалены могут быть не только кубики, примыкающие к поверхности исходного параллелепипеда; получившийся объект может в результате представлять собой несколько несвязанных частей; в площадь поверхности входит и площадь “внутренних поверхностей”, недоступных извне, если таковые имеются.

Напишите программу, которая помогает игроку подсчитать требуемую площадь.

### Input

Первая строка входа содержит четыре целых числа  $A$ ,  $B$ ,  $C$  и  $N$  ( $1 \leq A, B, C \leq 10^8$ ,  $0 \leq N \leq \min\{1,000, A \cdot B \cdot C - 1\}$ ).

Каждая из последующих  $N$  строк содержит целые неотрицательные числа  $x$ ,  $y$  и  $z$ , которые задают координаты очередного удаляемого кубика. Гарантируется, что кубик с такими координатами существует и ещё не был удалён.

### Output

Выведите площадь поверхности получившегося после удаления  $N$  кубиков объекта.

### Examples

standard input	standard output
2 2 2 1 0 0 0	24
1 1 5 2 0 0 1 0 0 3	18
3 3 3 1 1 1 1	60

### Алгоритм

Данная задача на реализацию. Необходимо грамотно выстроить структуру и уметь находить изменение площади кубика после убиания кубика на заданных координатах. Сложность  $O(N^2)$ .

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4 #define ll long long
5 class Point {
6 public:
7     ll x, y, z;
8     Point() {}
9     Point(ll a, ll b, ll c) {
10         x = a;
11         y = b;
12         z = c;
13     }
14     void setPoints(ll a, ll b, ll c) {
15         x = a;
16         y = b;
17         z = c;
18     }
19     bool eql(ll a, ll b, ll c) {
20         return a == x && b == y && c == z;
21     }
22 };
23 int main() {
24     ll a, b, c, n;
25     cin >> a >> b >> c >> n;
26     vector<Point> points(n);
27     ll square = 2*(a*b + b*c + a*c);
28     for (int i = 0; i < n; ++i) {
29         ll x, y, z;
30         cin >> x >> y >> z;
31         points[i].setPoints(x, y, z);
32         ll s = 0;
33         if (x == 0) ++s;
34         if (x == a-1) ++s;
35         if (y == 0) ++s;
36         if (y == b-1) ++s;
37         if (z == 0) ++s;
38         if (z == c-1) ++s;
39         ll p = 0;
40         for (int j = 0; j < i; ++j) {
41             if (points[j].eql(x-1, y, z)) ++p;
42             if (points[j].eql(x+1, y, z)) ++p;
43             if (points[j].eql(x, y-1, z)) ++p;
44             if (points[j].eql(x, y+1, z)) ++p;
45             if (points[j].eql(x, y, z-1)) ++p;
46             if (points[j].eql(x, y, z+1)) ++p;
47         }
48         square += 6-2*(s+p);
49     }
50     cout << square << "\n";
51     return 0;
52 }
```

## Задача K - Emoticon Counter

Три последовательных символа “:-)” в текстовом сообщении обозначают радостный смайлик, а три последовательных символа “:-(|” — грустный.

Общее настроение текстового сообщения определим так: если в сообщении весёлых смайликов больше, чем грустных, то ответ будет “`happy`”, если грустных больше, чем весёлых, то ответ будет “`sad`”, иначе ответ — “`neutral`”.

Напишите программу, которая определяет общее настроение сообщения.

### Input

Одна строка, содержащая от 1 до 255 символов с кодами, не меньшими 32 и не превосходящими 127.

### Output

Выведите строку, соответствующую общему настроению сообщения.

### Examples

standard input	standard output
Solved the problem :-) got TL :-( then Accepted :-)	happy
Happy? :)	neutral
Emo:-ticons are go:-(:-(ing out of co:-)ntrol	sad

## Алгоритм

В этой задаче необходимо найти все вхождения радостных и грустных смайликов и вывести в ответе каких больше. Решается поиском подстроки в строке. Сложность  $O(N * M)$ .

## Исходный код

```
1 str = gets
2 str = str + "$"
3 happy = str.split(":-)")
4 sad = str.split(":-(|")
5 if happy.size > sad.size
6   puts "happy"
7 elsif happy.size < sad.size
8   puts "sad"
9 else
10  puts "neutral"
11 end
```

# Задача L - Rogue Language

Задан шифр, строящийся следующим образом:

- Гласные ('a','i','o' and 'u' остаются на месте).
- Согласные заменяются тремя буквами: исходной согласной, ближайшей к ней гласной (для 'b' ответом будет 'a'), если расстояния от двух гласных равны, то гласной с наименьшим номером (для 'c' ответом также будет 'a'), и следующей за исходной согласной; для 'z' снова берётся 'z'.

Зашифруйте заданное слово.

## Input

Входной файл состоит из одного непустого слова, составленного из не более, чем 30 строчных латинских букв.

## Output

Выведите зашифрованное слово.

## Examples

standard input	standard output
opencup	ороqепорcadupoq
xyz	xuyyuzzuz

## Алгоритм

В этой задаче нужно понять как меняются символы и просто сделать замену всех символов по составленному словарю замен. Сложность  $O(1)$ .

## Исходный код

```
1 #!/usr/bin/perl
2 my %h = (
3     a => "a", b => "bac", c => "cad", e => "e", f => "feg",
4     g => "geh", h => "hij", i => "i", j => "jik", k => "kil",
5     l => "lim", m => "mon", n => "nop", o => "o", p => "poq",
6     q => "qor", r => "ros", s => "sut", t => "tuv", u => "u",
7     v => "vuw", w => "wux", x => "xuy", y => "yuz", z => "zuz");
8 my $a = <>;
9 chomp $a;
10 my $res = "";
11 for my $i (split "", $a) {
12     $res .= $h{$i};
13 }
14 print $res, "\n";
```

## Задача M - RPS

Алиса и Боб играют в игру “камень-ножницы-бумага”. В каждой партии они одновременно выбирают одну из трёх фигур: камень (rock), ножницы (scissors) или бумага (paper). При этом камень выигрывает у ножниц, ножницы — у бумаги, а бумага — у камня. Если выбраны одинаковые фигуры, фиксируется ничья.

Вычислите, сколько партий выиграла Алиса, и сколько — Боб.

### Input

Первая строка входа содержит целое число  $N$  ( $1 \leq N \leq 100$ ) — количество сыгранных партий.

Вторая строка задаёт последовательность ходов Алисы и содержит  $N$  слов.  $i$ -е слово в последовательности задаёт ход Алисы в  $i$ -й партии и может быть одним из вариантов “rock” (для камня), “paper” (для бумаги) и “scissors” (для ножниц).

Третья строка задаёт последовательность ходов Боба в том же самом формате.

### Output

Выведите два целых числа — количество партий, выигранных Алисой, и количество партий, выигранных Бобом, соответственно.

### Examples

standard input	standard output
4 paper scissors rock paper rock rock scissors paper	2 1

### Алгоритм

В этой задаче упор на реализацию. Нужно просто пройти по входным строкам и сосчитать количество выигрышных партий у каждого из игроков. Сложность  $O(N)$ .

## Исходный код

```
1 #!/usr/bin/perl
2 my %h = (paper => 0, scissors => 1, rock => 2);
3 my $n = <>;
4 my $A = <>;
5 my $B = <>;
6 chomp $A;
7 chomp $B;
8 my @A = split " ", $A;
9 my @B = split " ", $B;
10 my ($ac, $bc) = (0, 0);
11 for (my $i = 0; $i < $n; ++$i) {
12     my ($a, $b) = (@A[$i], @B[$i]);
13     if ($a eq "paper" && $b eq "rock") {
14         ++$ac;
15     }
16     elsif ($b eq "paper" && $a eq "rock") {
17         ++$bc;
18     }
19     elsif ($h{$a} > $h{$b}) {
20         ++$ac;
21     }
22     elsif ($h{$a} < $h{$b}) {
23         ++$bc;
24     }
25 }
26 print "$ac $bc\n";
```

## 2.3 OpenCup GrandPrix of Eurasia Div 2

### Результаты

19.	MAI #6: Makarov, Rik, Yakimenko	-	+2 2:24	-	-	-	-5 4:59	-	-	+ 4:21	-	+1 0:43	+1 3:07	4	716	50%	0.51
-----	---------------------------------	---	------------	---	---	---	------------	---	---	-----------	---	------------	------------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10323](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10323)

## Задача 2 - Плэй-офф

Рассмотрим турнир, который проходит по олимпийской системе. У нас есть  $2^N$  участников. Они разбиваются на пары. В парах выявляются победители, которые выходят в следующий раунд. В этом раунде количество участников равно  $2^{N-1}$ . Они снова разбиваются на пары, в которых выявляется победитель, и т.д., до тех пор, пока не выявится абсолютный победитель.

Введём для команд отношение *сильнее*. Будем считать, что если в одном из раундов команда  $A$  обыграла команду  $B$ , то команда  $A$  *сильнее* команды  $B$ . Будем считать, что отношение *сильнее* является транзитивным: если команда  $A$  *сильнее* команды  $B$ , а команда  $B$  *сильнее* команды  $C$ , то и команда  $A$  будет *сильнее* команды  $C$ .

Например, если в полуфинале команда  $A$  обыграла  $B$ , а тем временем  $C$  обыграла  $D$ , затем в финале  $A$  обыграла  $C$ , то мы можем сказать, что  $A$  *сильнее*  $D$ , однако, сказать кто *сильнее* из команд  $B$  и  $C$  мы не можем.

Вам даны результаты турнира, прошедшего по олимпийской системе. Также дано несколько пар команд, про каждую из которых требуется сказать, является ли одна из команд в паре *сильнее* другой.

### Формат входного файла

В первой строке входного файла записано целое число  $N$  ( $1 \leq N \leq 18$ ).

Следующие  $2^N$  строк содержат названия команд-участниц. Названия команд уникальны, состоят из не более, чем 20 строчных букв латинского алфавита.

В первом раунде в первой паре играют первая команда со второй, во второй паре — третья с четвёртой и т. д. Во втором раунде в первой паре играет победитель первой пары первого раунда с победителем второй, во второй паре — победитель третьей с победителем четвёртой и т.д. Аналогичным образом составляются пары и для следующих раундов. Нумерация всех игр турнира осуществляется в соответствии с нумерацией пар в раундах: номера от 1 до  $2^{N-1}$  по порядку присваиваются парам первого раунда, номера от  $2^{N-1} + 1$  до  $2^{N-1} + 2^{N-2}$  по порядку присваиваются парам второго раунда и т. д.

Следующая строка состоит из  $2^N - 1$  символов 'W' или 'L' и содержит результаты игр в описанном порядке, где символ 'W' означает победу первой команды из пары, а символ 'L' — второй.

В следующей строке записано целое число  $Q$  — количество запросов ( $1 \leq Q \leq 10^5$ ).

Следующие  $Q$  строк содержат описания запросов. Каждый запрос состоит из двух различных слов, разделённых пробелом, — названий команд.

Гарантируется, что размер входного файла не превосходит трёх мегабайт.

### Формат выходного файла

В выходной файл на каждый запрос необходимо вывести одно из слов:

- Win, если первая команда *сильнее* второй;
- Lose, если вторая команда *сильнее* первой;
- Unknown, если про данные две команды нельзя сказать, что одна из них *сильнее* другой.

### Пример

input.txt	output.txt
3 cska jokerit ska dynamo avangard akbars sibir metallurg WWLWLWW 3 jokerit avangard ska metallurg metallurg akbars	Unknown Win Lose

### Алгоритм

Задача на реализацию. Мы строим дерево команд и каждой команде ставим в соответствие массив команд, которые слабее рассматриваемой. Затем мы считываем запросы и выводим ответ. Сложность алгоритма  $O(Q * N * \log(N))$ .

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <map>
4 #include <algorithm>
5 #include <fstream>
6 using namespace std;
7 typedef struct {
8     string name;
9     vector<string> losers;
10 } Team;
11 int main(int argc, const char * argv[]) {
12     ifstream in("input.txt");
13     ofstream out("output.txt");
14     int n;
15     in >> n;
16     int t = 1;
17     for (int i = 0; i < n; i++) {
18         t *= 2;
19     }
20     map<int, vector<int>> teams; // first=team_name, second=losers_vector
21     map<string, int> index_for_name;
22     vector<pair<int, int>> pairs;
23     vector<pair<int, int>> prev_pairs;
24     vector<char> log(t - 1);
25     for (int i = 0; i < t; i = i + 2) {
26         string team_name_1, team_name_2;
27         in >> team_name_1 >> team_name_2;
28         index_for_name[team_name_1] = i;
29         index_for_name[team_name_2] = i + 1;
30         teams[i] = vector<int>();
31         teams[i + 1] = vector<int>();
32         pairs.push_back(make_pair(i, i + 1));
33     }
34     in.get();
35     for (int i = 0; i < t - 1; i++) {
36         char result = in.get();
37         log[i] = result;
38     }
39     in.get();
40     prev_pairs = pairs;
41     pairs.clear();
42     bool end = false;
43     while (!end) {
44         for (int i = 0; i < t / 2; i += 2) {
45             int first_winner, first_looser;
46             int second_winner, second_looser;
47             if (log[i] == 'W') {
48                 first_winner = prev_pairs[i].first;
49                 first_looser = prev_pairs[i].second;
50                 teams[first_winner].push_back(first_looser);
51                 teams[first_winner].insert(teams[first_winner].end(), teams[
52                     first_looser].begin(), teams[first_looser].end());
53             } else {
54                 first_winner = prev_pairs[i].second;
55                 first_looser = prev_pairs[i].first;
56                 teams[first_winner].push_back(first_looser);
57                 teams[first_winner].insert(teams[first_winner].end(), teams[
58                     first_looser].begin(), teams[first_looser].end());
59             }
60         }
61     }
62 }
```

```

57
58     }
59     if (i == t - 2) {
60         end = true;
61         break;
62     }
63     if (log[i + 1] == 'W') {
64         second_winner = prev_pairs[i + 1].first;
65         second_looser = prev_pairs[i + 1].second;
66         teams[second_winner].push_back(second_looser);
67         teams[second_winner].insert(teams[second_winner].end(), teams[
68             second_looser].begin(), teams[second_looser].end());
69         } else {
70             second_winner = prev_pairs[i + 1].second;
71             second_looser = prev_pairs[i + 1].first;
72             teams[second_winner].push_back(second_looser);
73             teams[second_winner].insert(teams[second_winner].end(), teams[
74                 second_looser].begin(), teams[second_looser].end());
75         }
76         pairs.push_back(make_pair(first_winner, second_winner));
77     }
78     log.erase(log.begin(), log.begin() + t / 2);
79     prev_pairs = pairs;
80     pairs.clear();
81     t /= 2;
82 }
83 int q;
84 in >> q;
85 for (int i = 0; i < q; i++) {
86     string t1, t2;
87     in >> t1 >> t2;
88     int index_1 = index_for_name[t1];
89     int index_2 = index_for_name[t2];
90     if (find(teams[index_1].begin(), teams[index_1].end(), index_2) != teams[
91         index_1].end()) {
92         out << "Win" << endl;
93     } else if (find(teams[index_2].begin(), teams[index_2].end(), index_1) != teams[
94         index_2].end()) {
95         out << "Lose" << endl;
96     } else {
97         out << "Unknown" << endl;
98     }
99 }
100 in.close();
101 out.clear();
102 return 0;
103 }
```

## Задача 9 - Хэш-функция

Дана следующая хэш-функция:

```
uint32 super_hash_func(uint32 value) {  
    uint32 hash = value;  
    hash = hash + (hash << 10);  
    hash = hash xor (hash >> 6);  
    hash = hash + (hash << 3);  
    hash = hash xor (hash >> 11);  
    hash = hash + (hash << 16);  
    return hash;  
}
```

Где:

- `uint32` — 32-битный беззнаковый целочисленный тип данных.
- $a + b$  — сумма целых чисел  $a$  и  $b$ . Поскольку возможно переполнение, сумма вычисляется по модулю  $2^{32}$ .
- $a \text{ xor } b$  — побитовое “исключающее или” чисел  $a$  и  $b$ .  $k$ -ый бит результата равен 1 тогда и только тогда, когда  $k$ -ые биты у чисел  $a$  и  $b$  различны.
- $a \ll b$  — операция сдвига битов числа  $a$  на  $b$  разрядов влево. При этом младшие  $b$  разрядов числа заполняются нулями, а старшие  $b$  битов отбрасываются.
- $a \gg b$  — операция сдвига битов числа  $a$  на  $b$  разрядов вправо. При этом старшие  $b$  разрядов числа заполняются нулями, а младшие  $b$  битов отбрасываются.

Необходимо научиться обращаться данную хэш-функцию.

### Формат входного файла

В первой строке входного файла дано целое число запросов  $Q$  ( $1 \leq Q \leq 100$ ).

Далее в следующих  $Q$  строках записано по одному беззнаковому 32-битному целому числу.

### Формат выходного файла

В выходной файл на каждый запрос необходимо вывести 32-битное число, значение хэш-функции от которого будет равно числу, данному в запросе. Гарантируется, что для каждого запроса такое число существует.

### Пример

input.txt	output.txt
4	1
614278301	2
1228622139	3
1841720774	4
2457244278	

### Алгоритм

В каждой операции можно заметить некоторую закономерность, по которой можно получить обратную операцию с использованием некоторого перебора. Обратив таким образом все операции, можно получить искомое число.

## Исходный код

```
1 #include <iostream>
2 #include <cinttypes>
3 using namespace std;
4 #define ll long long
5 #define ull unsigned long long
6
7 uint32_t dehash(uint32_t value) {
8     uint32_t hash = value;
9     hash = hash - (hash<<16);
10    uint32_t t = hash;
11    hash = (hash xor (hash >> 11)) & ~((1ULL<<10)-1);
12    for (int i = 0; i < 1023; ++i) {
13        if ((hash xor (hash >> 11)) == t) {
14            break;
15        }
16        ++hash;
17    }
18    ull h = hash;
19    for (ull i = 0;; ++i) {
20        if (((h|(i<<32)) % 9) == 0) {
21            hash = (h|(i<<32))/9;
22            break;
23        }
24    }
25    t = hash;
26    hash = (hash xor (hash >> 6)) & ~((1ULL<<20)-1);
27    for (int i = 0; i < 1048575; ++i) {
28        if ((hash xor (hash >> 6)) == t) {
29            break;
30        }
31        ++hash;
32    }
33    h = hash;
34    for (ull i = 0;; ++i) {
35        if (((h|(i<<32)) % 1025) == 0) {
36            hash = (h|(i<<32))/1025;
37            break;
38        }
39    }
40    return hash;
41 }
42
43 int main() {
44     uint32_t Q;
45     cin >> Q;
46     for (int i = 0; i < Q; ++i) {
47         ull d;
48         cin >> d;
49         cout << dehash(d) << "\n";
50     }
51     return 0;
52 }
```

## Задача 11 - Стулья мастера Гамбса

Отец Фёдор попал как-то раз на мебельную выставку, где гамбсовскую мебель всякую разную продают. С одной стороны, он знает от предводителя, что вся мебель у него дома была помечена, — «а то всякие разные там в гости ходят», — а с другой стороны, он понятия не имеет, как тот самый гарнитур выглядит. Поэтому пришел ему в голову следующий план: он покупает что-нибудь из тех гарнитуров, на которые у него хватит денег (ну хотя бы по одному предмету), и ищет там метку, а потом уже будет докупать все оставшиеся стулья из нужного комплекта на вновь выпрошенные у матушки деньги.

Естественно, он хочет проверить как можно больше различных гарнитуров, но денег у него осталось не так много, да и умом он после лазанья с колбасой по горам слегка тронулся. Поэтому с задачей этой он может и не справиться. Помогите ему: определите, сколько гарнитуров может проверить отец Фёдор, так чтобы из каждого гарнитура он купил хотя бы один предмет, и при этом на все купленные предметы ему хватило имеющихся денег.

### Формат входного файла

В первой строке входного файла записано два целых числа  $N$  и  $S$ , где  $N$  — количество продающихся предметов,  $S$  — имеющаяся сумма денег ( $1 \leq N \leq 10^5$ ,  $1 \leq S \leq 10^6$ ).

Далее следуют  $N$  строк, содержащих по два целых числа  $a$  и  $b$  — номер гарнитура, к которому относится данный предмет, и его стоимость соответственно ( $1 \leq a \leq N$ ,  $1 \leq b \leq 10^5$ ).

### Формат выходного файла

В выходной файл необходимо вывести одно целое число — максимально возможное количество гарнитуров, из которых можно купить предметы, суммарная стоимость которых не превосходит  $S$ .

### Пример

input.txt	output.txt
6 17	
2 5	
1 5	
2 6	
2 3	
4 400	
5 230	2

### Алгоритм

Для решения данной задачи мы отсортируем стулья по стоимости, а затем будем набирать стулья начиная с самых дешевых. Если находим стул, который не можем купить, то завершаем выполнение, так как дальше не будет стульев которые мы сможем купить. Сложность алгоритма  $O(N^2 * \log(N))$ .

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3 #include <fstream>
4 #include <vector>
5
6 using namespace std;
7
8 bool pairMinMax(const pair<int , int> &a, const pair<int , int> &b) {
9     return a.second < b.second;
10 }
11
12
13 int main() {
14
15     ifstream in("input.txt");
16     ofstream out("output.txt");
17
18
19     int N;
20     long long S;
21
22     in >> N >> S;
23
24     vector<pair<int , int> > element(N);
25     vector<bool> bought(N+1, false);
26
27     pair<int , int> temp;
28     for (int i = 0; i < N; i++) {
29         in >> temp.first >> temp.second;
30         element[i] = temp;
31     }
32
33     sort(element.begin() , element.end() , pairMinMax);
34
35     int count = 0;
36     for (int i = 0; i < N; i++) {
37         if (element[i].second > S) {
38             break;
39         }
40         if (element[i].second <= S && bought[element[i].first] == false) {
41             S -= element[i].second;
42             bought[element[i].first] = true;
43             count++;
44         }
45     }
46
47     out << count << endl;
48
49     in.close();
50     out.close();
51
52     return 0;
53 }
```

## Задача 12 - Эрудит

В настольной игре «Эрудит» игроки соревнуются в составлении слов из имеющихся у них фишек, на каждой из которых написана одна буква, на поле размером  $15 \times 15$ .

В начале игры каждый получает по 7 фишек с буквами из «банка» фишек. Игроки совершают ходы по очереди. В свой ход игрок составляет на поле несколько слов (возможно одно), при этом слова составляются последовательно одно за другим. Составляя очередное слово, игрок помещает некоторое ненулевое количество своих фишек с буквами на игровое поле, а также указывает на некоторые фишечки, из числа тех, которые уже находятся на поле. Число фишек, на которые указывает игрок, составляя слово, не может быть нулевым, если на поле уже есть фишечки, то есть, если это не первое слово, которое составлено за всю игру. Назовём фишками, образующими слово, все фишечки, которые игрок поместил на поле, и те, на которые он указал. Фишечки, образующие слово, должны находиться подряд либо на одной вертикали, либо на одной горизонтали. Если эти фишечки находятся на одной горизонтали, то составленное слово должно получаться, если читать буквы, написанные на них, слева направо, а если на вертикали — то если читать сверху вниз. Самое первое слово, которое будет составлено на поле, должно одной из своих фишек занимать центральную клетку поля. После того, как игрок сделал свой ход, он получает столько фишек с буквами, сколько ему не достаёт до семи. Если в «банке» недостаточно фишек, чтобы он мог добрать до семи, то он получает все оставшиеся.

Количество очков, которые игрок получает за свой ход, рассчитывается как сумма очков за все составленные в рамках этого хода слова. Количество очков за слово рассчитывается, исходя из стоимости букв слова и цветов клеток, на которых находятся фишечки, образующие слово (раскраска игрового поля приведена в таблице 1). Каждая буква имеет некоторую базовую цену, выраженную в очках (базовые цены приведены в таблице 3). Цвет клетки определяет коэффициент буквы и коэффициент слова (конкретные коэффициенты для цветов приведены в таблице 2). Количество очков за букву на фишке, поставленной на клетку поля, равно произведению базовой цены буквы и коэффициента буквы для цвета данной клетки. Количество очков за слово равно сумме очков по всем фишкам, образующим данное слово, умноженной на произведение коэффициентов слова по всем клеткам, занимаемым фишечками, образующими это слово. Если в свой ход игрок поместил на поле все свои 7 фишек, он дополнительно получает бонус в размере 15 очков. Если у игрока меньше 7 фишек, то бонус он получить не может.

Количество очков, которые игрок получает по итогам игры, равно сумме очков по всем его ходам. Данная последовательность ходов игроков, требуется посчитать набранное ими количество очков.

Все табличные данные, приведённые в данном условии, вы можете найти в электронном виде (вместе с примерами по кнопке Download Samples Zip из интерфейса ejudge).

### Формат входного файла

В первой строке входного файла заданы два целых числа  $N$  и  $M$  — количество игроков и количество ходов ( $2 \leq N \leq 4$ ,  $1 \leq M \leq 130$ ).

Затем следуют  $M$  описаний ходов. Каждое описание начинается с количества слов  $W$ , составленных в рамках этого хода ( $1 \leq W \leq 7$ ).

На следующих  $W$  строках описаны составленные слова. Описание слова начинается с длины слова  $L$ , за которым через пробел располагается символ, показывающий его направление ('**h**' — для слов, составленных слева направо, и '**v**' — для слов, составленных сверху вниз), а затем записываются координаты первой буквы слова  $X, Y$  ( $2 \leq L \leq 15, 1 \leq X, Y \leq 15$ ).  $X$  является номером столбца, а  $Y$  является номером строки. Столбцы нумеруются слева направо, строки — сверху вниз. Далее следуют  $L$  чисел, кодирующих буквы составленного слова.

## Формат выходного файла

В выходной файл необходимо вывести  $N$  чисел, по одному на строке, — очки игроков по итогам  $M$  совершенных ходов.

## Пример

input.txt	output.txt
2 6	121
2	102
3 h 7 8 13 6 24	
5 v 9 6 17 28 24 1 4	
1	
5 h 5 9 3 6 18 14 1	
2	
4 h 6 10 11 17 20 4	
4 h 7 6 11 15 17 5	
3	
5 v 6 8 18 6 11 1 24	
4 v 7 3 32 26 9 11	
3 h 8 9 14 1 10	
3	
5 h 4 12 16 15 24 11 1	
4 v 8 10 20 12 1 14	
2 h 7 4 26 9	
4	
3 h 7 4 26 9 19	
4 v 7 9 18 17 15 11	
4 h 2 12 5 6 16 15	
4 h 7 12 11 1 16 1	

## Алгоритм

Эта задача на реализацию. Требуется просто запрограммировать все действия, описанные в условии.

## Исходный код

```
1 #!/usr/bin/perl
2 use bigint;
3 my @cmap = qw( rwwgwwwrwwwgwww
4                 wbwwwywwwywwwbw
5                 wwbwwwwgwwwbw
6                 gwwbwwwgwwwbw
7                 wwwbwwwbw
8                 wywwwwwwwwwwyw
9                 wwgwwwwgwwwgww
10                rwwgwwwswwwgwww
11                wwgwwwwgwwwgww
12                wywwwwwwwwwwyw
13                wwwbwwwbw
14                gwwbwwwgwwwbw
15                wwbwwwwgwwwbw
16                wbwwwywwwywwwbw
17                rwwgwwwrwwwgwww ) ;
18 my @map;
19 for (my $i = 0; $i < 16; ++$i) {
20   for (my $j = 0; $j < 16; ++$j) {
21     $map[$i][$j] = 0;
22   }
23 }
24 my @alp = (1,3,2,3,2,1,5,5,1,2,2,2,2,1,1,2,2,2,2,3,10,5,10,5,10,10,10,5,5,10,10,3);
25
26 my %c1 =
27   (w => 1,
28    s => 1,
29    g => 2,
30    y => 3,
31    b => 1,
32    r => 1);
33
34 my %cw =
35   (w => 1,
36    s => 1,
37    g => 1,
38    y => 1,
39    b => 2,
40    r => 3);
41
42 sub getColor {
43   my ($x, $y) = @_;
44   substr $cmap[$y-1], $x-1, 1;
45 }
46 my $str;
47 $str = <ARGV>;
48 chomp $str;
49 my ($n, $m) = split(" ", $str);
50 my @scores;
51 for (my $i = 0; $i < $n; ++$i) {
52   $scores[$i] = 0;
53 }
54 for (my $si = 0; $si < $m; ++$si) {
55   my $str = <ARGV>;
56   chomp $str;
57   my $w = $str;
58   my $count = 0;
```

```

59 my $score = 0;
60 for (my $i = 0; $i < $w; ++$i) {
61     $str = <ARGV>;
62     chomp $str;
63     my @A = split(" ", $str);
64     my $l = shift @A;
65     my $dir = shift @A;
66     my $x = shift @A;
67     my $y = shift @A;
68     my $scoreL = 0;
69     my $scoreW = 1;
70     for my $char (@A) {
71         my $color = getColor($x, $y);
72         unless ($map[$x][$y]) {
73             $map[$x][$y] = 1;
74             ++$count;
75         }
76         $scoreL += $cl{$color} * $alp[$char - 1];
77         $scoreW *= $cw{$color};
78         if ($dir eq 'v') {
79             ++$y;
80         }
81         else {
82             ++$x;
83         }
84     }
85     $score += $scoreW * $scoreL;
86 }
87 if ($count == 7) {
88     $score += 15;
89 }
90 $scores[$si % $n] += $score;
91 }
92 print join("\n", @scores), "\n";

```

## 2.4 OpenCup GrandPrix of SPb Div 2

### Результаты

70.	MAI #6: Makarov, Rik, Yakimenko	-	-	-	-	+2 4:41	-	+ 2:20	-8 5:03	-2 4:17	-8 3:01	-	-	2	461	50%	0.38
-----	---------------------------------	---	---	---	---	------------	---	-----------	------------	------------	------------	---	---	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10324](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10324)

## Задача Е - Следующее разбиение

Рассмотрим все разбиения целого положительного числа  $N$  на  $K$  целых положительных слагаемых. Запишем каждое разбиение как последовательность чисел от больших слагаемых к меньшим. Отсортируем разбиения в обратном лексикографическом порядке. Найдите следующее разбиение  $N$  на  $K$  слагаемых в заданном порядке или определите, что его не существует.

### Формат входных данных

В первой строке дано целое число  $K$  ( $K \leq 10^5$ ,  $K \leq N$ ,  $1 \leq N \leq 10^9$ ). Во второй строке даны  $K$  целых чисел  $A_i$  — слагаемые разбиения ( $1 \leq A_i \leq N$ ).

### Формат выходных данных

Если следующее разбиение не существует, выведите  $-1$ . Иначе в первой строке выведите  $K$ . Во второй строке выведите  $K$  чисел — слагаемые следующего в обратном лексикографическом порядке разбиения.

### Примеры

next-partition.in	next-partition.out
6 4 2 2 2 1 1	6 3 3 3 1 1 1
3 2 2 2	-1

## Алгоритм

Задача решается с помощью перебора с некоторыми оптимизациями. Сложность  $O(n^3)$ .

## Исходный код

```
1 #include <iostream>
2 using namespace std;
3 bool pairMinMax(const pair<int, int> &a, const pair<int, int> &b) {
4     return a.second < b.second;
5 }
6 int main() {
7     int input;
8     ifstream in("next-partition.in");
9     ofstream out("next-partition.out");
10    int K;
11    in >> K;
12    vector<int> A(K);
13    int N = 0;
14    int temp;
15    for (int i = 0; i < K; i++) {
16        in >> temp;
17        A[i] = temp;
18        N += temp;
19    }
20    bool flag1 = false;
21    bool flag2 = false;
22    int sum = 0;
23    sort(A.begin(), A.end());
24    int iMarked = 0;
25    for (int i = 0; i < K - 1; i++) {
26        sum += A[i];
27        if (A[i + 1] - A[i] >= 2) {
28            iMarked = i + 1;
29            A[i + 1]--;
```

```

30     sum++;
31     flag1 = true;
32     break;
33   }
34 }
35 if (!flag1) {
36   for (int i = 0; i < K - 2; i++) {
37     for (int j = i + 2; j < K; j++) {
38       if (A[i] + 1 > A[i + 1]) {
39         break;
40       }
41       if (A[j] - A[i] > 2) {
42         break;
43       }
44       if (A[j] - A[i] == 2 && A[j] - 1 >= A[j - 1]) {
45         A[j]--;
46         A[i]++;
47         flag2 = true;
48         break;
49       }
50     }
51   }
52 }
53 if (flag1) {
54   sum -= iMarked;
55   for (int i = iMarked - 1; i >= 0; i--) {
56     if (sum > 0) {
57       temp = A[iMarked];
58       if (sum+1 >= temp) {
59         A[i] = temp;
60         sum = sum - temp + 1;
61       }
62     } else {
63       A[i] = sum + 1;
64       sum = 0;
65     }
66   }
67   else A[i] = 1;
68 }
69 out << K << endl;
70 for (int i = K - 1; i >= 0; i--) {
71   out << A[i] << " ";
72 }
73 }
74 else if (flag2) {
75   out << K << endl;
76   for (int i = K - 1; i >= 0; i--) {
77     out << A[i] << " ";
78   }
79 } else {
80   out << -1;
81 }
82 in.close();
83 out.close();
84 return 0;
85 }

```

## Задача G - Головоломка Обмен

Головоломка «Обмен» выглядит как доска  $4 \times 4$ , в клетках которой расставлены числа от 1 до 16, каждое по одному разу. Каждое число записывается ровно двумя десятичными цифрами: в числах 1–9 добавлены ведущие нули.

За одну операцию с головоломкой можно взять любые две клетки таблицы и поменять их местами.

Найдите любую кратчайшую последовательность операций, выполнив которую, можно получить упорядоченную таблицу:

01	02	03	04
05	06	07	08
09	10	11	12
13	14	15	16

### Формат входных данных

Входные данные состоят из четырёх строк, соответствующих строкам таблицы. В каждой строке записано четыре числа, каждое из которых состоит ровно из двух десятичных цифр — для этого в числах 1–9 добавлены **ведущие нули**. Соседние числа в строке разделены пробелом. Гарантируется, что каждое из чисел от 1 до 16 встречается в заданной таблице ровно один раз.

### Формат выходных данных

В первой строке выведите одно число — количество операций  $k$ . Это число должно быть минимально возможным. В следующих  $k$  строках выведите сами операции.

Для записи операций обозначим строки буквами  $a$ ,  $b$ ,  $c$ ,  $d$  сверху вниз, а столбцы цифрами 1, 2, 3, 4 слева направо. Операция обмена двух клеток записывается как  $r_1c_1-r_2c_2$ : строка и столбец одной клетки, знак «-» (минус, ASCII-код 45), строка и столбец другой клетки.

Если существует несколько возможных ответов, выведите любой из них.

### Пример

puzzle-swap.in	puzzle-swap.out
02 01 03 04	4
05 10 13 08	b2-c2
09 12 11 06	a1-a2
07 14 15 16	c4-b2 b3-d1

### Алгоритм

Так как доска всегда имеет размеры  $4 \times 4$ , то задачу можно решить полным перебором. Сначала перебором находим все неправильные пары чисел, а затем начинаем исправлять пары переставляя строки, пока доска не станет правильной.

## Исходный код

```
1 #include <iostream>
2 #include <sstream>
3 #include <vector>
4 #include <fstream>
5
6 using namespace std;
7
8 bool desk_is_correct(int desk[4][4])
9 {
10     for (int i = 0; i < 4; i++) {
11         for (int j = 0; j < 4; j++) {
12             int targer_i = (desk[i][j] - 1) / 4;
13             int targer_j = (desk[i][j] - 1) % 4;
14             if (i != targer_i || j != targer_j) {
15                 return false;
16             }
17         }
18     }
19     return true;
20 }
21
22 char character_for_num(int num)
23 {
24     return char('a' + num);
25 }
26
27 int main(int argc, const char * argv[])
28 {
29     stringstream ss;
30
31     ifstream in("puzzle-swap.in");
32     ofstream out("puzzle-swap.out");
33
34     int desk[4][4] = {0};
35
36     for (int i = 0; i < 4; i++) {
37         for (int j = 0; j < 4; j++) {
38             in >> desk[i][j];
39         }
40     }
41
42     int moves = 0;
43     vector<string> log;
44
45     // find all wrong pairs
46     for (int i1 = 0; i1 < 4; i1++) {
47         for (int j1 = 0; j1 < 4; j1++) {
48             for (int i2 = 0; i2 < 4; i2++) {
49                 for (int j2 = 0; j2 < 4; j2++) {
50                     if (i1 != i2 || j1 != j2) {
51                         int targer_i1 = (desk[i1][j1] - 1) / 4;
52                         int targer_j1 = (desk[i1][j1] - 1) % 4;
53                         int targer_i2 = (desk[i2][j2] - 1) / 4;
54                         int targer_j2 = (desk[i2][j2] - 1) % 4;
55                         if (targer_i1 == i2 && targer_j1 == j2 &&
56                             targer_i2 == i1 && targer_j2 == j1) {
57                             moves++;
58                             swap(desk[i1][j1], desk[i2][j2]);
59                         }
60                     }
61                 }
62             }
63         }
64     }
65
66     cout << moves;
67 }
```

```

59                         ss << character_for_num(i1) << j1 + 1 << "—" <<
60     character_for_num(i2) << j2 + 1 << endl;
61             string s;
62             ss >> s;
63             log.push_back(s);
64         }
65     }
66 }
67 }
68 }
69
70 while (!desk_is_correct(desk)) {
71     for (int i = 0; i < 4; i++) {
72         for (int j = 0; j < 4; j++) {
73             int targer_i = (desk[i][j] - 1) / 4;
74             int targer_j = (desk[i][j] - 1) % 4;
75             if (i != targer_i || j != targer_j) {
76                 moves++;
77                 swap(desk[i][j], desk[targer_i][targer_j]);
78                 ss << character_for_num(i) << j + 1 << "—" << character_for_num(
79                     targer_i) << targer_j + 1 << endl;
80                     string s;
81                     ss >> s;
82                     log.push_back(s);
83                 }
84             }
85         }
86     }
87     out << moves << endl;
88     for (size_t i = 0; i < log.size(); i++) {
89         out << log[i] << endl;
90     }
91
92     in.close();
93     out.close();
94
95     return 0;
96 }
```

## **2.5 ACM-ICPC Московский четвертьфинал**

Так как соревнование проводилось в ВШЭ, то турнирная таблица с результатами и исходные коды программ не доступны.

## 2.6 OpenCup GrandPrix of Yekaterinburg Div 2

### Результаты

50.	MAI #6: Makarov, Rik, Yakimenko	-	+3 1:20	-	+1 1:52	-	-5 4:55	+	0:09	-	-	-15 4:58	+1 1:29	4	392	55%	0.07
-----	---------------------------------	---	------------	---	------------	---	------------	---	------	---	---	-------------	------------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10325](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10325)

## Задача В - Black Widow

Наташа Романов, она же Чёрная Вдова, — легендарный агент Щ.И.Т. и непревзойдённый шпион. Она знает с десяток языков, владеет карате, айкидо, джиу-джитсу, самбо, сават, а также кунг-фу. Кроме того, Наташа обладает незаурядным умом и выдающимися актёрскими способностями, которые позволяют ей добывать информацию, даже когда её пытают.

Очередное задание Наташи — украсть описание нового секретного оружия Гидры. Ей не составило труда проникнуть в сейф через вентиляцию, однако выйти из него она может только через коридор, пол которого оснашён датчиками, реагирующими на давление. Конечно, в проекте коридора весь пол устлан датчиками. Но даже в Гидре воруют бюджетные средства! Поэтому на деле датчики представляют собой тонкие полосы от стены до стены, установленные в определённых точках коридора.

У Наташи, конечно же, есть план коридора, через который ей необходимо пройти. Поэтому ей известно расстояние до каждого из датчиков. Агент Романов хочет преодолеть этот коридор самым быстрым способом — цепочкой фляков. Однако, недостатком этого способа является тот факт, что расстояние между двумя последовательными касаниями пола (руками или ногами) на протяжении всего пути оказывается одинаковым (и выбирается спортсменом в момент первого прыжка).

Учитывая, что чем длиннее прыжок, тем больше сил на него тратится, вычислите, какое минимальное расстояние между двумя последовательными касаниями пола позволит Чёрной Вдове не попасться.

### Формат входных данных

В первой строке задано число  $N$  — количество датчиков на полу в коридоре ( $1 \leq N \leq 1000$  — кризис, сотрудникам тоже что-то есть надо, поэтому датчиков так мало). Во второй строке даны  $N$  чисел  $x_i$  — расстояния от точки, где первоначально стоит Наташа, до соответствующего датчика ( $1 \leq X \leq 10^9$  — коридор, ведущий к суперсекретному оружию, должен быть длинным даже в кризис).

### Формат выходных данных

В единственной строке выведите одно целое число — минимальное расстояние между двумя последовательными касаниями пола.

### Примеры

standard input	standard output
5 1 3 4 8 10	6
10 2 3 5 10 12 15 20 30 44 63	8

### Алгоритм

Перебираем всевозможные варианты перемещения и выберем тот, у которого расстояние между соседними точками будет минимальным.

## Исходный код

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     ull n;
5     cin >> n;
6     set<ull> v;
7     vector<ull> m;
8     ull maxV = 0;
9     ull a = 1;
10    for (int i = 0; i < n; ++i) {
11        ull d;
12        cin >> d;
13        if (d > maxV)
14            maxV = d;
15        v.insert(d);
16        m.push_back(d);
17    }
18    bool f = 1;
19    while (1) {
20        ++a;
21        bool f = 1;
22        for (int i = 0; i < n; ++i) {
23            if (m[i] >= a && m[i] % a == 0) {
24                f = 0;
25                break;
26            }
27        }
28        if (f) {
29            break;
30        }
31    }
32    while (1) {
33        f = 1;
34        for (ull x = 0; x < maxV+1; x += a) {
35            if (v.find(x) != v.end()) {
36                f = 0;
37                break;
38            }
39        }
40        if (f) {
41            cout << a << endl;
42            break;
43        }
44    }
45    while (1) {
46        ++a;
47        bool f = 1;
48        for (int i = 0; i < n; ++i) {
49            if (m[i] >= a && m[i] % a == 0) {
50                f = 0;
51                break;
52            }
53        }
54    }
55 }
56 return 0;
57 }
```

## Задача D - Dr. Banner

Доктор Беннер — выдающийся физик. Никто не знает о гамма-лучах так много, как он. К сожалению, у него бывают... ну эээ... вы знаете... “зелёные” проблемы. Что он только ни пробовал, чтобы справиться с ними! Выход оказался довольно простым. Чтобы погасить приступ нарастающего гнева, ему нужно сосредоточиться на чём-нибудь другом. Например, на построении пирамидок из детских кубиков.

Для построения очередной серии пирамидок Доктор Беннер сначала выбирает размер кубика в основании. Всё-таки он учёный и любит, чтобы все построенные им пирамидки имели одинаковые основания. После выбора основания процесс построения пирамидки предельно прост: нужно ставить очередной кубик на предыдущий. При этом, если предыдущий кубик имеет ребро размера  $K$ , то можно либо положить сверху кубик с ребром размера от 1 до  $K/2$ , либо остановиться и считать пирамидку построенной.

Беннер не любит повторяться, а значит, чтобы случайно не разозлиться, ему необходимо строить различные пирамидки.

Так как один из друзей Доктора Беннера — миллиардер Тони, в распоряжении Доктора есть неограниченное количество кубиков с ребром любой целой положительной длины.

Друзья очень любят Доктора Беннера, особенно, когда он незеленый. Они хотят понять, как долго можно находиться в безопасности рядом с Доктором при условии, что он справляется со строительством одной пирамидки за одну секунду.

### Формат входных данных

На вход подается одно число  $N$  — размер основания пирамидки, выбранный доктором. ( $1 \leq N \leq 10^5$ )

### Формат выходных данных

На выход программа должна выдать одно число — количество секунд, которое потребуется Доктору на построение всех возможных пирамидок. Ответ необходимо вывести по модулю числа  $10^9 + 7$ .

### Примеры

standard input	standard output
1	1
2	2

### Алгоритм

Для решения этой задачи нам нужно знать префиксную сумму по времени на отрезке. Мы можем вычислить вручную несколько первых значений, по которым потом сможешь получить любое значение, так как все они высчитываются на основе предыдущих. Сложность алгоритма  $O(N)$ .

## Исходный код

```
1 n = gets.to_i
2 m = (1e+9 + 7).to_i
3
4 amount = Array.new(1e+5 + 1)
5 sum_for_pos = Array.new(1e+5 + 1)
6
7 k = 5
8
9 amount[1] = 1
10 amount[2] = 2
11 amount[3] = 2
12 amount[4] = 4
13
14 sum_for_pos[1] = 1
15 sum_for_pos[2] = 3
16 sum_for_pos[3] = 5
17 sum_for_pos[4] = 9
18
19 while k <= n do
20     max_next_k = k / 2
21     amount[k] = (1 + sum_for_pos[max_next_k])
22     sum_for_pos[k] = sum_for_pos[k - 1] + amount[k]
23     k += 1
24 end
25
26 answer = (amount[n] % m)
27
28 puts answer
```

## Задача G - Groot

Я есть Грут.

### Формат входных данных

Я есть Грут.

### Формат выходных данных

Я есть Грут.

### Примеры

standard input	standard output
I am Groot	Pfff
I am Groot!	Wow
I am Groot!!!!!!	Woooooow

### Алгоритм

В этой задаче нужно посчитать количество восклицательных знаков во входной строке и вывести слово Wow, в котором столько же букв o, сколько знаков в строке. Если в строке нет ни одного знака, то вывести Pffff. Сложность  $O(1)$ .

### Исходный код

```
1 #!/usr/bin/perl
2 my $str = <>;
3 my @a = $str =~ /\!-/g;
4 my $n = scalar(@a);
5 if ($n == 0) {
6     print "Pfff";
7 }
8 else {
9     print "W". "o" x $n . "w";
10 }
```

## Задача L - Loki and Forks

Локи подарили на день рождения 5 наборов вилок. Каждый набор представляет из себя коробочку, внутри которой находится ровно 5 вилок.

Перед тем как выставить наборы в шкаф, Локи хочет узнать точное количество различных наборов вилок, чтобы освободить для них место. Каждую вилку можно охарактеризовать одним целым числом  $k$  — количеством зубьев. Вилки считаются одинаковыми, если они имеют равное количество зубьев. Два набора  $a$  и  $b$  считаются одинаковыми, если для любого  $k$  количество вилок с  $k$  зубьями в наборе  $a$  совпадает с количеством вилок с  $k$  зубьями в наборе  $b$ .

Помогите Локи выяснить количество различных наборов.

### Формат входных данных

В каждой из 5 строк находятся 5 чисел — количества зубьев. Каждая строка представляет отдельный набор вилок. Все числа во входных данных целые, положительные и не превосходят 100.

### Формат выходных данных

Выведите одно целое число — количество различных наборов.

### Примеры

standard input	standard output
1 1 7 1 1 1 7 1 7 1 7 1 1 1 7 7 7 7 7 7 7 1 1 1 7	3
1 1 2 2 1 1 2 3 1 1 2 3 1 1 1 1 2 3 1 1 1 1 2 2 1	2

### Алгоритм

Чтобы посчитать количество уникальных наборов нужно просто отсортировать все значения количества зубьев в наборах, затем построить из этих чисел строки путем конкатенирования и посчитать количество уникальных строк. Сложность  $O(N * \log(N))$ .

### Исходный код

```
1 #!/usr/bin/perl
2 my %h;
3 while (my $str = <>) {
4     chomp $str;
5     ++$h{join "", sort {$a <=> $b} split " ", $str};
6 }
7 print scalar(keys(%h));
```

## 2.7 OpenCup GrandPrix of Siberia Div 2

### Результаты

37.	MAI #6: Makarov, Rik, Yakimenko	+	+	-	+	-25	-	+	-	-	+	+2	6	467	25%	0.15
-----	---------------------------------	---	---	---	---	-----	---	---	---	---	---	----	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10326](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10326)

## Задача А - Detect a Mood

Lets call *regular bracket sequence* a sequence from the opening ('(') and closing (')') brackets, such as

- empty sequence is regular bracket sequence;
- if  $a$  is regular bracket sequence, then  $(a)$  is regular bracket sequence;
- if  $a$  and  $b$  are regular bracket sequences, their concatenation is regular bracket sequence.

Giga found a regular bracket sequence and decided to add some mood in it. For this purpose he added some additional brackets, representing happy (')) and sad ('(') smileys. Note that Giga's brackets may not form the regular bracket sequence.

Given the resulting sequence, find out overall mood of the new sequence — difference between number of happy and sad smileys, added by Giga.

### Input

Input consists of one line, containing non-empty string consisting of '(' and ')' brackets. String is not longer than 300 characters.

### Output

Print one integer: difference between number of happy and sad smileys, added by Giga.

### Example

standard input	standard output
((())()	-1
)()	0

### Алгоритм

В этой задаче нужно посчитать количество открывающих и закрывающих скобок, а затем найти их разность. Это и будет ответом. Сложность  $O(N)$ .

### Исходный код

```
1 #!/usr/bin/perl
2 my $s = <>;
3 chomp $s;
4 while ($s =~ s/\(\)//) {}
5 my $n = length($s);
6 $s =~ s/\)///g;
7 my $rn = length($s);
8 print $n - 2*$rn;
```

## Задача В - Painting Tracks

Identical uncolored tiles  $2 \times 1$  laid out as the two tracks in next way. Upper row is composed of  $N$  tiles with upper left corners  $(2, 1), (4, 1), \dots, (2N, 1)$ , while second is composed of  $M$  tiles with upper left corners  $(3, 0), (5, 0), \dots, (2M + 1, 0)$ .



You are given 3 different colors and must paint the tracks in such a way that each tile must be completely painted in one of those colors and two adjacent tiles need to be painted in a different color.

Determine the number of different possible paintings.

### Input

Input contains two integers  $N$  and  $M$  ( $1 \leq N, M \leq 10^{17}$ ,  $|N - M| \leq 55$ ).

### Output

Print one integer — number of different paintings.

### Examples

standard input	standard output
2 2	6
3 1	12

### Алгоритм

Если  $N = M$ , то ответ 6. Иначе ответ равен 3 умноженной на 2 в степени разности между  $N$  и  $M$ . Сложность  $O(1)$ .

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <cmath>
5 #include <set>
6
7 #define LL long long
8 #define ULL unsigned long long
9
10
11 using namespace std;
12
13 ULL myAbs(ULL a, ULL b) {
14     if(a > b) {
15         return a - b;
16     } else {
17         return b - a + 1;
18     }
19 }
20
21 int main() {
22
23     int temp;
24     ULL N, M;
25     cin >> N >> M;
26     if(N == M) {
27         cout << 6 << endl;
28         return 0;
29     }
30     else {
31         ULL answ = 3;
32         for(int i = 0; i < myAbs(N, M); i++) {
33             answ *= 2;
34         }
35         cout << answ << endl;
36     }
37
38     return 0;
39 }
40 }
```

## Задача D - Match of the Giants

As you know, some universities have a wonderful tradition to hold the so-called “Battle of the Giants”: coach of each university selects  $N$  teams from each university, and then summary results of selected teams for those universities at same programming contest are compared.

Very reputable universities GTU and TSU use their own way to calculate results of match: each coach selects ordered list of teams and then teams play one vs. one: if  $i$ -team in list of GTU is placed higher in standings than  $i$ -th team in list of TSU, then GTU scores for  $i$ -th team, else TSU scores (consider that places cannot be shared, scoring team adds 1 to score, other team's score not changed). So final score for each university is sum of scores for all  $N$  pairs. In this way order of teams in list is very important, so why coaches keep their lists in secret till the contest starts.

Today the Battle between GTU ad TSU was postponed due to the coincidence in time with an important matsh of Georgian rugby team. So bookmakers decided to use this pause to calculate odds for the upcoming Battle.

Bookmakers somehow got unordered list of teams from each side with information about team's strength; it is guaranteed that more strong team will always score against weaker one. It happened that no two teams from different universities have same strength.

But bookmakers does not know ordering, which was selected by coaches, so they want you to calculate maximal possible score for each university,

### Input

The first line of the input consists of one integer  $N$  — number of teams from the each university ( $1 \leq N \leq 2 \cdot 10^5$ ). Second line contains  $N$  integers  $G_i$  ( $1 \leq G_i \leq 10^5$ ) — strengths of GTU teams. Third line contains  $N$  integers  $T_i$  ( $1 \leq T_i \leq 10^5$ ) — strengths of TSU teams. It is guaranteed that  $T_i \neq G_j$  for any  $1 \leq i, j \leq N$ .

### Output

Print two integers — maximal possible score for GTU and maximal possible score for TSU.

### Examples

standard input	standard output
4 7 1 5 3 2 4 6 4	3 3
1 2 3	0 1
5 2 3 3 2 3 4 1 1 1 4	3 2

### Алгоритм

В этой задаче нужно найти максимально возможный счет для каждой команды. Для этого отсортируем значения каждой команды по убыванию и для каждой команды найдем значение в противоположной команде, которое даст максимальный выигрыш. Сложность  $O(N * \log(N))$ .

## Исходный код

```
1 #include <iostream>
2 using namespace std;
3 int main(int argc, const char * argv[]) {
4     int n;
5     cin >> n;
6     vector<int> gtu(n);
7     vector<int> tsu(n);
8     int val;
9     for (int i = 0; i < n; i++) {
10         cin >> val;
11         gtu[i] = val;
12     }
13     for (int i = 0; i < n; i++) {
14         cin >> val;
15         tsu[i] = val;
16     }
17     sort(gtu.begin(), gtu.end(), greater<int>());
18     sort(tsu.begin(), tsu.end(), greater<int>());
19     int gtu_score = 0;
20     int tsu_score = 0;
21     vector<int> tsu_search = tsu;
22     int next_start_pos = 0;
23     for (int i = 0; i < n; i++) {
24         bool found = false;
25         for (int j = next_start_pos; j < n; j++) {
26             if (gtu[i] > tsu_search[j]) {
27                 gtu_score++;
28                 tsu_search[j] = INT_MAX;
29                 found = true;
30                 next_start_pos = j + 1;
31                 break;
32             }
33         }
34         if (!found) {
35             break;
36         }
37     }
38     next_start_pos = 0;
39     for (int i = 0; i < n; i++) {
40         bool found = false;
41         for (int j = next_start_pos; j < n; j++) {
42             if (tsu[i] > gtu[j]) {
43                 tsu_score++;
44                 gtu[j] = INT_MAX;
45                 found = true;
46                 next_start_pos = j + 1;
47                 break;
48             }
49         }
50         if (!found) {
51             break;
52         }
53     }
54     cout << gtu_score << " " << tsu_score << endl;
55     return 0;
56 }
```

## Задача G - Files list

You are given a set of files, and you are to output the number of files with each extension. The file extension is a sequence of characters in the name of the file after a dot character.

The file system is case-sensitive: even if the file names differ only by a characters case, they are still considered to be different.

### Input

In the first line of the input file there is an integer  $N$ , which is number of file names ( $1 \leq N \leq 10^3$ ). In each of the following  $N$  lines there is a file name that is no more than 200 characters in length. The file name consists of only lower and upper Latin letters, numbers and a dot character ‘.’. It is guaranteed that a dot character can be found in the file name exactly once. Also, there is at least one symbol before and after the dot. It is guaranteed that each file name is present only once in the input file.

### Output

For each of the extensions that are present in the input file, output the number of files with this extension in the form of <extension>:<number>. Output extensions in order of the first mention in the input file.

### Example

standard input	standard output
6 218052.pdf Movie00.mkv Invoice.xls book.pdf book.epub Movie01.mkv	pdf: 2 mkv: 2 xls: 1 epub: 1

### Алгоритм

В этой задаче нужно найти количество уникальных расширений файлов. Для этого каждое расширение в строке будем заносить в хеш-таблицу и запоминать количество вхождений. Сложность  $O(N * \log(N))$ .

### Исходный код

```
1 #!/usr/bin/perl
2 <>;
3 my %h;
4 my $i;
5 my @keys;
6 while (my $s = <>) {
7     chomp $s;
8     my ($suf) = $s =~ /\.(.+?)$/;
9     push @keys, $suf unless exists $h{$suf};
10    ++$h{$suf};
11 }
12 foreach my $key (@keys) {
13     print $key, ": ", $h{$key}, "\n";
14 }
```

## Задача K - Hive

There is a plane, which is tiled with regular hexagons. There are honeycombs on this plane. Worker bees at first misunderstood the project documentation, and now they have to turn beehive honeycombs around queen bee by 60 degree clockwise.

Beehive consists of hexagonal honeycombs, which are oriented in such manner that there are hexagon nodes below and above, and there are edges to the left and right which the honeycomb shares with its adjacent honeycombs in the row. Every consequent row is shifted relative to the previous row by half a honeycomb. The  $Ox$  axis goes from left to right along the horizontal row of honeycombs. The  $Oy$  axis is inclined 60 degrees relative to the  $Ox$  axis. The axes intersect at the honeycomb with coordinates  $(0, 0)$ . Example explanation contains illustrations showing the tiles numeration.

### Input

In the first line of the input file there are three integers  $N$ ,  $X$  and  $Y$ , where  $N$  is number of honeycombs in the hive (excluding the honeycomb of a queen bee),  $X$  and  $Y$  are coordinates of the honeycomb of a queen bee ( $1 \leq N \leq 10^4$ ;  $|X|, |Y| \leq 10^4$ ). In each of the following  $N$  lines there is a pair of integers  $x$  and  $y$ , being the coordinates of a honeycomb of the hive ( $|x|, |y| \leq 10^4$ ). The coordinates of all honeycombs in the input file are different.

### Output

For each of  $N$  honeycombs, you should output two integers on the separate line: its coordinates after the turn. The coordinates must be ordered as they are in the input file.

### Example

standard input	standard output
2 4 0	5 0
4 1	6 0
4 2	
1 0 0	2 -1
1 1	

### Алгоритм

Задача решается с помощью простой формулы, которая находится в исходном коде на 5 строке. Сложность  $O(1)$ .

### Исходный код

```
1 #!/usr/bin/perl
2 my ($n, $x0, $y0) = split " ", <>;
3 for (my $i = 0; $i < $n; ++$i) {
4     my ($x, $y) = split " ", <>;
5     print $x0+$y-$y0+$x-$x0, " ", $y0-$x+$x0, "\n";
6 }
```

## Задача L - Side effect

The programmer is developing an application. Being a part of a club “Young Friends of Functional Programming”, he wants to know how many of its functions have side effects. Initially, for each function of the program it is known, whether it has a side effect or it has not; also, it is known that no function calls any other. Yet the programmer decides to change the logic of the application and starts to put some calls of the functions to other functions. The programmer does not write new functions. If a function calls a function with side effects, then the caller function also starts to have side effects (and so on in the chain of calls). Recursion in calls is allowed. Also, one function can call another several times. You need to determine how many functions with side effects are present in the program after each addition of a function call by the programmer.

### Input

The first line of the input file contains three integers  $N$ ,  $K$  and  $M$ , where  $N$  is total number of functions,  $K$  is initial number of functions with side effects,  $M$  is number of function calls added by the programmer ( $1 \leq N, K, M \leq 10^5$ ;  $K \leq N$ ). Functions are numbered in order from 1 to  $N$ .

Next, there are  $K$  different numbers ranging from 1 to  $N$  in one line, being indices of functions which have side effects initially. In each of the following  $M$  lines there is a pair of integers  $a$  and  $b$ , which means that the programmer has added the call of function with the number  $b$  from the function number  $a$  ( $1 \leq a, b \leq N$ ).

### Output

For each of  $M$  additions of function calls, print the number of functions with side effects at the time after the addition of this call.

### Example

standard input	standard output
3 1 5	1
3	2
1 1	2
2 3	2
3 2	2
2 1	
3 1	

### Алгоритм

Эту задачу можно решить поиском в ширину в графе вызовов функций, чтобы найти все зависимости в вызовах. Сложность алгоритма  $O(N + M)$ .

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 using namespace std;
5 int main(int argc, const char * argv[]) {
6     int n, k, m;
7     cin >> n >> k >> m;
8     vector<vector<int>> g(n + 1, vector<int>());
9     vector<bool> side_effect(n + 1, false);
10    int func_num;
11    for (int i = 0; i < k; i++) {
12        cin >> func_num;
13        side_effect[func_num] = true;
14    }
15    int a, b;
16    int count = k;
17    for (int i = 0; i < m; i++) {
18        cin >> a >> b;
19        g[b].push_back(a);
20        if (!side_effect[a] && side_effect[b]) {
21            side_effect[a] = true;
22            count++;
23            queue<int> q;
24            q.push(a);
25            vector<bool> used(n + 1);
26            used[a] = true;
27            while (!q.empty()) {
28                int v = q.front();
29                q.pop();
30                for (size_t i = 0; i < g[v].size(); ++i) {
31                    int to = g[v][i];
32                    if (!used[to]) {
33                        used[to] = true;
34                        if (!side_effect[to]) {
35                            side_effect[to] = true;
36                            count++;
37                            q.push(to);
38                        }
39                    }
40                }
41            }
42        }
43        cout << count << endl;
44    }
45    return 0;
46 }
47 }
```

## 2.8 OpenCup GrandPrix of Europe Div 2

### Результаты

48.	MAI #6: Makarov, Rik, Yakimenko	+ 0:51	-3 2:55	+ 0:16	-	+ 0:33	-4 3:07	-	-	-	-	-11 4:46	3	101	0%	0.07
-----	---------------------------------	-----------	------------	-----------	---	-----------	------------	---	---	---	---	-------------	---	-----	----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10327](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10327)

## Задача А - ASCII Addition

Требуется сложить два целых числа, представленных в виде *ASCII art*.

*ASCII art* представляет собой матрицу высоты 7 из символов, при этом в матрице могут встречаться только точка и строчная латинская буква ‘x’.

Дано выражение в форме  $a + b$ , где  $a$  и  $b$  — целые положительные числа в десятичной записи. Выражение записано в *ASCII art* следующим образом: все символы (цифры чисел  $a$  и  $b$  и знак ‘+’) представляют собой матрицы  $7 \times 5$ , после чего они объединяются в общую картинку так, что между любыми двумя соседними матрицами-символами вставляется колонка, состоящая из точек.

Вот матрицы для всех цифр и знака ‘+’: are as follows:

```
xxxxx . . . x xxxx . . . x . . . .  
x . . . x . . . x . . . x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x  
x . . . x . . . x . . . x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x  
x . . . x . . . x xxxx . . . xxxx  
x . . . x . . . x x . . . . . . . x . . . x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x . . . x  
x . . . x . . . x x . . . . . . . x . . . x . . . x x . . . x x . . . x x . . . x x . . . x x . . . x . . . x  
xxxxx . . . x xxxx . . . xxxx
```

По заданному выражению в *ASCII art* виде найдите сумму и также запишите её в *ASCII art* в соответствии с вышеуказанными правилами.

### Input

Вход состоит ровно из 7 строк и содержит *ASCII art* запись выражения в форме  $a + b$ , где  $a$  и  $b$  — целые положительные числа, записываемые не более, чем 9 десятичными цифрами без ведущих нулей.

### Output

Выведите 7 строк, содержащие *ASCII art* запись суммы (тоже без ведущих нулей).

### Example

standard input
....x.xxxxx.xxxxx.x...x.xxxxx.xxxxx.xxxxx....xxxxx.xxxxx.xxxxx
....x.....x....x.x....x.....x....x....x....x.x....x.x....x
....x.....x.x....x.x....x.....x....x....x.x....x.x....x.x....x
....x.xxxxx.xxxxx.xxxxx.xxxxx.xxxxx....x.xxxxx.xxxxx.xxxxx.x...x
....x.x.....x....x.x....x.....x....x....x.x....x....x.x....x
....x.x.....x....x.x....x.....x....x....x.x....x....x.x....x
....x.xxxxx.xxxxx....x.xxxxx.xxxxx....x....xxxxx.xxxxx.xxxxx

standard output
....x.xxxxx.xxxxx.x...x.xxxxx.xxxxx
....x.....x....x.x....x....x.....x
....x.....x....x.x....x....x.....x
....x.xxxxx.xxxxx.xxxxx.xxxxx....x
....x.x.....x....x....x.....x....x
....x.x.....x....x....x.....x....x
....x.xxxxx.xxxxx....x....xxxxx....x

### Алгоритм

Задача просто на реализацию.

## Исходный код

```
1 my %digits = (
2     "xxxxx" .
3     "x...x" .
4     "x...x" .
5     "x...x" .
6     "x...x" .
7     "x...x" .
8     "xxxxx" => 0 ,
9     "...x" .
10    "...x" .
11    "...x" .
12    "...x" .
13    "...x" .
14    "...x" .
15    "...x" => 1 ,
16
17    "xxxxx" .
18    "...x" .
19    "...x" .
20    "xxxxx" .
21    "x...." .
22    "x...." .
23    "xxxxx" => 2 ,
24
25    "xxxxx" .
26    "...x" .
27    "...x" .
28    "xxxxx" .
29    "...x" .
30    "...x" .
31    "xxxxx" => 3 ,
32
33    "x...x" .
34    "x...x" .
35    "x...x" .
36    "xxxxx" .
37    "...x" .
38    "...x" .
39    "...x" => 4 ,
40
41    "xxxxx" .
42    "x...." .
43    "x...." .
44    "xxxxx" .
45    "...x" .
46    "...x" .
47    "xxxxx" => 5 ,
48
49    "xxxxx" .
50    "x...." .
51    "x...." .
52    "xxxxx" .
53    "x...x" .
54    "x...x" .
55    "xxxxx" => 6 ,
56
57    "xxxxx" .
58    "...x" .
```

```

59      " .... x" .
60      " .... x" .
61      " .... x" .
62      " .... x" .
63      " .... x" => 7 ,
64
65      "xxxxx" .
66      "x ... x" .
67      "x ... x" .
68      "xxxxx" .
69      "x ... x" .
70      "x ... x" .
71      "xxxxx" => 8 ,
72
73      "xxxxx" .
74      "x ... x" .
75      "x ... x" .
76      "xxxxx" .
77      " .... x" .
78      " .... x" .
79      "xxxxx" => 9 ,
80
81      " .... " .
82      "... x .." .
83      "... x .." .
84      "xxxxx" .
85      "... x .." .
86      "... x .." .
87      "... . . ." => "+"
88      );
89 my %dig;
90 foreach my $key (keys %digits) {
91     $dig{$digits{$key}} = $key;
92 }
93 my @screen;
94 for (my $i = 0; $i < 7; ++$i) {
95     my $str = <>;
96     chomp $str;
97     my $n = length($str)+1;
98     my $dig_cnt = $n/6;
99     for (my $j = 0; $j < $dig_cnt; ++$j) {
100        my $a = substr substr($str, $j*6, 6), 0, 5;
101        $screen[$j] .= $a;
102    }
103 }
104 my $expr = "";
105 foreach my $s (@screen) {
106     $expr .= $digits{$s};
107 }
108 my $res = eval $expr;
109 for (my $i = 0; $i < 7; ++$i) {
110     my $str = "";
111     foreach my $d (split "", $res) {
112         my $a = substr $dig{$d}, $i*5, 5;
113         $str .= $a . ".";
114     }
115     chop $str;
116     print $str, "\n";
117 }
```

## Задача С - Counting Cities

Карл часто ездит в командировки. Каждая командировка представляет собой посещение ровно одного города.

Сейчас новый начальник Карла хочет узнать, во сколько различных городов Карл съездил в командировки. Босс попросил по списку командировок Карла посчитать, в каком количестве города Карл был хотя бы однажды.

### Input

Первая строка входа содержит целое положительное число  $T \leq 50$  — количество тестовых примеров.

Первая строка каждого тестового примера содержит одно целое положительное число  $n$  ( $1 \leq n \leq 100$ ), обозначающее количество поездок Карла в командировке. Каждая из последующих  $n$  строк содержит одно непустое слово, состоящее из строчных латинских букв и имеющее длину не более 20 — имя города, в который ездил Карл во время очередной командировки.

### Output

Для каждого тестового примера выведите одну строку, содержащую количество различных городов, которые Карл посетил за время командировок.

### Examples

standard input	standard output
2	4
7	1
zagreb	
krakow	
warsaw	
krakow	
prague	
zagreb	
krakow	
3	
barnaul	
barnaul	
barnaul	

### Алгоритм

В этой задаче надо посчитать количество уникальных посещенных городов. Для этого каждую входную строку будем помещать в `std::set`, а потом просто выведем размер множества.

## Исходный код

```
1 #include <iostream>
2 #include <set>
3 #include <string>
4
5 using namespace std;
6
7 int main(int argc, const char * argv[]) {
8
9     int t;
10    cin >> t;
11    set<string> s;
12    while (t--) {
13        int n;
14        cin >> n;
15        string name;
16        for (int i = 0; i < n; i++) {
17            cin >> name;
18            s.insert(name);
19        }
20        cout << s.size() << endl;
21        s.clear();
22    }
23
24    return 0;
25 }
```

## Задача E - Electoral Estimations

В Байтландии стартовал первый тур президентских выборов. Так как в выборах участвует более двух кандидатов, то возможно, что победитель (кандидат, который получит наибольшее количество голосов) не наберёт большинства всех голосов. В этом случае будет проведён второй тур.

Для того, чтобы оценить предвыборные ожидания, байтландские учёные промоделировали голосование. Они попросили Вас написать программу, которая по количеству голосов, набранных кандидатами, определяет лидера голосования (в случае, если наибольшее количество голосов собрал один кандидат) и определяет, является ли его победа финальной или же потребуется второй тур.

### Input

Первая строка входа содержит целое положительное число  $T \leq 500$  — количество тестовых примеров.

Первая строка каждого тестового примера содержит целое положительное число  $n$  ( $2 \leq n \leq 10$ ) — количество кандидатов.  $i$ -я из последующих  $n$  строк содержит целое неотрицательное число  $v_i$  ( $0 \leq v_i \leq 50\,000$ ) — количество голосов, поданных за кандидата с номером  $i$ . Гарантируется, что в голосовании принимал участие хотя бы один избиратель.

### Output

Для каждого тестового примера выведите “Victory” и через пробел — номер выигравшего кандидата, если победитель набрал более половины количества голосов и второй тур не потребуется. В противном случае выведите “Leader” и номер лидирующего кандидата, если ровно один кандидат набрал наибольшее количество голосов; если же таких кандидатов как минимум два, выведите “Tie”. Кандидаты пронумерованы последовательными целыми числами от 1 до  $n$ .

### Example

standard input	standard output
4	Victory 2
3	Leader 1
10	Tie
21	Leader 4
10	
3	
20	
10	
10	
3	
10	
10	
10	
4	
15	
15	
15	
45	

### Алгоритм

Для решения этой задачи нужно подсчитать голоса для каждого кандидата, отсортировать их и затем найти отношения количества голосов за победителя к остальным, чтобы определить, является он абсолютным победителем или нет.

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 typedef struct {
5     int number;
6     int votes;
7 } Person;
8 bool cmp(const Person &a, const Person &b) {
9     return a.votes > b.votes;
10 }
11 using namespace std;
12 int main(int argc, const char * argv[]) {
13     int t;
14     cin >> t;
15     while (t--) {
16         int n;
17         cin >> n;
18         int total_votes = 0;
19         vector<Person> persons(n);
20         for (int i = 0; i < n; i++) {
21             persons[i].number = i + 1;
22             cin >> persons[i].votes;
23             total_votes += persons[i].votes;
24         }
25         sort(persons.begin(), persons.end(), cmp);
26         int top_votes = persons[0].votes;
27         int half_votes = total_votes / 2;
28         if (t == 1) {
29             cout << " " << endl;
30         }
31         if (top_votes > half_votes) {
32             if (persons[1].votes != top_votes) {
33                 cout << "Victory " << persons[0].number << endl;
34             } else {
35                 cout << "Tie" << endl;
36             }
37         } else {
38             if (persons[1].votes != top_votes) {
39                 cout << "Leader " << persons[0].number << endl;
40             } else {
41                 cout << "Tie" << endl;
42             }
43         }
44     }
45     return 0;
46 }
```

## 2.9 OpenCup GrandPrix of Peterhof Div 2

### Результаты

13.	MAI #6: Makarov, Rik, Yakimenko	+ 0:30	-	+ 2:39	-4 2:59	+3 2:35	-5 4:59	+ 0:53	-	-	4	459	42%	0.31
-----	---------------------------------	-----------	---	-----------	------------	------------	------------	-----------	---	---	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10328](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10328)

## Задача А - (a, b)-башня

Маленький Артём задался вопросом: как, записав всего несколько цифр, получить очень большое число? Конечно, один из способов — написать башню из степеней. Как известно, башня из степеней вычисляется сверху вниз: в выражении вида  $x^{y^z}$  сначала вычисляется  $y^z$ , а затем число  $x$  возводится в полученную степень. Например, короткая запись  $2^{3^4}$  равна огромному числу  $2^{3 \cdot 3 \cdot 3} = 2^{81} = 2417851639229258349412352$ .

Артём выписал целые числа от  $a$  до  $b$  по диагонали, двигаясь вверх и вправо. Получилась башня из степеней:

$$a^{(a+1)^{(a+2)^{\dots^{(b-1)^b}}}}$$

Помогите ему выяснить, чему равна последняя десятичная цифра этого — возможно, огромного — числа.

### Формат входных данных

В первой строке записано два целых числа  $a$  и  $b$  — параметры башни ( $1 \leq a \leq b \leq 10$ ).

### Формат выходных данных

Выведите одно число — последнюю десятичную цифру числа

$$a^{(a+1)^{(a+2)^{\dots^{(b-1)^b}}}}$$

### Примеры

abtower.in	abtower.out
2 4	2
3 3	3

## Алгоритм

Задача решается с помощью рекурсии. Для того, чтобы узнать последнюю цифру конечного числа, на каждом шаге рекурсии можно отбрасывать все кроме последней цифры промежуточного числа.

## Исходный код

```
1 #!/usr/bin/env ruby
2 def func(l, r)
3   if l < r
4     (l ** func(l+1, r))%10
5   else
6     r%10
7   end
8 end
9 a, b = gets.split.map{|x| x.to_i}
10 puts func(a,b)
```

## Задача С - Отношение эквивалентности

Егор изучает понятие отношения эквивалентности.

*Отношением эквивалентности* на множестве  $S$  называется такое подмножество пар  $R = \{(x, y) | x \in S, y \in S\}$ , что выполнены три условия:

- для всех  $x$  верно  $(x, x) \in R$ ;
- если  $(x, y) \in R$ , то  $(y, x) \in R$ ;
- если  $(x, y) \in R$  и  $(y, z) \in R$ , то  $(x, z) \in R$ .

Несложно показать, что при таких условиях множество  $S$  можно разбить на несколько классов, так, что элементы в одном классе попарно эквивалентны, а в разных не эквивалентны. Такие классы называются *классами эквивалентности*.

Введём следующее отношение эквивалентности на строках из нулей и единиц: строки называются эквивалентными, если можно получить одну из другой с помощью цепочки следующих операций.

- Удаление или вставка двух нулей подряд в любом месте строки.
- Удаление или вставка трёх единиц подряд в любом месте строки.

Операции можно совершать сколько угодно раз в любом порядке.

Вам дано множество строк. Выведите разбиение этого множества на классы эквивалентности.

### Формат входных данных

В первой строке дано число  $n$  ( $2 \leq n \leq 4096$ ). В следующих  $n$  строках дано по одной строке из множества. Все строки непустые, а их суммарная длина не превосходит 50 000 символов.

### Формат выходных данных

В первой строке выведите одно число  $k$  — количество классов эквивалентности. В следующих  $k$  строках выведите сами классы. Каждый класс следует выводить в следующем формате: сначала выведите количество элементов в классе, потом номера самих элементов в возрастающем порядке. Классы следует выводить в порядке возрастания наименьшего номера в классе. Строки нумеруются с единицы в порядке, в котором они заданы.

### Пример

equivalence.in	equivalence.out
5	4
0101010101	1
101010110101011	2
0101	3
0000	4 5
111	

### Алгоритм

Задача на реализацию. Необходимо найти классы эквивалентности.

## Исходный код

```
1 n = gets.to_i
2
3 data = Array.new
4
5 for i in 0...n
6     s = gets.chomp
7     data.push(s)
8 end
9
10 data.each do |s|
11     begin
12         s_copy = s.clone
13         s.gsub!(/0{2}/, '')
14         s.gsub!(/1{3}/, '')
15     end until s == s_copy
16 end
17
18 h = Hash.new()
19
20 data.each_index do |i|
21     s = data[i]
22     if h[s] == nil
23         h[s] = Array.new
24     end
25     h[s].push(i + 1)
26 end
27
28 res = Array.new
29
30 h.each_value do |v|
31     res.push(v)
32 end
33
34 res.sort! do |a, b|
35     a.first <=> b.first
36 end
37
38 puts "#{res.size}"
39
40 res.each do |l|
41     l.each do |x|
42         print "#{x} "
43     end
44     puts
45 end
```

## Задача Е - Идеальная фотография

Опять в финале Лиги Чемпионов встречаются команды Реал Мадрид и Челси. По традиции перед началом матча обе команды должны спеть гимны своих стран. Суммарно в каждой из команд (включая тренеров и помощников) оказалось  $N$  человек. Все они выстраиваются в два ряда параллельно средней линии, разделяющей две половины поля (её уравнение  $y = 0$ ). Члены команды Реал Мадрид встали на прямую  $y = 1$ , а Челси на прямую  $y = -1$ . Съёмка с вертолёта показала, что  $i$ -й игрок Реала имеет координаты  $(r_i, 1)$ , а  $i$ -й игрок Челси стоит в точке  $(c_i, -1)$ . После исполнения гимна они все должны встать на среднюю линию для общей групповой фотографии.

Главный фотограф решил, что для получения самой эффектной фотографии все члены команды должны образовать ряд так, чтобы между двумя соседними людьми в ряду расстояние было равно 1. Чтобы футболисты не переутомились, вам требуется выбрать позиции и сказать футболистам, кто должен куда переместиться, так, чтобы суммарное пройденное ими расстояние было минимально.

### Формат входных данных

Первая строка содержит одно целое число  $N$  ( $1 \leq N \leq 100$ ). Вторая строка содержит  $N$  целых чисел  $r_i$  ( $-10^6 \leq r_i \leq 10^6$ ). Третья строка содержит  $N$  целых чисел  $c_i$  ( $-10^6 \leq c_i \leq 10^6$ ). Все  $r_i$  различны, все  $c_i$  также различны.

### Формат выходных данных

Выведите одно число — какое минимальное суммарное расстояние пройдут игроки до своих целей, чтобы удовлетворить всем требованиям фотографа.

Ваш ответ будет считаться правильным, если его абсолютная или относительная погрешность не будет превосходить  $10^{-9}$ . А именно: пусть ваш ответ равен  $a$ , а ответ жюри равен  $b$ . Проверяющая программа будет считать ваш ответ правильным, если  $\frac{|a-b|}{\max(1,b)} \leq 10^{-9}$ .

### Пример

make-a-row.in	make-a-row.out
3 1 4 5 -1 3 5	7.3730791040629358

### Алгоритм

Нужно единожды выстроить всех людей в одну линию по порядку и потом двигать эту линию, пока расстояние не станет наименьшим с заданной точностью.

## Исходный код

```
1 #!/usr/bin/env ruby
2 def calc_sum(m, i)
3   s = 0
4   for a in m do
5     s += Math.sqrt((a-i)*(a-i) + 1)
6     i += 1
7   end
8   return s
9 end
10 n = gets.to_i
11 r = gets.split.map{|x| x.to_i}
12 c = gets.split.map{|x| x.to_i}
13 m = (r + c).sort
14 x = 0
15 step = 1000000
16 sum_curr = calc_sum(m, x)
17 sum_prev = sum_curr + 1
18 sign = 1
19 x += step
20 while ((sum_prev-sum_curr).abs > 0.0000000001)
21   sum_prev = sum_curr
22   sum_curr = calc_sum(m, x)
23   if sum_curr > sum_prev && sign == 1
24     sign = -1
25     step /= 2.0
26     sign_counter = 0
27   elsif sum_curr > sum_prev && sign == -1
28     sign = 1
29     step /= 2.0
30     sign_counter = 0
31   end
32   x += step*sign
33 end
34 puts sum_curr
```

## Задача G - Случайные тоннели

*Это интерактивная задача.*

В галактике Туманный Путь — бесконечное количество звёзд, пронумерованных целыми числами. Чтобы быстро перемещаться между ними, можно пользоваться червоточинами — надпространственными тоннелями, соединяющими звёзды. Однако не любая пара звёзд соединена таким тоннелем.

Когда демиург создавал Туманный Путь, он задал вероятность  $p$  того, что для любой пары целых чисел  $(i, j)$  от звезды  $i$  есть прямой тоннель к звезде  $j$ , а далее предоставил конструирование тоннелей воле случая. Вероятность существования каждого тоннеля не зависит от вероятности существования любых других тоннелей. Все тоннели односторонние, то есть при  $i \neq j$  тоннель от  $i$  к  $j$  и тоннель от  $j$  к  $i$  — это два разных тоннеля, каждый из которых существует с вероятностью  $p$ .

Звездолёт находится у звезды с номером 0. Навигационный компьютер звездолёта имеет ограниченную функциональность: он может принимать запрос вида «следует переместиться по прямому тоннелю к звезде  $x$ », где  $x$  — целое число от 0 до  $2^{32} - 1$ . После такого запроса, если существует тоннель от звезды, где находится звездолёт, к звезде  $x$ , звездолёт перемещается туда, а на табло загорается слово «yes». В противном случае на табло появляется слово «no», а звездолёт остаётся на месте.

Капитан звездолёта хочет попасть к звезде с номером 1. Вы — навигатор этого звездолёта. Помогите капитану оказаться у нужной звезды!

### Протокол взаимодействия

Решение должно выводить каждый запрос в стандартный поток вывода на отдельной строке. Запрос — это одно целое число  $x$  от 0 до  $2^{32} - 1$  — номер звезды, к которой следует переместить звездолёт при наличии прямого тоннеля.

Чтобы предотвратить буферизацию вывода, после каждого выведенного запроса следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Ответ на запрос — слово «yes» или «no» — решение получает в стандартный поток ввода, также на отдельной строке.

Как только звездолёт оказался у звезды с номером 1, решение должно сразу же корректно завершить свою работу.

После 30 000 запросов к навигационному компьютеру у него кончается электричество, и следующий запрос сделать не удаётся. В таком случае миссия считается проваленной.

В каждом teste к этой задаче зафиксирована вероятность  $p$  (целое число от 3 до 99 в процентах). После этого для каждой возможной пары звёзд зафиксировано, есть ли между ними тоннель.

### Пример

запросы участника	ответы проверяющей программы
1	no
3	yes
3	no
1	yes

### Алгоритм

Для выбора следующей звезды можно использовать случайные числа и при успешном перемещении на эту звезду нужно проверить, существует ли путь до звезды 1, если нет, то повторить алгоритм сначала.

## Исходный код

```
1 #!/usr/bin/env ruby
2 def send_cmd(cmd)
3   puts cmd
4   STDOUT.flush
5 end
6 a = 1
7 while (true) do
8   send_cmd a
9   case gets
10  when /yes/
11    exit if a == 1
12    a = 1
13  when /no/
14    a = rand(1<<31)
15 end
16 end
```

## 2.10 OpenCup Grand Prix of Asia Div 2

### Результаты

15.	MAI #6: Makarov, Rik, Yakimenko	-	+7 3:14	-	-	+1 0:31	-	-	-1 3:02	+5 1:03	+ 0:16	-	3	232	66%	0.16
-----	---------------------------------	---	------------	---	---	------------	---	---	------------	------------	-----------	---	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10329](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10329)

## Задача E - Mirror Rice Cake

Зеркальный рисовый пирог (стопка рисовых пирогов) — японское кушанье, которое готовится на Новый Год.

У Snuke есть  $N$  рисовых пирогов и он хочет сложить их в стопку.

Вес  $i$ -го рисового пирога равен  $a_i$ . Чтобы собрать Зеркальный рисовый пирог, нужно сложить пироги один на другого так, чтобы для каждого рисового пирога суммарный вес всех пирогов выше него был строго меньше, чем его собственный вес.

Snuke хочет узнать, какое наибольшее количество рисовых пирогов он может использовать для Зеркального рисового пирога.

### Input

Первая строка входа содержит одно целое число  $N$  ( $1 \leq N \leq 1000$ ). Каждая из последующих  $N$  строк содержит вес  $a_i$  очередного пирога ( $1 \leq a_i \leq 10^9$ ).

### Output

Выведите одно целое число — ответ к задаче.

### Example

standard input	standard output
5	3
3	
20	
5	
8	
6	

### Алгоритм

Сортируем и складываем все числа до тех пор, пока общая сумма не станет больше следующего слагаемого.

### Исходный код

```
1 public class Test {
2     public static void main(String [] args) {
3         Scanner sc = new Scanner(System.in);
4         int N = sc.nextInt();
5         long a[] = new long[N];
6         for(int i = 0; i < N; i++) {
7             a[i] = sc.nextInt();
8         }
9         int count = 1;
10        Arrays.sort(a);
11        long sum = a[0];
12        for(int i = 1; i < N; i++) {
13            if(a[i] > sum) {
14                count++;
15                sum += a[i];
16            }
17        }
18        System.out.println(count);
19    }
```

## Задача L - String Modification

На Новый Год Snuke получил от Укконена в подарок строку  $s$ . Определите, сможет ли Snuke переделать эту строку в строку  $t$ , повторяя какое-то (возможно, нулевое) количество раз следующую операцию:

Выбрать символ в строке  $s$  и вставить после этого символа какой-либо другой, обязательно отличный от данного.

Например, он может переделать “`abca`” в “`adbca`” за одну операцию, выбрав первую букву ‘`a`’ и вставив сразу после неё ‘`d`’. А вот “`abca`” в “`aabca`” он переделать не может.

### Input

Первая строка входа содержит строку  $s$ , вторая — строку  $t$ . Обе строки составлены из строчных латинских букв и для них выполняется ограничение  $1 \leq |s| \leq |t| \leq 5000$ .

### Output

Выведите “`Yes`” в случае, когда Snuke может переделать  $s$  в  $t$ , и “`No`” в противном случае.

### Examples

standard input	standard output
<code>snuke</code> <code>snukent</code>	<code>Yes</code>
<code>snuke</code> <code>ssnuke</code>	<code>No</code>

## Алгоритм

Задача на реализацию.

### Исходный код

```
1 a = gets.chomp; b = gets.chomp
2 i = 0; j = 0
3 f = false
4 fc = b[0]
5 while j < b.length do
6   f = true if fc != b[j]
7   if i < a.length && a[i] == b[j]
8     i += 1
9     j += 1
10  elsif i == 0
11    puts 'No'; exit
12  else
13    if b[j-1] == b[j] && !f
14      puts 'No'; exit
15    end
16    j += 1
17  end
18 end
19 if i == a.length
20   puts 'Yes'
21 else
22   puts 'No'
23 end
```

## Задача N - Soccer Match

Результатом футбольного матча для одной команды может быть либо победа, либо ничья, либо поражение. За победу команда получает 3 очка, за ничью — одно очко, за поражение — 0.

В старой книге по истории футбола вы нашли упоминание о том, что в первом сезоне в профессиональной лиге команда, за которую Вы болеете, сыграла  $N$  матчей и набрала  $K$  очков.

Вы хотите выяснить список всех возможных распределений результатов (то есть количество побед, ничьих и поражений в сезоне). Напишите программу, которая выводит все такие распределения.

### Input

Вход содержит два целых числа  $N$  ( $0 < N \leq 100$ ) и  $K$  ( $0 \leq K \leq 300$ ) — количество сыгранных матчей и количество набранных при этом очков, разделённые пробелом.

Гарантируется, что существует как минимум одно возможное сочетание побед, ничьих и поражений, при которых сыграно  $N$  матчей и набрано  $K$  очков.

### Output

Выведите список возможных результатов по одному в строке. Каждый результат состоит из трёх целых чисел, разделённых пробелами — количества побед  $w$ , количества ничьих  $t$  и количества поражений  $l$ . Результаты должны выводиться в порядке убывания количества побед.

### Example

standard input	standard output
6 10	3 1 2 2 4 0

### Алгоритм

Перебор всех возможных вариантов с использованием формул. Сложность  $O(n)$ .

### Исходный код

```
1 input = gets.split(' ').map(&:to_i)
2 games = input.first
3 score = input.last
4 wins = score / 3
5 ties = 0
6 losses = 0
7 while wins > games
8   wins = wins - 1
9 end
10 while wins > -1
11   ties = score - wins * 3
12   losses = games - wins - ties
13   break if ties < 0 or losses < 0
14   puts "#{wins} #{ties} #{losses}"
15   wins = wins - 1
16 end
```

## 2.11 OpenCup GrandPrix of Saratov Div 2

### Результаты

22.	MAI #6: Makarov, Rik, Yakimenko	+	-16	-	+	+11	-	-7	-	-	-	3	471	78%	0.20
-----	---------------------------------	---	-----	---	---	-----	---	----	---	---	---	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10340](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10340)

## Задача А - Алфавит

Мальчик Костя несколько недель назад начал изучение алфавита. Для упражнений мама купила ему набор картинок, на которых изображено по одному предмету или животному.

На каждой карточке написано одно слово. Костя хочет разложить карточки в стопки, чтобы первая буква у всех слов в стопке совпадала. Помогите Косте разложить карточки.

### Input

В первой строке входных данных записано одно целое число  $N$  ( $1 \leq N \leq 100$ ) — количество карточек у Кости. Далее в  $N$  строках записано по одному слову. Слова содержат только маленькие буквы латинского алфавита. Все слова на карточках различные. Длина слов не превосходит 20 символов.

### Output

Выведите в первой строке число  $K$  — количество стопок, в которые Костя должен разложить все карточки. Далее выведите  $K$  строк. В каждой из этих строк выведите слова на карточках, которые попали в одну стопку. Слова в строках должны быть разделены пробелами.

Слова в строках должны быть упорядочены лексикографически. Стопки необходимо выводить в алфавитном порядке.

### Examples

standard input	standard output
4 apple cat dog angel	3 angel apple cat dog
3 he she it	3 he it she

## Алгоритм

Группируем все слова по первой букве с помощью ассоциативного массива и затем выводим в отсортированном виде.

## Исходный код

```
1 n = gets.chomp.to_i
2 h = {}
3 for i in 0...n
4   word = gets.chomp
5   c = word[0]
6   unless h.has_key?(c)
7     h[c] = []
8   end
9   h[c].push(word)
10 end
11 puts h.size
12 for a in h.keys.sort
13   puts h[a].sort.join(' ')
14 end
```

## Задача D - Возрастающие последовательности

Не так давно мальчик Костя изучил сравнения целых чисел, но так как эта тема оказалась для него слишком простой, то папа рассказал ему про новый для мальчика математический объект — строго возрастающие последовательности.

Будем говорить, что последовательность  $A = (a_1, a_2, \dots, a_n)$ , ( $n > 1$ ) строго возрастающая, если выполнено условие  $a_1 < a_2 < \dots < a_n$ .

Папа предложил Косте потренироваться в определении строго возрастающих последовательностей. Для этого он выписал  $M$  пар чисел  $(l_i, r_i)$  и для каждой из них попросил определить, является ли последовательность  $A_i = (a_{l_i}, a_{l_i+1}, \dots, a_{r_i})$  строго возрастающей.

Костя потратил много времени на решение этого упражнения. Помогите ему проверить свой результат, найдите правильные ответы для поставленной задачи.

### Input

В первой строке входных данных записаны два целых числа  $N$  и  $M$  ( $1 \leq N, M \leq 5 \cdot 10^5$ ). Во второй строке записаны  $N$  целых чисел  $a_i$  ( $1 \leq a_i \leq 10^6$ ). Далее в  $M$  строках записаны пары целых чисел  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq N$ ).

### Output

Выведите одну строку состоящую из  $M$  символов.  $i$ -ый символ должен быть равен Y, если последовательность  $A_i = (a_{l_i}, a_{l_i+1}, \dots, a_{r_i})$  строго возрастающая. Иначе  $i$ -ый символ должен быть равен N.

### Examples

standard input	standard output
7 7 1 3 5 6 2 4 7 1 7 1 4 2 5 2 4 5 7 4 7 4 4	NYNYYNN

### Алгоритм

Задача решается простым перебором. Сложность  $O(n)$ .

## Исходный код

```
1 #include <iostream>
2 using namespace std;
3 ULL myAbs(ULL a, ULL b) {
4     if(a > b) {
5         return a - b;
6     } else {
7         return b - a + 1;
8     }
9 }
10 typedef struct {
11     int first;
12     int second;
13     int d;
14 } dist;
15 bool comp(const dist &a, const dist &b) {
16     return a.d > b.d;
17 }
18 int main() {
19     int temp;
20     int N, M;
21     cin >> N >> M;
22     vector <int> up(N, 0);
23     int cur, prev;
24     cin >> prev;
25     int k = 0;
26     for(int i = 1; i < N; i++) {
27         cin >> cur;
28         if(cur > prev) {
29             up[i] = k;
30         }
31         else {
32             k++;
33             up[i] = k;
34         }
35         prev = cur;
36     }
37     int l, r;
38     for(int i = 0; i < M; i++) {
39         cin >> l >> r;
40         if(up[l-1] == up[r-1] && l != r) {
41             cout << "Y";
42         }
43         else {
44             cout << "N";
45         }
46     }
47     return 0;
48 }
```

## Задача Е - Карточки

У мальчика Кости есть  $N$  карточек, на которых написано по одному целому числу от 1 до 1000. Костя хочет выбрать какие-то из этих карточек и разложить их парами.

Определите какое максимальное количество пар карточек может выбрать Костя, чтобы для каждой пары сумма чисел, написанных на них была равна одному и тому же числу  $S$ .

### Input

В первой строке входных данных записано одно целое число  $N$  ( $2 \leq N \leq 500$ ). Во второй строке записаны  $N$  целых чисел  $a_i$  ( $1 \leq a_i \leq 1000$ ). Числа в строке разделены одиночными пробелами.

### Output

В единственной строке выведите максимальное количество пар карточек с одинаковой суммой, которые может выбрать Костя.

### Examples

standard input	standard output
7 1 2 3 4 5 6 7	3
10 2 6 8 1 3 2 6 6 2 7	4

### Алгоритм

Задача решается с помощью перебора.

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <set>
5 using namespace std;
6
7 int main() {
8     long n;
9     cin >> n;
10    vector<long> a(n);
11    for (long i = 0; i < n; ++i) {
12        long d;
13        cin >> d;
14        a[i] = d;
15    }
16    sort(a.begin(), a.end());
17    set<long> h;
18    long max_s = 1;
19    for (long i = 0; i < n; ++i) {
20        long j = n-1;
21        while (j > i) {
22            if (j-i <= 2 * max_s)
23                break;
24            long sum = a[i] + a[j];
25            if (h.find(sum) != h.end()) {
26                j -= 1;
27                continue;
28            }
29            else {
30                h.insert(sum);
31            }
32            long count = 1;
33            long y = j-1;
34            for (long k = i+1; k <= y; ++k) {
35                bool f = false;
36                long x = y;
37                while (!f && x > k) {
38                    long sum_k = a[k] + a[x];
39                    if (sum_k == sum) {
40                        count += 1;
41                        y = x - 1;
42                        f = true;
43                    }
44                    else if (sum_k < sum) {
45                        break;
46                    }
47                    x -= 1;
48                }
49            }
50            if (count > max_s) {
51                max_s = count;
52            }
53            j -= 1;
54        }
55    }
56    cout << max_s << '\n';
57    return 0;
58 }
```

## 2.12 OpenCup GrandPrix of China Div 2

### Результаты

37.	MAI #6: Makarov, Rik, Yakimenko	-	-	-	-	+ 0:39	+2 0:49	+2 2:09	+5 3:23	-9 4:59	4	602	69%	0.17
-----	---------------------------------	---	---	---	---	-----------	------------	------------	------------	------------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10341](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10341)

## Задача L - Cut The Rectangle

Два треугольника заданы длинами сторон. Определите, можно ли получить эти треугольники, разрезав прямоугольник одним отрезком на две части и затем, возможно, повернув получившиеся части каким-либо образом или отразив их.

### Input

Первая строка входа содержит три целых числа — стороны первого треугольника. Вторая строка входа содержит три целых числа — стороны второго треугольника.

Гарантируется, что заданные тройки чисел могут быть сторонами треугольника ненулевой площади. Длина каждой стороны треугольника не менее 1 и не более 100.

### Output

Если существует прямоугольник, который может быть разрезан так, чтобы получились два заданных треугольника, выведите 1. В противном случае выведите 0.

### Examples

standard input	standard output
6 8 10 8 6 10	1
7 4 5 4 5 7	0

### Алгоритм

Для решения задачи нужно просто проверить равенство треугольника. Сложность  $O(1)$ .

### Исходный код

```
1 a = gets.split(' ').map(&:to_i)
2 b = gets.split(' ').map(&:to_i)
3 as = a.sort.join(',')
4 bs = b.sort.join(',')
5 if (a[0]**2 + a[1]**2 == a[2]**2 ||
6     a[0]**2 + a[2]**2 == a[1]**2 ||
7     a[2]**2 + a[1]**2 == a[0]**2 &&
8     as == bs)
9   puts 1
10 else
11   puts 0
12 end
```

## Задача M - Diversity

Определим *разнообразность* строки как количество попарно различных букв в строке. Например, строка “`асш`” имеет разнообразность 3, равно как и строка “`icpc`”.

Бобо нравятся строки, разнообразность которых равна 1 или 2. Ему подарили некоторую строку; Бобо хочет превратить её в строку, которая ему нравится. За одно действие он может удалить одну букву в любом месте строки. За какое минимальное количество действий Бобо сумеет превратить данную строку в строку с разнообразностью 2 или менее?

### Input

На вход подаётся непустая строка, состоящая не более, чем из 100 строчных латинских букв.

### Output

Выведите одно целое число — наименьшее количество действий, которые потребуются Бобо, чтобы превратить данную строку в строку, которая ему нравится.

### Examples

standard input	standard output
<code>bobo</code>	0
<code>china</code>	3
<code>acmicpc</code>	3

### Алгоритм

Нужно удалять из слова буквы, которых меньше всего, до тех пор, пока различных букв не станет 1 или 2 штуки.

### Исходный код

```
1 aa = gets.chomp
2 h = {}
3 count = 0
4 for i in 0...a.size
5   if h.has_key?(a[i])
6     h[a[i]] += 1
7   else
8     h[a[i]] = 1
9     count += 1
10  end
11 end
12 if count <= 2
13   puts 0
14 else
15   m = h.values.sort
16   sum = 0
17   n = m.size - 2
18   for i in 0...n
19     sum += m[i]
20   end
21   puts sum
22 end
```

## Задача N - Mechanics

На плоскости размещён набор из  $n$  зубчатых колёс. Центр каждого зубчатого колеса находится в точке с целыми координатами, радиус также является целым. Требуется вычислить, что будет происходить с последним перечисленным колесом при попытке вращения первого. Есть следующие варианты:

- Первое колесо заблокировано и не может двигаться, так как какое-то колесо при его вращении должно будет вращаться в две противоположные стороны одновременно.
- Первое колесо может двигаться, однако оно не соединено с последним колесом, которое тем самым остаётся неподвижным.
- Первое колесо приводит последнее в движение с определённым передаточным отношением.

В случае, если первое колесо заблокировано, выведите эту информацию вне зависимости от того, соединено ли первое колесо с последним.

### Input

Первая строка входа содержит целое число  $n$  ( $1 \leq n \leq 1,000$ ) — количество зубчатых колёс. Далее следуют  $n$  строк, каждая из которых задаёт одно зубчатое колесо и содержит три целых числа  $x, y$  ( $-10^4 \leq x, y \leq 10^4$ ) и  $r$  ( $1 \leq r \leq 10^4$ ) — координаты оси колеса и его радиус. Первое колесо (которое вращают) перечислено первым, колесо, информацию о движении которого надо вывести, перечислено  $n$ -м.

Считается, что два любых зубчатых колеса, которые касаются, зацеплены между собой. Гарантируется, что никакие два зубчатых колеса не пересекаются по фигуре ненулевой площади.

### Output

Выведите строку со следующим содержанием:

- $-1$ , если первое колесо заблокировано.
- $0$ , если первое колесо может вращаться, но при этом последнее колесо остаётся неподвижным.
- $a : b$ , если при вращении первого колеса последнее тоже вращается;  $a$  и  $b$  — два целых числа, разделённых пробелом, и  $a : b$  — отношение частоты вращения первого колеса к последнему, причём  $a$  и  $b$  взаимно просты,  $a$  всегда положительно, а  $b$  положительно, если последнее колесо вращается в ту же сторону, что и первое, и отрицательно в противном случае.

## Example

standard input	standard output
2 0 0 10 0 30 20	2 -1
2 0 0 1 0 3 1	0
3 0 0 11 0 33 22 44 0 33	-1

## Алгоритм

Задача на реализацию. Нужно учитывать, что если соединения колёс от первого к последним образуют дерево, то колёса могут крутиться, и направление у колёс чередуется. Также, для подсчёта частоты вращения требуется сокращать дробь на каждом шаге.

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 using namespace std;
5 #define ll long long
6 #define ull unsigned long long
7 #define eps 0.0000000001
8 struct Gear {
9     ll x, y, r;
10    bool f;
11    char sign;
12    ll a, b;
13    vector<ll> list;
14    ll parent;
15    Gear(ll a, ll b, ll c) {
16        x = a;
17        y = b;
18        r = c;
19        f = 0;
20    }
21 };
22 double eq_eps(double a, double b) {
23     return fabs(a - b) < eps;
24 }
25 int main() {
26     ll n;
27     cin >> n;
28     vector<Gear *> v(n);
29     for (int i = 0; i < n; ++i) {
30         ll a, b, c;
31         cin >> a >> b >> c;
32         v[i] = new Gear(a, b, c);
33     }
```

```

34     for (int i = 0; i < n; ++i) {
35         for (int j = i+1; j < n; ++j) {
36             bool res = eq_eps(sqrt(pow(v[i]->x - v[j]->x, 2) + pow(v[i]->y - v[j]->y, 2)), 
37             v[i]->r + v[j]->r);
38             if (res) {
39                 v[i]->list.push_back(j);
40                 v[j]->list.push_back(i);
41             }
42         }
43     }
44     ll x = 0;
45     v[x]->a = 1;
46     v[x]->b = 1;
47     v[x]->sign = 1;
48     queue<ll> qu;
49     qu.push(x);
50     while (!qu.empty()) {
51         x = qu.front();
52         qu.pop();
53         Gear * cur_gear = v[x];
54         for (ll i = 0; i < cur_gear->list.size(); ++i) {
55             if (cur_gear->list[i] == cur_gear->parent)
56                 continue;
57             Gear * next_gear = v[cur_gear->list[i]];
58             if (!next_gear->f) {
59                 qu.push(cur_gear->list[i]);
60                 next_gear->f = 1;
61                 next_gear->parent = x;
62             }
63             else {
64                 cout << -1 << endl;
65                 return 0;
66             }
67             next_gear->sign = cur_gear->sign * -1;
68             ll new_a, new_b, a, b;
69             a = new_a = cur_gear->r * cur_gear->a;
70             b = new_b = next_gear->r * cur_gear->b;
71             while (b != 0) {
72                 ll r = a%b;
73                 a = b; b = r;
74             }
75             next_gear->a = new_a/a;
76             next_gear->b = new_b/a;
77         }
78     }
79     Gear * last_gear = v[n-1];
80     if (last_gear->f == 0) {
81         cout << 0 << endl;
82         return 0;
83     }
84     ll a = last_gear->a;
85     ll b = last_gear->b;
86     while (b != 0) {
87         ll r = a%b;
88         a = b; b = r;
89     }
90     cout << last_gear->b/a << ' ' << last_gear->sign * last_gear->a/a << endl;
91     return 0;
92 }

```

## Задача О - Pairs

Карточная игра «Pairs» играется специальными картами. Каждой карте соответствуют два целых числа: одно задаёт масть, второе задаёт цену. Карты в колоде могут повторяться.

Игрок получает от сдающего некоторый набор карт. После чего он для каждой масти, карты которой присутствуют в руке, должен сбросить ровно две карты этой масти. Если в какой-то масти ровно одна карта — игрок проиграл и получил бонус 0.

Иначе бонус определяется как сумма значений карт, которые остались у игрока.

По заданному набору карт определите, какие карты нужно сбросить, чтобы максимизировать бонус.

### Input

Первая строка входа содержит одно целое число  $N$  ( $1 \leq N \leq 299999$ ) — количество карт, которое раздал сдающий.  $i$ -я из последующих  $N$  строк задаёт  $i$ -ю слева карту в руке и содержит два целых числа: масть  $s_i$  ( $1 \leq s_i \leq 10^9$ ) и значение  $v_i$  ( $0 \leq v_i \leq 10^9$ ).

### Output

Если игрок с данным набором карт проиграет, выведите  $-1$ . Иначе выведите в первой строке целое число  $K$  — количество мастей, которые есть у игрока, затем для каждой масти в новой строке выведите два числа — номера (при перечислении в соответствии со входным файлом, слева направо, в нумерации, начиная с 1) карт, которые должны быть сброшены в этой масти.

Порядок мастерей при выводе можно выбирать произвольно. Если решений несколько, выведите любое.

### Examples

standard input	standard output
10	3
9 2	3 4
9 3	1 2
7 1	8 9
7 2	
9 4	
9 3	
1 2	
1 0	
1 0	
7 3	

### Алгоритм

Задача на реализацию.

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 using namespace std;
5 typedef struct {
6     long suit;
7     long cost;
8     int id;
9 } Card;
10 bool cmp_suit(const Card &a, const Card &b) {
11     return a.suit < b.suit;
12 }
13 bool cmp_cost(const Card &a, const Card &b) {
14     return a.cost < b.cost;
15 }
16 int main() {
17     int n;
18     cin >> n;
19     vector<Card> hand(n);
20     for (int i = 0; i < n; i++) {
21         cin >> hand[i].suit >> hand[i].cost;
22         hand[i].id = i + 1;
23     }
24     stable_sort(hand.begin(), hand.end(), cmp_cost);
25     stable_sort(hand.begin(), hand.end(), cmp_suit);
26     vector< pair<int, int> > output;
27     vector< vector<int> > jazz;
28     for (int i = 1; i <= n; i++) {
29         vector<int> temp;
30         temp.push_back(hand[i - 1].id);
31         while (i < n && hand[i].suit == hand[i - 1].suit) {
32             temp.push_back(hand[i].id);
33             i++;
34         }
35         jazz.push_back(temp);
36     }
37     for (size_t i = 0; i < jazz.size(); i++) {
38         if (jazz[i].size() < 2) {
39             cout << "-1" << endl;
40             return 0;
41         } else {
42             output.push_back(make_pair(jazz[i][0], jazz[i][1]));
43         }
44     }
45     cout << output.size() << endl;
46     for (size_t i = 0; i < output.size(); i++) {
47         cout << output[i].first << " " << output[i].second << endl;
48     }
49     return 0;
50 }
```

## 2.13 OpenCup GrandPrix of Bashkortostan Div 2

### Результаты

17.	MAI #6: Makarov, Rik, Yakimenko	-	+3 3:00	-5 4:00	-	-	-	-	+ 3:28	+1 4:31	+	0:29	+	0:45	+	1:39	+	0:47	7	962	36%	0.27
-----	---------------------------------	---	------------	------------	---	---	---	---	-----------	------------	---	------	---	------	---	------	---	------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10342](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10342)

## Задача C - Constant Ratio

Дано целое число  $N$ . Требуется найти количество способов разложить его на два или более целых слагаемых  $a_i$  так, чтобы отношение  $a_i/a_{i-1}$  не зависело от  $i$  и было целым для всех  $i > 1$ .

### Input

На вход подаётся одно целое число  $n$  ( $1 \leq n \leq 10^5$ ).

### Output

В выходной файл выведите одно число — количество искомых разбиений числа.

### Examples

стандартный ввод	стандартный вывод
1	0
5	2
567	21

### Алгоритм

Задача решается перебором.

### Исходный код

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main() {
5     long x;
6     scanf("%ld", &x);
7     long count = 0;
8     long d;
9     for (d = 1; d < x; d++) {
10         if (x % d == 0) {
11             count++;
12         }
13     }
14     long b1, q, n;
15     for (b1 = 1; b1 <= (x/3 + 1); b1++) {
16         long q_max = ceil((double)x / b1) - 1;
17         for (q = 2; q <= q_max; q++) {
18             for (n = 2; n <= 100000; n++) {
19                 long s = (b1 * ((long)pow((double)q, (double)n) - 1)) / (q - 1);
20                 if (s == x) {
21                     count++;
22                 } else if (s > x) {
23                     break;
24                 }
25             }
26         }
27     }
28     printf("%ld\n", count);
29     return 0;
30 }
```

## Задача L - Игра со строкой

Маленькая Элизабет любит играть со строками. У нее есть строка длины  $n$ , полностью состоящая из букв «а». Она делает со строкой следующие действия:

- Если первая буква в строке «а», то Элизабет дописывает в конец строки «bc», после чего удаляет из строки первые 2 буквы.
- Если первая буква в строке «b», то тогда Элизабет дописывает в конец строки «a», после чего также удаляет из строки первые 2 буквы.
- Если же первая буква в строке «c», то Элизабет дописывает в конец строки «aaa», после чего опять удаляет из строки первые 2 буквы.

Элизабет останавливается после того, как у нее получится строка, состоящая из одной буквы. Например, при  $n = 4$ , для достижения цели ей требуется 6 шагов:

aaaa → aabc → bcbc → bca → aa → bc → a

Элизабет обнаружила, что для некоторых  $n$  нужно делать слишком много операций, и не понятно, завершится ли процесс вообще. Поэтому она попросила Вас написать для этого программу.

### Input

В единственной строке дано одно целое число  $n$  ( $2 \leq n \leq 10^6$ ).

### Output

Выведите одно число — количество требуемых шагов. Если процесс будет продолжаться бесконечно — выведите  $-1$ .

### Examples

стандартный ввод	стандартный вывод
4	6
3	24

### Алгоритм

Задача решается с помощью конечного автомата, полученного при исследовании поведения строки при различных входных данных.

## Исходный код

```
1 n = gets.to_i
2 counter = 0
3 state = 'a'
4 a = n
5 while a > 1
6   if state == 'a'
7     if a&1 == 1
8       state = 'c'
9     else
10      state = 'b'
11    end
12    counter += (a.to_f / 2.0).ceil
13  elsif state == 'b'
14    d = a / 2
15    counter += d
16    a -= d
17    state = 'a'
18  else
19    state = 'a'
20    d = (a.to_f / 2.0).ceil
21    counter += d
22    a += d
23  end
24 end
25 puts counter
```

## Задача N - Numbers

На доске в ряд выписаны целые числа от 1 до  $n$ . За одно действие вы можете заменить два произвольных числа  $a$  и  $b$ , стоящих на различных позициях (значения чисел могут совпадать), числом  $|a - b|$ . Можно ли данными операциями сделать так, чтобы на доске осталось единственное целое число  $x$ ?

### Input

Первая строка входных данных содержит число  $T$  — количество тестовых примеров, для которых необходимо решить поставленную задачу ( $1 \leq T \leq 10^5$ ).

В следующих  $T$  строках содержатся по два целых числа  $n_i, x_i$ , ( $1 \leq n_i \leq 10^9$ ,  $0 \leq x_i \leq 10^9$ ).

### Output

Выведите  $T$  строк. В  $i$ -ой строке выведите «YES» (без кавычек), если из чисел  $1, 2, \dots, n_i$  приведёнными в условии задачи действиями можно получить единственное число  $x_i$ , и «NO» (без кавычек) иначе.

### Examples

standard input	standard output
5	YES
3 0	NO
3 1	YES
3 2	NO
3 3	NO
3 5	

### Алгоритм

Задача на реализацию.

### Исходный код

```
1 t = gets.to_i
2 t.times do
3   inp = gets.split(' ').map(&:to_i)
4   n = inp.first
5   x = inp.last
6   first_class = true
7   if n % 4 == 0 or (n + 1) % 4 == 0
8     first_class = false
9   end
10  nums = []
11  if first_class
12    if !x.even? and x <= n
13      puts 'YES'
14    else
15      puts 'NO'
16    end
17  else
18    if x.even? and x <= n
19      puts 'YES'
20    else
21      puts 'NO'
22    end; end; end
```

## Задача O - Ones as the Difference

Задано число  $n$ . Найдите  $n$ -е число, в десятичной записи которого разность двух любых соседних цифр равна единице.

### Input

Первая строка входного файла содержит одно целое число  $n$ , ( $1 \leq n \leq 512$ ).

### Output

В выходной файл выведите одно число — ответ к задаче.

### Examples

standard input	standard output
1	1
2	2
10	10

## Алгоритм

Генерируем все варианты ответов с помощью перебора и вставляем в код конечной программы.

### Исходный код

Генератор:

```
1 n = gets.to_i
2 if n < 10
3   puts n
4 else
5   count = 9; num = 10
6   while true
7     good_num = true; num_as_str = num.to_s
8     for i in 1...(num_as_str.size)
9       digit_l = num_as_str[i - 1].to_i
10      digit_r = num_as_str[i].to_i
11      if (digit_r - digit_l).abs != 1
12        good_num = false; break
13      end
14    end
15    if good_num
16      count = count + 1; print "#{num},"
17    end
18    if count == n
19      puts num; break
20    end
21    num = num + 1
22  end
23 end
```

Программа с предподсчитанными числами:

```
1 n = gets.to_i
2 answer = [1,2,3,4,5,...generated_numbers...,2123456,2123456]
3 puts answer[n - 1]
```

## Задача Р - Places at the Olympics

Для сравнения результатов стран на Олимпийских играх используется два способа подсчёта медалей. Первый — «по количеству» учитывает общее количество медалей без учёта их достоинства. Второй — «по качеству» учитывает сначала количество золотых медалей, при их равенстве — количество серебряных и при их равенстве — количество бронзовых.

Вам задано количество золотых, серебряных и бронзовых медалей, завоёванных соответственно Берляндсией и Байтландсией. Определите, в каком из зачётов Берляндия победила.

### Input

Первая строка входа содержит одно целое число  $T$  ( $1 \leq T \leq 20$ ) — количество тестовых примеров.

Каждый тестовый пример состоит из шести целых чисел от 0 до 500 включительно. Первые три числа задают количество золотых, серебряных и бронзовых медалей соответственно, завоёванных Берляндсией, следующие три — количество медалей, завоёванных Байтландсией (в аналогичном формате).

### Output

:

Для каждого тестового примера выведите “Win”, если Берляндия победила в обоих зачётах, “Count”, если она выиграла только по количеству, “Quality”, если выиграла только по качеству и “None” во всех оставшихся случаях.

### Example

standard input	standard output
5	Win
20 10 30 20 2 0	Count
15 10 20 15 11 15	Quality
13 6 11 6 21 31	None
14 4 15 14 9 30	None
1 2 3 1 2 3	

## Алгоритм

Задача на реализацию.

## Исходный код

```
1 t = gets.to_i
2
3 t.times do
4   medals = gets.split(',').map(&:to_i)
5   temp = medals.each_slice(3).to_a
6   berland = temp.first
7   byteland = temp.last
8   # test Win
9   berland_count = berland.inject(:+)
10  byteland_count = byteland.inject(:+)
11  count_win = false
12  if (berland_count > byteland_count)
13    count_win = true
14  end
15  # test Quality
16  quality_win = false
17  if berland[0] > byteland[0]
18    quality_win = true
19  elsif berland[0] == byteland[0] and
20    berland[1] > byteland[1]
21    quality_win = true
22  elsif berland[0] == byteland[0] and
23    berland[1] == byteland[1] and
24    berland[2] > byteland[2]
25    quality_win = true
26  end
27  if count_win and quality_win
28    puts "Win"
29  elsif count_win
30    puts "Count"
31  elsif quality_win
32    puts "Quality"
33  else
34    puts "None"
35  end
36 end
```

## Задача Q - Quaqua the Frog

Лягушка Кваква решила перебраться в новое болото. Кваква может прыгать на целое число клеток, перепрыгивая при этом не более чем через  $j$  клеток. Оба болота расположены на полосе  $1 \times c$  клеток. Старое болото находится в первой клетке полосы, а новое — в последней. Некоторые клетки полосы содержат ямы с кислотой, попасть в которые для лягушек небезопасно. Так что Кваква собирается перепрыгивать эти ямы. Дополнительная проблема заключается в том, что сегодня дует сильный ветер в направлении от старого болота к новому, и Кваква не может прыгать против ветра, то есть прыгать лягушка может только в направлении от старого болота к новому.

Выясните, сможет ли Кваква сегодня добраться до нового болота и, если сможет, выведите минимальное количество прыжков, которое ей для этого необходимо.

### Input

Первая строка входа содержит одно целое число  $T$  ( $1 \leq T \leq 35$ ) — количество тестовых примеров.

Далее следуют  $n$  тестовых примеров. Каждый тестовый пример состоит из двух строк. Первая строка содержит два целых числа  $c$  ( $2 \leq c \leq 50$ ) и  $j$  ( $0 \leq j \leq 50$ ) — количество клеток на пути между старым и новым болотами (включая оба болота) и максимальное количество клеток, через которое может перепрыгнуть Кваква, соответственно. Вторая строка содержит  $c$  символов,  $i$ -й из которых равен 'A', если эта клетка содержит яму с кислотой, и '.', если соответствующая клетка безопасна. Гарантируется, что первый и последний символы строки равны '..'.

### Output

Для каждого тестового примера выведите одно целое число — наименьшее количество прыжков, необходимое для того, чтобы Кваква добралась от старого болота до нового. Если же это сделать невозможно, выведите  $-1$ .

### Example

standard input	standard output
4	2
8 3	1
.AA.A.A.	-1
3 50	3
...	
8 1	
...AA...	
10 4	
..AAAA.AA.	

### Алгоритм

Сложность  $O(n^2)$ .

## Исходный код

```
1 import java.util.*;
2 public class Test {
3     public static void main(String[] args) {
4         Scanner sc = new Scanner(System.in);
5         int T = sc.nextInt();
6         int c;
7         int j;
8         String s;
9         for(int i = 0; i < T; i++) {
10            c = sc.nextInt();
11            j = sc.nextInt();
12            char arr[];
13            sc.nextLine();
14            s = sc.nextLine();
15            arr = s.toCharArray();
16            boolean frogCan = false;
17            int count = 0;
18            int cur = 0;
19            while(true) {
20                if(cur + j + 1 < c - 1) {
21                    if(arr[cur+j + 1] == '.') {
22                        cur += j + 1;
23                        count++;
24                        frogCan = true;
25                    } else {
26                        for(int jTemp = j - 1; jTemp >= 0; jTemp--) {
27                            if(arr[cur + jTemp + 1] == '.') {
28                                cur += jTemp + 1;
29                                count++;
30                                frogCan = true;
31                                break;
32                            }
33                        }
34                    }
35                    if(!frogCan) {
36                        break;
37                    }
38                    frogCan = false;
39                } else {
40                    count++;
41                    frogCan = true;
42                    break;
43                }
44            }
45            if(!frogCan) {
46                System.out.println("-1");
47            } else {
48                System.out.println(count);
49            }
50        }
51    }
52 }
```

## Задача R - Road in Mountains

Извилистые горные дороги, иначе называемые серпантинами, могут быть представлены как линии, составленные из дуг окружностей. Большая Берляндская Горная дорога составлена из  $N$  полуокружностей.  $i$ -й участок имеет радиус  $r_i$  и рекомендованная скорость движения на этом участке равна  $v_i$ . Считается, что скорость на каждом участке постоянна и в момент перемещения к следующему участку она мгновенно изменяется на  $v_{i+1}$ .

Берляндское агентство путешествий поручило Вам вычислить среднюю скорость на Большой Берляндской дороге при движении с рекомендованной скоростью на каждом её участке.

### Input

В первой строке входного файла содержится одно число  $N$  — количество участков дороги.  $1 \leq N \leq 300$ . Каждая из  $N$  следующих строк содержит два числа:  $r_i$ ,  $v_i$  — радиус соответствующего участка дороги и рекомендованную скорость ( $0 \leq r_i \leq 10^6$ ,  $0 < v_i \leq 10^6$ ).

### Output

Выведите среднее значение скорости на всём пути следования с абсолютной или относительной погрешностью не хуже  $10^{-6}$ .

### Examples

standard input	standard output
2 1 1 1 1	1.0

### Алгоритм

Задача на реализацию. Сложность  $O(n)$ .

### Исходный код

```
1 public class Test {  
2     public static void main(String [] args) {  
3         Scanner sc = new Scanner(System.in);  
4         int N = sc.nextInt();  
5         double temp; double S = 0; double t = 0;  
6         double arr [][] = new double[N][2];  
7         for(int i = 0; i < N; i++) {  
8             temp = sc.nextDouble();  
9             arr [i][0] = temp;  
10            S += arr [i][0];  
11            temp = sc.nextDouble();  
12            arr [i][1] = temp;  
13            t += arr [i][0] / arr [i][1];  
14        }  
15        System.out.println(S/t);  
16    }
```

## 2.14 OpenCup GrandPrix of Tatarstan Div 2

### Результаты

60.	MAI #6: Makarov, Rik, Yakimenko	-	-	-	-	-	+1 1:02	-8 1:04	-1 1:51	-6 3:47	-3 2:30	-	+3 1:16	2	218	66%	0.03
-----	------------------------------------	---	---	---	---	---	------------	------------	------------	------------	------------	---	------------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10343](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10343)

## Задача G - Кастрюли

В магазине кастрюль резко упали продажи. Маркетологи магазина провели исследования и поняли, что причина в сковородках. Люди перестали покупать кастрюли, так как сковородки и дешевле, и компактнее при хранении. Совет директоров магазина принял решение расширить ассортимент и начать продажи сковородок. Первая партия уже заказана.

Отделу складской логистики поручено подготовить место под новый товар. Сейчас на складе размещены  $N$  кастрюль, для каждой из которых известен диаметр  $D_i$ . Существует единственный способ экономии пространства — можно в любую кастрюлю вложить ровно одну кастрюлю меньшего диаметра, в которую могут быть вложены другие.

Помогите логисту найти минимальное количество кастрюль на складе, в которые можно будет вложить все остальные.

### Input

В первой строке записано единственное число  $N$  ( $1 \leq N \leq 1000$ ). Во второй строке —  $N$  целых чисел  $D_i$ , разделенных пробелом ( $1 \leq D_i \leq 10\,000$ ).

### Output

Выведите искомое число.

### Examples

стандартный ввод	стандартный вывод
5 7 5 2 5 2	2

### Алгоритм

Подсчитываем количество каждого вида кастрюль и выбираем наибольшее. Сложность  $O(n)$ .

### Исходный код

```
1 n = gets.to_i
2 data = gets.split(' ').map(&:to_i)
3
4 h = Hash.new
5 data.each do |item|
6   el = h[item]
7   if el == nil
8     h[item] = 1
9   else
10    h[item] = el + 1
11  end
12 end
13
14 top = 0
15 h.each_value do |val|
16   if val > top
17     top = val
18   end
19 end
20 puts top
```

## Задача М - Наименьшая дробь

В настоящее время миру известно огромное количество сенсационных археологических находок. Во время последних раскопок были обнаружены носители информации, относящиеся к эпохе программирования XX века. Расшифровка файлов позволила ученым доказать гипотезу о том, что древние программисты обладали искусством производить простейшие арифметические операции с дробями. Многие тексты были расшифрованы, многие загадки были разгаданы. Однако среди нерешенных оказалась задача о вычислении наименьшей положительной дроби, при делении которой на каждую из  $n$  заданных дробей получаются целые числа.

Возможно, это удастся сделать вам...

### Input

В первой строке записано одно целое число  $n$  — количество заданных дробей ( $1 \leq n \leq 6$ ). В каждой из следующих  $n$  строк записано два целых числа  $a_i, b_i$  — числитель и знаменатель несократимой дроби ( $1 \leq a_i \leq 10^3, 1 \leq b_i \leq 10^9$ ).

### Output

В единственной строке запишите через пробел два положительных целых числа — числитель и знаменатель наименьшей несократимой дроби, удовлетворяющей условию задачи.

### Examples

стандартный ввод	стандартный вывод
2 1 2 3 4	3 2
2 2 3 4 5	4 1

### Алгоритм

Задача решается с помощью итеративного вычисления наибольшего общего делителя и наименьшего общего множителя. Сложность  $O(n)$ .

## Исходный код

```
1 import java.util.Scanner;
2 public class OpenCup {
3     public static long gcd (long a, long b) {
4         if (b == 0)
5             return a;
6         else
7             return gcd (b, a % b);
8     }
9     public static long lcm (long a, long b) {
10        return a / gcd (a, b) * b;
11    }
12    public static void main(String [] args) {
13        Scanner in = new Scanner(System.in);
14        int n = in.nextInt();
15        long [] a = new long[n];
16        long [] b = new long[n];
17        for(int i = 0; i < n; i++) {
18            a[i] = in.nextLong();
19            b[i] = in.nextLong();
20        }
21        long A = a[0], B = b[0];
22        for(int i = 0; i < n - 1; i++) {
23            b[i + 1] = gcd(b[i], b[i + 1]);
24            a[i + 1] = lcm(a[i], a[i + 1]);
25        }
26        A = a[n - 1];
27        B = b[n - 1];
28        long t = gcd(A, B);
29        A = A / t;
30        B = B / t;
31        System.out.println(A + " " + B);
32    }
33 }
```

## 2.15 OpenCup GrandPrix of Baltics Div 2

### Результаты

19.	MAI #6: Makarov, Rik, Yakimenko	-	-	-	-	+1 2:20	-1 0:39	+2 1:15	-4 4:42	-4 4:57	+4 1:49	3	466	70%	0.36
-----	---------------------------------	---	---	---	---	------------	------------	------------	------------	------------	------------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10344](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10344)

## Задача Е - Поиск уникального элемента

Это интерактивная задача

А вы слышали о такой задаче: «Дан массив чисел, в котором все числа кроме одного, встречаются два раза, а оставшееся — один раз. Найти это число.» ?

Вам предстоит решить почти такую же. В этой задаче массив чисел отсортирован, содержит все элементы кроме одного по два раза, оставшийся элемент содержит один раз, но он вам не дан! Вместо этого, можно по одному запрашивать его элементы.

### Протокол взаимодействия

В начале взаимодействия на вход вашей программе будет подано нечетное число  $n$  ( $1 \leq n \leq 199\,999$ ) — длина массива.

После этого вы можете делать два типа запросов.

- «?  $x$ ». Запросить элемент массива на позиции  $x$  ( $1 \leq x \leq n$ ). Разрешено сделать не более 40 таких запросов. При превышении этого лимита решение получит вердикт «Wrong Answer».
- «!  $v$ ». Ответить на задачу. Число  $v$  должно быть равно единственному элементу массива, который встречается один раз.

В ответ на запрос первого типа будет получена одна строка, в которой содержится соответствующий элемент массива. Это положительное целое число не превосходящее  $10^9$ .

После выполнения запроса второго типа решение должно корректно завершиться.

Каждый запрос следует выводить на отдельной строке. Чтобы предотвратить буферизацию вывода, после каждого выведенного запроса следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

### Пример

запросы участника	ответы проверяющей программы	Загаданный массив
? 1	5	1 1 2 3 3
? 5	1	
? 3	3	
! 2	2	

### Алгоритм

Задача решается с помощью бинарного поиска. Сложность  $O(\log_2(n))$ .

## Исходный код

```
1 n = gets.to_i
2 if n == 1
3   puts "? 1"
4   STDOUT.flush
5   puts "! #{gets}"
6   STDOUT.flush
7   exit
8 end
9 idx1 = 1
10 idx2 = 2
11 l = 1
12 r = n
13 a = 0
14 b = 0
15 while true
16   idx1 = (l + r) / 2
17   if idx1.even?
18     idx1 = idx1 - 1
19   end
20   idx2 = idx1 + 1
21   if idx2 > n or l == r
22     puts "? #{idx1}"
23     STDOUT.flush
24     a = gets.to_i
25     puts "! #{a}"
26     STDOUT.flush
27     exit
28   end
29   puts "? #{idx1}"
30   STDOUT.flush
31   a = gets.to_i
32   puts "? #{idx2}"
33   STDOUT.flush
34   b = gets.to_i
35   if a != b
36     r = idx1
37   else
38     l = idx2 + 1
39   end
40 end
```

## Задача G - Перевёрнутая цепная дробь

Конечная цепная дробь — это последовательность вида  $[a_0; a_1, a_2, \dots, a_n]$ . На элементы цепной дроби и её размер накладываются следующие ограничения:

- $n$  — неотрицательное конечное целое число,
- элементы  $a_0, a_1, a_2, \dots, a_n$  — целые числа,
- $a_i > 0$  при  $i > 0$ ,
- $a_n > 1$ , если  $n > 0$ .

Эти ограничения позволяют установить взаимно однозначное соответствие между рациональными числами и конечными цепными дробями: каждому рациональному числу  $x$  соответствует единственная цепная дробь  $[a_0; a_1, a_2, \dots, a_n]$  такая, что

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}}.$$

Для обозначения соответствия используется знак равенства:  $x = [a_0; a_1, a_2, \dots, a_n]$ . Например,

$$\frac{17}{25} = 0 + \cfrac{1}{\frac{25}{17}} = 0 + \cfrac{1}{1 + \cfrac{8}{17}} = 0 + \cfrac{1}{1 + \cfrac{1}{\frac{17}{8}}} = 0 + \cfrac{1}{1 + \cfrac{1}{2 + \frac{1}{8}}},$$

поэтому мы пишем  $\frac{17}{25} = [0; 1, 2, 8]$ .

Вам дано рациональное число  $x$  ( $0 < x \leq \frac{1}{2}$ ). Пусть  $x = [0; a_1, a_2, \dots, a_n]$ . Найдите рациональное число, равное  $[0; a_n, a_{n-1}, \dots, a_1]$ .

### Формат входных данных

В единственной строке даны два положительных целых числа  $p$  и  $q$  ( $1 \leq p < q \leq 10^9$ ). Гарантируется, что  $\frac{p}{q}$  — несократимая дробь, причём  $0 < \frac{p}{q} \leq \frac{1}{2}$ .

### Формат выходных данных

Выведите искомое число в виде несократимой дроби. Числитель и знаменатель дроби следует выводить через пробел.

### Примеры

reversed-fraction.in	reversed-fraction.out
3 7	2 7
2 7	3 7
2 5	2 5

### Алгоритм

Задача на реализацию.

## Исходный код

```
1 n, d = gets.split(' ').map(&:to_i)
2 if n == 1
3   puts "#{n} #{d}"
4   exit
5 end
6 nn = n
7 dd = d
8 arr = []
9 while true
10   nn, dd = dd, nn
11   val = nn / dd
12   arr.push(val)
13   nn = nn - dd * val
14   if nn == 1
15     arr.push(dd)
16     break
17   end
18 end
19 nn = 1
20 dd = arr.first
21 for i in (1...arr.size)
22   nn = arr[i] * dd + nn
23   nn, dd = dd, nn
24 end
25 puts "#{nn} #{dd}"
```

## Задача J - Нарисуй прямой обход

Из википедии мы знаем:

*Двоичное дерево поиска (англ. binary search tree, BST) – это двоичное дерево, для которого выполняются следующие дополнительные условия (свойства дерева поиска):*

- Оба поддерева – левое и правое – являются двоичными деревьями поиска.
- У всех узлов левого поддерева произвольного узла  $x$  значения ключей данных меньше, нежели значение ключа данных самого узла  $x$ .
- В то время, как значения ключей данных у всех узлов правого поддерева (того же узла  $x$ ) больше, нежели значение ключа данных узла  $x$ .

Рассмотрим следующую процедуру вывода BST:

```
void outTree( Node* v ) {  
    if (v == 0) return;  
    cout << v->x << " ";  
    outTree(v->left);  
    outTree(v->right);  
}  
outTree(root);
```

Вам дан вывод непустого дерева, сделанный этой функцией. Известно, что по этому выводу можно однозначно восстановить структуру исходного дерева. Преобразуйте его в красивую 2D картинку.

Обозначим 2D-изображение дерева с корнем в  $v$  как  $\text{rect}(v)$ .  $\text{rect}(v)$  – прямоугольник, получаемый вызовом рекурсивной функции. Возьмём  $\text{rect}(v.left)$ ,  $\text{rect}(v.right)$  и строку  $s$ , задающую  $v.x$  ( $s$  тоже прямоугольник, её высота 1, а длина равна длине десятичного представления числа  $v.x$ ). Нарисуем слева направо  $\text{rect}(v.left)$ ,  $s$ ,  $\text{rect}(v.right)$  таким образом, что правый верхний угол  $\text{rect}(v.left)$  совпадает с левым нижним углом  $s$  и верхний левый угол  $\text{rect}(v.right)$  совпадает с правым нижним углом  $s$ . Прямоугольник, ограничивающий полученное изображение, и есть  $\text{rect}(v)$ .

### Формат входных данных

В единственной строке вывод функции `outTree`. Ключи дерева – целые числа от 0 до  $10^9$ . Все ключи различны.

### Формат выходных данных

Выведите 2D-картинку. Гарантируется, что размер вывода не превосходит 5 мегабайт. Строки не должны содержать хвостовые пробелы. Каждая строка должна оканчиваться переводом строки. В примере пробелы сделаны видимыми, но на самом деле следует выводить обычные пробелы с ASCII-кодом 32.

### Пример

xlrtod.in	xlrtod.out
2 1 400 3 500	2 1 400 3 500

### Алгоритм

Восстанавливаем структуру двоичного дерева и затем просто обходим дерево слева направо в порядке возрастания чисел и при этом в выводе отражаем глубину дерева.

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 struct Tree{
6     long val;
7     Tree *left;
8     Tree *right;
9     int depth;
10    Tree () {
11        val = -1;
12    }
13 };
14
15 struct Line_part {
16     long val;
17     bool isVal;
18     Line_part(long a, bool b) {
19         val = a;
20         isVal = b;
21     }
22 };
23 void tree_add(Tree *nd, long val, int depth) {
24     if (nd->val == -1) {
25         nd->val = val;
26         nd->left = new Tree();
27         nd->right = new Tree();
28         nd->depth = depth;
29     }
30     else if (nd->val > val) {
31         tree_add(nd->left, val, nd->depth + 1);
32     }
33     else {
34         tree_add(nd->right, val, nd->depth + 1);
35     }
36 }
37
38 struct Line {
39     int length;
40     vector<Line_part> parts;
41     Line() {
42         length = 0;
43     }
44 };
45 vector<Line> rect;
46 long n = 0;
47 void tree_print (Tree *nd) {
48     if (nd->val == -1)
49         return;
50     tree_print(nd->left);
51     int c = 1;
52     long t = nd->val;
53     while (t >= 10) {
54         t /= 10;
55         ++c;
56     }
57     if (rect.size() <= nd->depth) {
58         rect.resize((nd->depth + 1) * 2, Line());
```

```

59     }
60     Line *line = &rect[nd->depth];
61     line->parts.push_back(Line_part(n - line->length, false));
62     line->parts.push_back(Line_part(nd->val, true));
63     line->length += c + n - line->length;
64     n += c;
65     tree_print(nd->right);
66 }
67
68 int main() {
69     long d;
70     Tree *root = new Tree();
71     while (1) {
72         cin >> d;
73         tree_add(root, d, 0);
74         if (cin.get() != ',')
75             break;
76     }
77     tree_print(root);
78     for (int i = 0; i < rect.size(); ++i) {
79         Line *line = &rect[i];
80         if (line->length == 0)
81             break;
82         for (int j = 0; j < line->parts.size(); ++j) {
83             if (line->parts[j].isVal) {
84                 cout << line->parts[j].val;
85             }
86             else {
87                 for (int k = 0; k < line->parts[j].val; ++k)
88                     cout << ',';
89             }
90         }
91         cout << endl;
92     }
93     return 0;
94 }
```

## 2.16 OpenCup GrandPrix of Moscow Div 2

### Результаты

30.	MAI #6: Makarov, Rik, Yakimenko	-1 0:36	-	-	-	-	-	-3 4:55	-	+5 2:41	-1 3:46	+	2 0:12	2	273	71%	0.17
-----	---------------------------------	------------	---	---	---	---	---	------------	---	------------	------------	---	-----------	---	-----	-----	------

Ссылка на контест: [http://opentrains.snarknews.info/~ejudge/team.cgi?contest\\_id=10345](http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10345)

## Задача N - Notebooks

Ценность ноутбука для потребителя, согласно исследованию маркетинговой компании «BadBook», определяется двумя параметрами — продуктивностью и компактностью.

Некоторые предпочтут большой и тяжёлый ноутбук с высокой производительностью, некоторые — лёгкий и тонкий ноутбук, даже если его производительность невысока.

Компания «Badbook» составила список ноутбуков, сопроводив каждую модель двумя числами от 0.0 до 10.0.

Первое число обозначает оценку производительности (чем выше, тем лучше для потребителя). Второе — оценку размера (чем ниже, тем лучше для потребителя).

По заданному списку ноутбуков найдите длину наибольшей подпоследовательности ноутбуков, каждый из которых строго лучше для потребителя по каждому из параметров, чем предыдущий.

### Input

Первая строка входа содержит одно целое число  $T$  ( $1 \leq T \leq 20$ ) — количество тестовых примеров.

Каждый тестовый пример начинается строкой, содержащей одно целое число  $N$  ( $1 \leq N \leq 200$ ) — количество ноутбуков в списке. Далее следуют  $N$  строк, каждая из которых содержит два вещественных числа  $p_i$  и  $w_i$  ( $0 \leq p_i, w_i \leq 10.0$ ) — оценку производительности и размера очередного ноутбука.

### Output

Для каждого тестового примера выведите строку, содержащую длину наибольшей подпоследовательности ноутбуков, каждый из которых строго превосходит предыдущий по обоим параметрам.

### Example

standard input	standard output
2	2
2	1
1.0 2.0	
2.4 0.0	
3	
3.1 3.1	
3.2 3.1	
3.3 3.1	

### Алгоритм

Задача решается перебором. Сложность  $O(n^2)$ .

## Исходный код

```
1 t = gets.to_i
2 t.times do
3     n = gets.to_i
4     data = Array.new(n)
5     for i in 0...n
6         data[i] = gets.split(' ').map(&:to_f)
7     end
8     d = Array.new(10)
9     for i in 0...n
10        d[i] = 1;
11        for j in 0...i
12            if data[j].first < data[i].first and data[j].last > data[i].last
13                d[i] = [d[i], 1 + d[j]].max
14            end
15        end
16    end
17    ans = d[0]
18    for i in 0...n
19        ans = [ans, d[i]].max
20    end
21    puts ans
22 end
```

## Задача P - Convex Polyhedron

Для вписанного в сферу многогранника соблюдается следующее соотношение: значение  $x = V - E + F$ , где  $V$  задаёт количество вершин,  $E$  — количество рёбер и  $F$  — количество граней, постоянно.

По заданным  $V$  и  $E$  требуется найти  $F$ .

### Input

Первая строка входа содержит целое число  $T$  ( $1 \leq T \leq 100$ ) — количество тестовых примеров.

Каждый тестовый пример состоит из одной строки, содержащей два целых числа  $V$  и  $E$  ( $4 \leq V, E \leq 100$ ) — количества вершин и количества рёбер, соответственно.

### Output

Для каждого тестового примера в отдельной строке выведите ответ — количество граней.

### Example

standard input	standard output
1	4
4 6	

### Алгоритм

Задача решается по формуле  $x = 2 - V + E$ . Сложность  $O(1)$ .

### Исходный код

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4 import java.util.Scanner;
5 public class C {
6     public static void main(String [] args) {
7         Scanner sc = new Scanner(System.in);
8         int T = sc.nextInt();
9         for(int i = 0; i < T; i++) {
10             int V = sc.nextInt();
11             int E = sc.nextInt();
12             System.out.println(2 - V + E);
13         }
14     }
15 }
```

## **2.17 Vekua Cup 2016**

Так как соревнование проводилось в МФТИ, то турнирная таблица с результатами и исходные коды программ не доступны.

### 3 Журнал по личным контестам Макарова Н.А.

#### 3.1 Codeforces Round 320 Div 2

##### Результаты

Задачи		Название		
№				
A	<a href="#">Выращиваем бактерии</a>	стандартный ввод/вывод 1 с, 256 МБ		x7116
B	<a href="#">Ищем напарника</a>	стандартный ввод/вывод 2 с, 256 МБ		x3630
C	<a href="#">Задача про ломаную</a>	стандартный ввод/вывод 1 с, 256 МБ		x1525
D	<a href="#">Игра</a>	стандартный ввод/вывод 2 с, 256 МБ		x1152
E	<a href="#">Слабость и бедность</a>	стандартный ввод/вывод 2 с, 256 МБ		x364
F	<a href="#">И снова НОП!</a>	стандартный ввод/вывод 2 с, 256 МБ		x24

Ссылка на контест: <http://codeforces.com/contest/579>

## Задача А - Выращиваем бактерии

Вы — большой любитель бактерий. Вам хочется вырастить немного бактерий в коробочке.

Изначально коробочка пуста. Каждое утро можно положить любое количество бактерий в коробочку. Каждую ночь каждая бактерия делится на две бактерии. Когда-нибудь вы надеетесь увидеть ровно  $x$  бактерий в коробочке.

Какое минимальное количество бактерий вам суммарно надо положить в коробочку для достижения этой цели?

### Входные данные

В единственной строке записано одно целое число  $x$  ( $1 \leq x \leq 10^9$ ) — количество бактерий.

### Выходные данные

Единственная строка, содержащая одно целое число — ответ на задачу.

#### Примеры

входные данные
5
выходные данные
2

## Алгоритм

В этой задаче нужно вычислить, сколько битов входного числа  $n$  являются ненулевыми. Это и будет ответом. Сложность решения  $O(n)$ .

## Исходный код

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 int main(int argc, const char * argv[]) {
5     long n = 0;
6     scanf("%ld", &n);
7     int i, count = 0;
8     for (i = 0; i < 32; i++) {
9         if (n & (1L << i)) {
10             count++;
11         }
12     }
13     printf("%d", count);
14     return 0;
15 }
```

## Задача В - Ищем напарника

Есть соревнование по программированию под названием SnakeUp, в котором хотят принять участие  $2n$  человек. Принять участие в контесте можно только в составе команды из ровно двух людей. Известна сила каждой возможной команды из двух людей. Все значения сил **различны**.

Каждый соревнующийся надеется, что он может найти напарника, с которым он образует как можно более сильную команду. Таким образом, соревнующийся стремится создать команду с наибольшей возможной силой, путем выбора сокомандника из тех, кто согласен быть с ним в команде. Более формально, два человека,  $A$  и  $B$  смогут создать команду, если каждый из них является наилучшим возможным напарником (среди соревнующихся, не нашедших к текущему моменту пару) друг для друга.

Определить, кто с кем будет участвовать в команде?

### Входные данные

Ввод состоит из  $2n$  строк.

В первой строке записано целое число  $n$  ( $1 \leq n \leq 400$ ) – количество команд, которые надо образовать.

В  $i$ -й строке ( $i > 1$ ) записано  $i - 1$  чисел  $a_{i1}, a_{i2}, \dots, a_{i(i-1)}$ . Здесь  $a_{ij}$  ( $1 \leq a_{ij} \leq 10^6$ , все  $a_{ij}$  различны) обозначает силу команды, состоящей из человека  $i$  и человека  $j$  (люди пронумерованы, начиная с 1)

### Выходные данные

Выведите строку, содержащую  $2n$  чисел. Из них  $i$ -е число должно обозначать номер напарника  $i$ -го человека.

### Примеры

#### входные данные

```
2  
6  
1 2  
3 4 5
```

#### выходные данные

```
2 1 4 3
```

#### входные данные

```
3  
487060  
3831 161856  
845957 794650 976977  
83847 50566 691286 498447  
698377 156232 59015 382455 626960
```

#### выходные данные

```
6 5 4 3 2 1
```

## Алгоритм

Для решения этой задачи будем использовать очередь с приоритетами. При считывании будем помечать каждого участника как не состоящего в команде, а команду вставлять в очередь с приоритетами по силе. Затем будем доставать команды из очереди и если оба участника еще не состоят в команде, создавать из них команду. Задача решается за  $O(n * \log(n))$ .

## Исходный код

```
1 #include <iostream>
2 #include <queue>
3
4 using namespace std;
5
6 typedef struct {
7     long player1;
8     long player2;
9     long power;
10 } Team;
11
12 class CompareTeams {
13 public:
14     bool operator() (Team a, Team b) {
15         return a.power < b.power;
16     }
17 };
18
19 int main(int argc, const char * argv[]) {
20
21     priority_queue<Team, vector<Team>, CompareTeams> teams;
22     int n;
23     cin >> n;
24     for (int i = 0; i < (2 * n); i++) {
25         for (int j = 0; j < i; j++) {
26             long power;
27             cin >> power;
28             Team team;
29             team.player1 = i + 1;
30             team.player2 = j + 1;
31             team.power = power;
32             teams.push(team);
33         }
34     }
35
36     vector<long> res(2 * n, -1);
37
38     int teamCount = 0;
39
40     while (teamCount != n) {
41         Team team = teams.top();
42         teams.pop();
43         if (res[team.player1 - 1] == -1 && res[team.player2 - 1] == -1) {
44             res[team.player1 - 1] = team.player2;
45             res[team.player2 - 1] = team.player1;
46             teamCount++;
47         }
48     }
49
50     for (int i = 0; i < 2 * n; i++) {
51         cout << res[i] << " ";
52     }
53
54     cout << endl;
55
56     return 0;
57 }
58 }
```

### 3.2 Codeforces Round 323 Div 2

#### Результаты

Задачи		Nº	Название			
A	<a href="#">Асфальтирование дорог</a>		стандартный ввод/вывод 1 с, 256 МБ			x6858
B	<a href="#">Задача робота</a>		стандартный ввод/вывод 1 с, 256 МБ			x5490
C	<a href="#">Таблица НОД</a>		стандартный ввод/вывод 2 с, 256 МБ			x2393
D	<a href="#">И снова ...</a>		стандартный ввод/вывод 1 с, 256 МБ			x847
E	<a href="#">Превосходящие периодические подмассивы</a>		стандартный ввод/вывод 1 с, 256 МБ			x74

Ссылка на контест: <http://codeforces.com/contest/583>

## Задача А - Асфальтирование дорог

Вы — большой любитель бактерий. Вам хочется вырастить немного бактерий в коробочке.

Изначально коробочка пуста. Каждое утро можно положить любое количество бактерий в коробочку. Каждую ночь каждая бактерия делится на две бактерии. Когда-нибудь вы надеетесь увидеть ровно  $x$  бактерий в коробочке.

Какое минимальное количество бактерий вам суммарно надо положить в коробочку для достижения этой цели?

### Входные данные

В единственной строке записано одно целое число  $x$  ( $1 \leq x \leq 10^9$ ) — количество бактерий.

### Выходные данные

Единственная строка, содержащая одно целое число — ответ на задачу.

#### Примеры

входные данные
5
выходные данные
2

## Алгоритм

В этой задаче нужно заметить, что так как изначально не заасфальтирована ни одна дорога, то бригада сможет асфальтировать только на тех перекрестках, чьи оба индекса совпадают. Поэтому в ответ пойдет номер дня, когда индексы будут одинаковые. Решается за считывание.

## Исходный код

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main(int argc, const char * argv[]) {
6
7     int n;
8     cin >> n;
9     int a, b;
10
11    for (int i = 0; i < n * n; i++) {
12        cin >> a >> b;
13        if (a == b) {
14            cout << i + 1 << " ";
15        }
16    }
17
18    return 0;
19 }
```

## Задача В - Задача робота

Робот Док находится в зале, в котором в ряд стоят  $n$  компьютеров, пронумерованных слева направо от 1 до  $n$ . В каждом компьютере содержится **ровно одна** часть информации, каждую из которых Док хочет в итоге получить. Компьютеры оснащены системой защиты, поэтому, чтобы взломать  $i$ -й из них, роботу нужно собрать не менее  $a_i$  любых частей информации из других компьютеров. Осуществить взлом Док может, только находясь непосредственно рядом с компьютером.

Робот собран по современным технологиям и может двигаться вдоль ряда компьютеров в любом из двух возможных направлений, однако смена направления движения требует большое количество ресурсов Дока. Сообщите минимальное количество смен направлений движения, которое придется сделать роботу для сбора всех  $n$  частей информации, если изначально он находится рядом с компьютером с номером 1.

**Гарантируется**, что существует хотя бы одна последовательность действий робота, приводящая к сбору всей информации. Изначально Док не владеет ни одной из частей информации.

### Входные данные

В первой строке содержится число  $n$  ( $1 \leq n \leq 1000$ ). Во второй строке содержатся  $n$  целых неотрицательных чисел  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i < n$ ), разделенных пробелом. Гарантируется, что робот может собрать все части информации.

### Выходные данные

Выведите единственное число — минимальное количество смен направлений движения, которое придется сделать роботу для сбора всех  $n$  частей информации.

### Примеры

<b>входные данные</b>
3
0 2 0
<b>выходные данные</b>
1

<b>входные данные</b>
5
4 2 3 0 1
<b>выходные данные</b>
3

<b>входные данные</b>
7
0 3 1 0 5 2 6
<b>выходные данные</b>
2

## Алгоритм

Задача на реализацию. Нужно симулировать движение робота. Решается за  $O(n^2)$ .

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 int main(int argc, const char * argv[])
7 {
8     int n;
9     cin >> n;
10
11    vector<pair<int, bool>> v(n);
12    for (int i = 0; i < n; i++) {
13        pair<int, bool> temp;
14        cin >> temp.first;
15        temp.second = false;
16        v[i] = temp;
17    }
18
19    int data = 0;
20    int switch_count = 0;
21    int i = 0;
22    bool to_right = true;
23
24    while (data != n) {
25        if (!v[i].second && data >= v[i].first) {
26            data++;
27            v[i].second = true;
28        }
29
30        to_right ? i++ : i--;
31
32        if (i == n) {
33            to_right = false;
34            i = n - 1;
35            if (data != n) {
36                switch_count++;
37            }
38        } else if (i == -1) {
39            to_right = true;
40            i = 0;
41            if (data != n) {
42                switch_count++;
43            }
44        }
45    }
46
47    cout << switch_count << endl;
48
49    return 0;
50 }
```

### 3.3 Codeforces Round 324 Div 2

#### Результаты

Задачи			
№	Название		
A	<a href="#">Олеся и Родион</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x7935</a>
B	<a href="#">Коля и Таня</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x4512</a>
C	<a href="#">Марина и Вася</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x3029</a>
D	<a href="#">Дима и Лиза</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x2988</a>
E	<a href="#">Антон и Ира</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x759</a>

Ссылка на контест: <http://codeforces.com/gym/584>

## Задача А - Олеся и Родион

Олеся любит числа длины  $n$ , а Родиону нравятся только числа, делящиеся на  $t$ . Найдите какое-нибудь число, устраивающее их обоих.

Ваша задача — по заданным  $n$  и  $t$  вывести целое число строго больше нуля, состоящие из  $n$  цифр, которое делится на  $t$ . Если же такого числа не существует, выведите - 1.

### Входные данные

В единственной строке записаны два числа  $n$  и  $t$  ( $1 \leq n \leq 100$ ,  $2 \leq t \leq 10$ ) — длина числа и на что оно должно делиться.

### Выходные данные

Выполните одно целое положительное число без ведущих нулей, являющееся ответом на задачу, или - 1, если такого числа не существует. Если возможных ответов несколько, разрешается вывести любой.

### Примеры

входные данные
3 2
выходные данные
712

## Алгоритм

В этой задаче нужно построить строку представляющую число, которое делится на  $t$  и имеет длину  $n$  знаков. Сначала нужно определить голову числа, это будет число, которое делится на  $n$ . Затем мы делаем из него число кратное 10 путем добавления оставшихся нулей. Если это количество нулей отрицательное, значит число нельзя построить и ответ будет -1. Сложность -  $O(n)$ .

## Исходный код

```
1 input = gets.split(" ")
2 n = input[0].to_i
3 t = input[1].to_i
4 rest = n
5 answer = ""
6 answer += t.to_s
7 if t == 10
8   rest -= 2
9 else
10   rest -= 1
11 end
12 if rest < 0
13   puts "-1"
14 else
15   if rest == 0
16     puts answer
17   else
18     for i in 0...rest
19       answer += "0"
20     end
21   puts answer
22 end
23 end
```

### 3.4 Codeforces Beta Round 40 Div 2

#### Результаты

Задачи		Название		
№				
A	<a href="#">Перевод<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x10168
B	<a href="#">Марсианский доллар<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x1180
C	<a href="#">Email-адрес<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x1374
D	<a href="#">Пешка<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x527
E	<a href="#">З-циклы<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x661

Ссылка на контест: <http://codeforces.com/contest/41>

## Задача А - Перевод

Перевод с берляндского языка на бирляндский — задача не из легких. Эти языки очень похожи: слово на бирляндском языке отличается от такого же по смыслу слова на берляндском только тем, что оно пишется (и произносится) наоборот. Например, слову *code* в берляндском языке соответствует слово *edoc* в бирляндском. Несмотря на это, при «переводе» легко ошибиться. Вася перевел слово *s* с берляндского на бирляндский как *t*. Помогите ему: определите, правильно ли он выполнил перевод?

### Входные данные

В первой строке записано слово *s*, во второй строке записано слово *t*. Слова состоят из маленьких латинских букв. Входные данные не содержат лишних пробелов. Слова непустые, и их длины не превосходят 100 символов.

### Выходные данные

Если слово *t* является словом *s*, записанным наоборот, выведите YES, иначе выведите NO.

### Примеры

входные данные
code edoc
выходные данные
YES
входные данные
abb aba
выходные данные
NO
входные данные
code code
выходные данные
NO

## Алгоритм

В этой задаче нужно проверить, является ли одна строка реверсом другой строки. Делается стандартной функцией *reverse()*. Сложность  $O(n)$ .

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3
4 using namespace std;
5
6 int main() {
7     string s, t;
8     cin >> s >> t;
9     reverse(t.begin(), t.end());
10    if (s.compare(t) == 0) {
11        cout << "YES" << endl;
12    } else {
13        cout << "NO" << endl;
14    }
15    return 0;
16 }
```

### 3.5 Codeforces Round 258 Div 2

#### Результаты

Задачи			
№	Название		
A	<a href="#">Игра с палочками</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x9805</a>
B	<a href="#">Сортируем массив</a>	стандартный ввод/вывод 1 с, 256 МБ	<a href="#">x5646</a>
C	<a href="#">Предскажите исход игр</a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x2033</a>
D	<a href="#">Считаем хорошие подстроки</a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x1592</a>
E	<a href="#">Devu и цветы</a>	стандартный ввод/вывод 4 с, 256 МБ	<a href="#">x621</a>

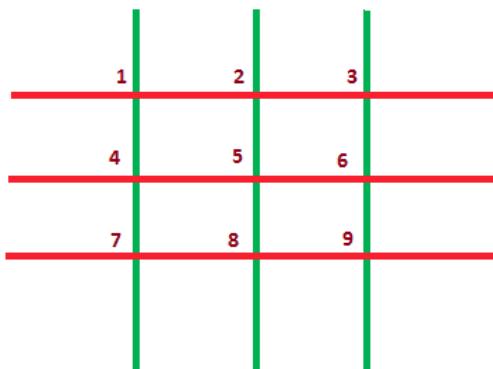
Ссылка на контест: <http://codeforces.com/contest/451>

## Задача А - Игра с палочками

Завоевав золото и серебро в IOI 2014, Akshat и Malvika захотели немного развлечься. Сейчас они заняты игрой на сетке, составленной из  $n$  горизонтальных и  $m$  вертикальных палочек.

Точкой пересечения на сетке называется любая точка сетки, образованная пересечением одной из горизонтальных и одной из вертикальных палочек.

На сетке, показанной ниже,  $n = 3$  и  $m = 3$ . Сетка образована  $n + m = 6$  палочками (вертикальные палочки выделены зеленым цветом, а горизонтальные — красным цветом). На сетке есть  $n \cdot m = 9$  точек пересечения, которые пронумерованы от 1 до 9.



Правила игры очень простые. Игроки ходят по очереди. Akshat завоевал золото, поэтому он делает первый ход. На своем ходу игрок должен выбрать любую оставшуюся точку пересечения, а затем удалить из сетки все палочки, которые проходят через эту точку. Игрок проигрывает, если не может сделать ход (на сетке не осталось точек пересечения).

Предположим, что оба игрока играют оптимально. Кто победит в игре?

### Входные данные

Первая строка содержит два целых числа через пробел,  $n$  и  $m$  ( $1 \leq n, m \leq 100$ ).

### Выходные данные

Выведите "Akshat" или "Malvika" (без кавычек), в зависимости от того, кто победит при оптимальной игре.

### Примеры

входные данные
2 2
выходные данные
Malvika

## Алгоритм

В этой задаче достаточно заметить, что победа игрока зависит от размера сетки. Если минимальное число из размеров сетки четное, то победит Malvika, иначе победит Akshat. Сложность  $O(1)$ .

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3
4 using namespace std;
5
6 int main() {
7     int n, m;
8     cin >> n >> m;
9     int t = n;
10
11    if (n != m) {
12        t = min(n, m);
13    }
14
15    if (t % 2 == 0) {
16        cout << "Malvika" << endl;
17    } else {
18        cout << "Akshat" << endl;
19    }
20
21    return 0;
22 }
```

### 3.6 Codeforces Round 196 Div 2

#### Результаты

Задачи		Nº	Название			
A	<a href="#">Пазлы<sup>1</sup></a>		стандартный ввод/вывод 1 с, 256 МБ			x9974
B	<a href="#">Бытовая задача<sup>1</sup></a>		стандартный ввод/вывод 1 с, 256 МБ			x2503
C	<a href="#">Викторина<sup>1</sup></a>		стандартный ввод/вывод 1 с, 256 МБ			x1515
D	<a href="#">Книга Зла<sup>1</sup></a>		стандартный ввод/вывод 2 с, 256 МБ			x809
E	<a href="#">Дерево делителей</a>		стандартный ввод/вывод 0,5 с, 256 МБ			x284

Ссылка на контест: <http://codeforces.com/contest/337>

## Задача А - Пазлы

Учебный год подходит к концу, и классной руководительнице Манане Тариевне скоро придется прощаться с очередным классом. На прощанье учительница решила подарить каждому из своих  $n$  учеников «пазл» (согласно wikipedia, пазл — игра-головоломка, в которой требуется составить мозаику из множества фрагментов рисунка различной формы).

В магазине учительнице сказали, что у них есть  $m$  пазлов, но они возможно не все одинаковой сложности и размера. Конкретно, первый пазл состоял из  $f_1$  фрагментов, второй — из  $f_2$ , и так далее.

Манана Тариевна решила, что разница между количествами фрагментов в подаренных ею пазлах должна быть как можно меньше, иначе дети могут обидеться. Поэтому она хочет выбрать такие  $n$  пазлов, что если  $A$  — это количество фрагментов в самом большом, а  $B$  — количество фрагментов в самом маленьком из них, то  $A - B$  должно быть минимальным возможным. Помогите учительнице и найдите наименьшую возможную разницу  $A - B$ .

### Входные данные

В первой строке через пробел записаны целые числа  $n$  и  $m$  ( $2 \leq n \leq m \leq 50$ ). Во второй строке через пробел записано  $m$  целых чисел  $f_1, f_2, \dots, f_m$  ( $4 \leq f_i \leq 1000$ ) — количества фрагментов в пазлах, продающихся в магазине.

### Выходные данные

Выведите единственное целое число — минимальную возможную разницу между максимальным и минимальным количеством фрагментов среди пазлов, которые должна приобрести учительница.

### Примеры

входные данные
4 6
10 12 10 7 5 22
выходные данные
5

## Алгоритм

Для решения этой задачи нужно отсортировать имеющиеся в магазине пазлы, а затем пройти по всем пазлам циклом и сравнить позицию  $i$  и позицию  $i + n - 1$ . При этом нужно запоминать минимальную разницу, которая будет ответом. Сложность алгоритма  $O(n \log(n))$ .

## Исходный код

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     int n, m;
5     cin >> n >> m;
6     vector<int> puzzle(m);
7     for (int i = 0; i < m; i++) {
8         cin >> puzzle[i];
9     }
10    sort(puzzle.begin(), puzzle.end());
11    int min = INT_MAX;
12    for (int i = 0; i < m - n + 1; i++) {
13        int diff = puzzle[i + n - 1] - puzzle[i];
14        if (diff < min) {
15            min = diff;
16        }
17    }
18    cout << min << endl;
19    return 0;
20 }
```

### 3.7 Codeforces Round 304 Div 2

#### Результаты

Задачи		Nº	Название			
A	<a href="#">Солдат и бананы</a>		стандартный ввод/вывод 1 с, 256 МБ			x10644
B	<a href="#">Солдат и значки</a>		стандартный ввод/вывод 3 с, 256 МБ			x5865
C	<a href="#">Солдат и карты</a>		стандартный ввод/вывод 2 с, 256 МБ			x4470
D	<a href="#">Солдат и игра с числами</a>		стандартный ввод/вывод 3 с, 256 МБ			x3040
E	<a href="#">Солдат и путешествия</a>		стандартный ввод/вывод 1 с, 256 МБ			x995

Ссылка на контест: <http://codeforces.com/contest/546>

## Задача А - Солдат и бананы

Солдат хочет купить  $w$  бананов в магазине. Ему надо заплатить  $k$  долларов за первый банан,  $2k$  долларов — за второй и так далее (иными словами, за  $i$ -й банан надо заплатить  $i \cdot k$  долларов).

У него есть  $n$  долларов. Сколько долларов ему придется одолжить у однополчанина, чтобы купить  $w$  бананов?

### Входные данные

В первой строке записано три положительных целых числа  $k, n, w$  ( $1 \leq k, w \leq 1000, 0 \leq n \leq 10^9$ ), стоимость первого банана, изначальное количество долларов у солдата и количество бананов, которые он хочет купить.

### Выходные данные

Выведите единственное целое число — количество долларов, которое солдату надо одолжить у однополчанина. Если деньги одолживать не надо, выведите 0.

#### Примеры

входные данные
3 17 4
выходные данные
13

## Алгоритм

В этой задаче надо вычислить сумму арифметической прогрессии и найти разность с  $n$ . Если полученное значение меньше нуля - вывести 0, иначе вывести разность. Сложность алгоритма  $O(1)$ .

## Исходный код

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <climits>
5
6 using namespace std;
7
8 int main() {
9     long k, n, w;
10    cin >> k >> n >> w;
11    long ans = k * ((w + w * w) / 2) - n;
12    if (ans < 0) {
13        ans = 0;
14    }
15    cout << ans << endl;
16    return 0;
17 }
```

### 3.8 Codeforces Round 277 Div 2

#### Результаты

Задачи			
№	Название		
A	<a href="#">Подсчёт функции</a>	стандартный ввод/вывод 1 с, 256 МБ	x9637
B	<a href="#">OR в матрице</a>	стандартный ввод/вывод 1 с, 256 МБ	x4071
C	<a href="#">Преобразование в палиндром</a>	стандартный ввод/вывод 1 с, 256 МБ	x3021
D	<a href="#">Допустимые множества</a>	стандартный ввод/вывод 1 с, 256 МБ	x1122
E	<a href="#">Наибольшие возрастающие подпоследовательности</a>	стандартный ввод/вывод 2 с, 256 МБ	x632

Ссылка на контест: <http://codeforces.com/contest/486>

## Задача А - Подсчет функции

Для положительного целого числа  $n$  определим функцию  $f$ :

$$f(n) = -1 + 2 - 3 + \dots + (-1)^n n$$

Ваша задача — посчитать  $f(n)$  для данного целого числа  $n$ .

### Входные данные

В единственной строке записано положительное целое число  $n$  ( $1 \leq n \leq 10^{15}$ ).

### Выходные данные

Выведите  $f(n)$  в единственной строке.

#### Примеры тестов

входные данные	выходные данные
4	2
входные данные	выходные данные
5	-3

## Алгоритм

В этой задаче нужно просто определить закономерность по предложенному ряду и написать функцию вычисления значения функции для произвольного входного числа. Сложность  $O(1)$ .

## Исходный код

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     long long n;
7     cin >> n;
8     long long ans = n / 2;
9     if (n % 2 != 0) {
10         ans = -ans - 1;
11     }
12     cout << ans << endl;
13     return 0;
14 }
```

### 3.9 Codeforces Round 103 Div 2

#### Результаты

Задачи			
№	Название		
A	<a href="#">Приезд генерала<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x9404</a>
B	<a href="#">Совещание<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x2119</a>
C	<a href="#">Поиск анаграмм<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x1864</a>
D	<a href="#">Ракетные шахты<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x1177</a>
E	<a href="#">Соревнование<sup>1</sup></a>	стандартный ввод/вывод 2 с, 256 МБ	<a href="#">x286</a>

Ссылка на контест: <http://codeforces.com/contest/144>

## Задача А - Приезд генерала

В Самую Секретную Военную Часть под командованием полковника Покрышкина приехал с проверкой генерал из Министерства Обороны. По этому случаю полковник приказал всем  $n$  солдатам из своей Части построиться на плацу.

Согласно военному уставу, солдаты должны стоять в порядке невозрастания их роста, но так как времени на построение совсем не осталось, то солдаты выстроились в произвольном порядке. Однако у генерала весьма плохое зрение, и поэтому он считает, что солдаты построены правильно, если самый первый в строю — солдат с максимальным ростом, а самый последний — солдат с минимальным ростом. Обратите внимание, что неважно то, как расположены остальные солдаты, в том числе и в случае нескольких максимальных или минимальных по росту солдат. Важны лишь росты **первого** и **последнего** солдата.

Например, генерал считает последовательность ростов  $(4, 3, 4, 2, 1, 1)$  правильной, а последовательность  $(4, 3, 1, 2, 2)$  — нет.

За одну секунду полковник может обменять местами любых двух соседних солдат. Помогите ему подсчитать, какое минимальное количество секунд понадобится, чтобы получившийся строй понравился генералу.

### Входные данные

Первая строка входных данных содержит единственное целое число  $n$  ( $2 \leq n \leq 100$ ) — количество солдат в строю. Вторая строка содержит целые числа  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 100$ ) — величины ростов солдат в порядке от начала строя к концу. Числа разделены пробелом. Числа  $a_1, a_2, \dots, a_n$  не обязательно различны.

### Выходные данные

Выведите единственное целое число — какое минимальное количество секунд понадобится полковнику, чтобы получившийся строй понравился генералу.

### Примеры

входные данные	
4	
33 44 11 22	
выходные данные	
2	
входные данные	
7	
10 10 58 31 63 40 76	
выходные данные	
10	

## Алгоритм

Для решения этой задачи нужно найти самого высокого и самого низкого солдата, а затем посчитать количество обменов для того чтобы поставить их в начало и конец. Сложность  $O(1)$ .

## Исходный код

```
1 #include <iostream>
2 #include <climits>
3
4 using namespace std;
5
6 int main() {
7     int n, h;
8     cin >> n;
9     int fh = INT_MIN;
10    int fi = n;
11    int lh = INT_MAX;
12    int li = -1;
13    for (int i = 0; i < n; i++) {
14        cin >> h;
15        if (h > fh) {
16            fh = h;
17            fi = i;
18        } else if (h == fh && i < fi) {
19            fi = i;
20        }
21        if (h < lh) {
22            lh = h;
23            li = i;
24        } else if (h == lh && i > li) {
25            li = i;
26        }
27    }
28    int ans = fi + n - li - 1;
29    if (fi > li) {
30        ans--;
31    }
32    cout << ans << endl;
33 }
```

### 3.10 VK Cup 2016 - Квалификация 2

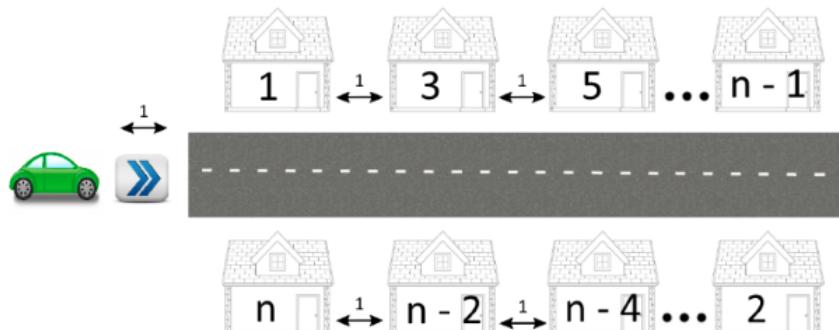
#### Результаты

Задачи		Название		
№				
A	<a href="#">Номера домов</a>	стандартный ввод/вывод 1 с, 256 МБ	 	 x1597
B	<a href="#">Сборка генома в Берляндии</a>	стандартный ввод/вывод 1 с, 256 МБ	 	 x757
C	<a href="#">Ремонт дорог</a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x404
D	<a href="#">Трёхмерный черепаший суперкомпьютер</a>	стандартный ввод/вывод 3 с, 256 МБ	 	 x329

Ссылка на контест: <http://codeforces.com/contest/638>

## Задача А - Номера домов

Главная улица в Берляндии представляет собой прямую, вдоль которой построены  $n$  домов ( $n$  — чётное число). Дома расположены с обеих сторон улицы. Дома с нечётными номерами находятся с одной стороны улицы и нумеруются от 1 до  $n - 1$  в порядке от начала улицы к её концу (на картинке слева направо). Дома с чётными номерами находятся с другой стороны улицы и нумеруются от 2 до  $n$  в порядке от конца улицы к её началу (на картинке справа налево). Соответствующие дома с чётными и нечётными номерами находятся строго друг напротив друга, то есть напротив дома номер 1 находится дом номер  $n$ , напротив дома номер 3 находится дом номер  $n - 2$ , напротив дома номер 5 находится дом номер  $n - 4$  и так далее.



Вася необходимо как можно скорее добраться до дома номер  $a$ . Он заезжает с начала улицы и движется на автомобиле по ней до дома номер  $a$ . Чтобы доехать от начала улицы до домов с номерами 1 и  $n$ , он тратит ровно 1 секунду. На то, чтобы проехать расстояние между двумя соседними домами, он также тратит ровно одну секунду. Вася может припарковаться с любой стороны дороги, поэтому расстояние от начала улицы до домов, стоящих друг напротив друга, следует считать одинаковым.

Перед вами стоит задача: найти минимальное время, по истечении которого Вася сможет добраться до дома с номером  $a$ .

### Входные данные

В первой строке входных данных содержатся два целых числа  $n$  и  $a$  ( $1 \leq a \leq n \leq 100\,000$ ) — количество домов на улице и номер дома, до которого нужно доехать Вася, соответственно. Гарантируется, что число  $n$  чётно.

### Выходные данные

Выведите единственное целое число — минимальное время, за которое Вася сможет добраться от начала улицы до дома с номером  $a$ .

### Примеры

входные данные	
4 2	
выходные данные	
2	
входные данные	
8 5	
выходные данные	
3	

## Алгоритм

Для решения этой задачи нужно вывести формулу. Сложность  $O(1)$ .

## Исходный код

```
1 n, a = gets.split(' ').map(&:to_i)
2 puts a.even? ? 1 + n / 2 - a / 2 : 1 + a / 2
```

## Задача В - Сборка генома в Берляндии

Перед берляндскими учёными всталася важнейшая задача — восстановить по имеющимся частям коротких фрагментов ДНК геном динозавра! У берляндского динозавра геном совсем не похож на привычный нам: в нём могут встречаться 26 различных типов нуклеотидов, причём нуклеотид каждого типа может встречаться **не более одного** раза. Если присвоить всем нуклеотидам различные буквы английского алфавита, то геном берляндского динозавра будет представлять собой непустую строку, состоящую из строчных букв английского алфавита, такую что каждая буква встречается в ней не более одного раза.

У учёных есть  $n$  фрагментов генома, которые представляют собой **подстроки** (непустые последовательности подряд идущих нуклеотидов) искомого генома.

Перед вами стоит задача: помочь учёным восстановить геном динозавра. Гарантируется, что входные данные непротиворечивы и хотя бы одна подходящая строка всегда существует. Узнав, что вы сильный программист, учёные дополнительно попросили вас из подходящих строк выбрать ту, длина которой минимальна. Если и таких строк несколько, то их устроит любая.

### Входные данные

В первой строке входных данных следует целое положительное число  $n$  ( $1 \leq n \leq 100$ ) — количество фрагментов генома.

В каждой из следующих строк следует по одному описанию фрагмента. Каждый фрагмент — это непустая строка, состоящая из различных строчных букв английского алфавита. Не гарантируется, что заданные фрагменты различны. Фрагменты могли перекрываться произвольным образом, и один фрагмент мог быть подстрокой другого.

Гарантируется, что найдется такая строка из различных букв, которая содержит все заданные фрагменты в качестве подстрок.

### Выходные данные

В единственной строке выходных данных выведите геном минимальной длины, который содержит все заданные части. Все нуклеотиды в геноме должны быть различными. Если подходящих строк несколько, выведите строку минимальной длины. Если подходящих строк минимальной длины также несколько, то разрешается вывести любую из них.

#### Примеры

<b>входные данные</b>
3
bcd
ab
cdef
<b>выходные данные</b>
abcdef
<b>входные данные</b>
4
x
y
z
w
<b>выходные данные</b>
xyzw

## Алгоритм

Для решения этой задачи нужно составить из заданных строк составить большую строку, учитывая что маленькие строки могут пересекаться. Сложность  $O(n^2)$ .

## Исходный код

```
1 class GenomPart
2     @genom
3     @chars
4     def initialize
5         @genom = ""
6         @chars = []
7     end
8     def is_valid_part(part)
9         if @genom == ""
10            return true
11        end
12        for i in (0...part.size)
13            if @chars.include?(part[i])
14                return true
15            end
16        end
17        return false
18    end
19    def insert_part(part)
20        if @genom == ""
21            @genom = part
22            @chars = part.chars
23            return
24        end
25        @chars |= part.chars
26        count = @genom.count(part)
27        if count == @genom.size # genon is substring of part
28            @genom = part
29            #@@chars = part.chars
30        elsif count == part.size
31            # nothing
32        else
33            if @genom.index(part[0...count]) != nil
34                @genom = (@genom.chars | part.chars).join
35            else
36                @genom = (part.chars | @genom.chars).join
37            end
38            #@@chars = @genom.chars
39        end
40    end
41    def genom
42        @genom
43    end
44 end
45
46 n = gets.to_i
47 parts = []
48 n.times do
49     s = gets.chomp
50     parts.push(s)
51 end
52
53 genom_parts = []
54 while !parts.empty?
55     genom = GenomPart.new
56     while true
57         action = false
58         temp = parts.clone
```

```
59     temp.each do |item|
60       if genom.is_valid_part(item)
61         action = true
62         genom.insert_part(item)
63         parts.delete(item)
64       end
65     end
66     if !action
67       break
68     end
69   end
70   genom_parts.push(genom)
71 end
72
73 answer = ""
74 genom_parts.each do |part|
75   answer += part.genom
76 end
77
78 puts answer
```

### 3.11 VK Cup 2016 - Квалификация 1

#### Результаты

Задачи		Название		
№				
A	<a href="#">Голосование за фотографии</a>	стандартный ввод/вывод 1 с, 256 МБ	 	 x2097
B	<a href="#">Порядок чатов</a>	стандартный ввод/вывод 3 с, 256 МБ	 	 x1933
C	<a href="#">Промокоды с ошибками</a>	стандартный ввод/вывод 1 с, 256 МБ	 	 x1238
D	<a href="#">Бег с препятствиями</a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x955

Ссылка на контест: <http://codeforces.com/contest/638>

## Задача В - Порядок чатов

Поликарп – большой любитель проводить время в соцсетях. Страница со списком чатов в его любимой соцсети устроена так, что при отсылке сообщения собеседнику чат с ним поднимается на самый верх страницы. Относительный порядок всех остальных чатов при этом не изменяется. Если ранее чата с этим собеседником не было, то просто в верх списка чатов вставляется новый чат.

Считая, что изначально список чатов пуст, по последовательности сообщений Поликарпа постройте список чатов после обработки всех его сообщений. Считайте, что никто из собеседников ничего Поликарпу не писал.

### Входные данные

В первой строке записано целое число  $n$  ( $1 \leq n \leq 200\,000$ ) – количество сообщений Поликарпа. Далее в  $n$  строках перечислены адресаты сообщений в порядке отсылки сообщений. Имя каждого адресата – непустая последовательность строчных букв английского алфавита длины не более 10.

### Выходные данные

Выведите всех адресатов, с кем общался Поликарп, в порядке расположения чатов с ними сверху вниз.

### Примеры

входные данные
4 alex ivan roman ivan
выходные данные
ivan roman alex
входные данные
8 alina maria ekaterina darya darya ekaterina maria alina
выходные данные
alina maria ekaterina darya

## Алгоритм

Для решения этой задачи нужно вставить все имена в массив, а затем доставать их по одному и проверять наличие в множестве. Если в множестве этого имени нет, то вывести его и добавить в множество. Сложность  $O(\log(n))$ .

## Исходный код

```
1 n = gets.to_i
2 chats = []
3 n.times do
4   chats.push(gets.chomp)
5 end
6 s = Set.new
7 chats.reverse.each do |name|
8   if !s.include?(name)
9     puts name
10    s << name
11  end
12 end
```

## 3.12 VK Cup 2016 - Раунд 1

### Результаты

Задачи		Название		
№				
A	<a href="#">Медвежонок и отображаемые друзья</a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x1183
B	<a href="#">Медвежонок и забытое дерево 3</a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x935
C	<a href="#">Медвежонок и многочлены</a>	стандартный ввод/вывод 2 с, 256 МБ	 	 x302
D	<a href="#">Медвежонок и вклад</a>	стандартный ввод/вывод 4 с, 256 МБ	 	 x181
E	<a href="#">Медведь и парадокс</a>	стандартный ввод/вывод 3,5 с, 256 МБ	 	 x55
F	<a href="#">Медведь и химия</a>	стандартный ввод/вывод 6 с, 256 МБ	 	 x27

Ссылка на контест: <http://codeforces.com/contest/639>

## Задача А - Медвежонок и отображаемые друзья

Полярный медвежонок Лимак любит переписываться с другими медведями в социальных сетях. Всего у него  $n$  друзей и уровень дружбы с  $i$ -м другом описывается целым числом  $t_i$ . Чем больше это значение, тем сильнее дружат два медвежонка. Известно, что все  $t_i$  различны.

Приближается весна и медведи начинают просыпаться. Лимак проснулся самым первым и сразу зашёл в любимую социальную сеть. Все его друзья ещё спят и пока что находятся оффлайн. В ближайшие часы некоторые из них проснутся и тоже сразу зайдут в социальную сеть.

В системе в специальном окне отображаются те из друзей, кто сейчас находится в сети, но не более чем  $k$  медведей. Если онлайн находится больше чем  $k$ , то из них показываются  $k$  самых близких друзей (то есть с наибольшими значениями  $t_i$ ).

Необходимо обрабатывать запросы двух типов:

- «1  $id$ » — друг с номером  $id$  заходит в социальную сеть. Гарантируется, что до этого он не был онлайн.
- «2  $id$ » — проверить, отображается ли друг с номером  $id$  в специальном окне. Вывести «YES» или «NO».

Как вы помните, Лимак ещё очень маленький, поэтому ему требуется ваша помощь в обработке запросов.

### Входные данные

В первой строке входных данных записаны три числа  $n, k$  и  $q$  ( $1 \leq n, q \leq 150\,000$ ,  $1 \leq k \leq \min(6, n)$ ) — количество друзей, размер специального окна для отображения друзей онлайн и количество запросов соответственно.

Во второй строке записаны  $n$  целых чисел  $t_1, t_2, \dots, t_n$  ( $1 \leq t_i \leq 10^9$ ),  $i$ -е из которых показывает уровень дружбы Лимака с другом номер  $i$ .

В  $i$ -й из последующих  $q$  строк записаны два целых числа  $type_i$  и  $id_i$  ( $1 \leq type_i \leq 2$ ,  $1 \leq id_i \leq n$ ) — параметры  $i$ -го запроса. Если  $type_i = 1$ , то данный запрос означает, что друг  $id_i$  зашёл в социальную сеть. Если же  $type_i = 2$ , то требуется проверить, отображается ли друг  $id_i$  в специальном окошке.

Гарантируется, что все запросы первого типа содержат различные  $id_i$  и что во входных данных присутствует хотя бы один запрос второго типа.

### Выходные данные

Для каждого запроса второго типа выведите «YES» (без кавычек), если соответствующий друг присутствует в этот момент в специальном окне, и «NO» (без кавычек), если не присутствует.

### Примеры

входные данные
4 2 8
300 950 500 200
1 3
2 4
2 3
1 1
1 2
2 1
2 2
2 3

выходные данные
NO
YES
NO
YES
YES

## Алгоритм

Задача на реализацию. Нужно смоделировать процесс, описанный в условии задачи. Сложность  $O(n * \log(n))$ .

## Исходный код

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4 import java.util.Scanner;
5
6 public class C {
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         int n, k, q;
10        n = sc.nextInt();
11        k = sc.nextInt();
12        q = sc.nextInt();
13        long[] friendsLevel = new long[n];
14        for(int i = 0; i < n; i++) {
15            friendsLevel[i] = sc.nextLong();
16        }
17        boolean[] isOnline = new boolean[n+1];
18        for(int i = 0; i < n+1; i++){
19            isOnline[i] = false;
20        }
21
22        int first, second;
23
24        ArrayList<long[]> window = new ArrayList<>();
25
26        for(int i = 0; i < q; i++) {
27            long minT = 1000000002;
28            int minIndex = -1;
29            first = sc.nextInt();
30            second = sc.nextInt();
31            if(first == 1){
32                isOnline[second] = true;
33                if(window.size() < k) {
34                    long[] temp = new long[2];
35                    temp[0] = (long)second;
36                    temp[1] = friendsLevel[second-1];
37                    //Pair temp = new Pair((long)second, friendsLevel[second-1]);
38                    window.add(temp);
39                } else {
40                    for(int j = 0; j < k; j++) {
41                        if(window.get(j)[1] < minT){
42                            minT = window.get(j)[1];
43                            minIndex = j;
44                        }
45                    }
46                    if(friendsLevel[second - 1] > minT){
47                        long[] temp = new long[2];
48                        temp[0] = (long)second;
49                        temp[1] = friendsLevel[second - 1];
50                        //Pair temp = new Pair((long)second, friendsLevel[second
51                        - 1]);
52                        window.set(minIndex, temp);
53                    }
54                }
55            } else {
56                if(!isOnline[second]) {
57                    System.out.println("NO");
58                }
59            }
60        }
61    }
62}
```

```
58 } else {
59     boolean flag = false;
60     //Pair temp = new Pair((long)second, friendsLevel[second - 1]);
61     for(int j = 0; j < window.size(); j++) {
62         if(window.get(j)[0] == (long)second) {
63             System.out.println("YES");
64             flag = true;
65             break;
66         }
67     }
68     if(!flag){
69         System.out.println("NO");
70     }
71 }
72 }
73 }
74 }
75 }
```

### 3.13 VK Cup 2016 - Уайлд-кард раунд 1

#### Результаты

Задачи		Название		
№				
A	<a href="#">Ленивый блинорез</a>	стандартный ввод/вывод 2 с, 64 МБ		x173
B	<a href="#">Времена года</a>	стандартный ввод/вывод 2 с, 64 МБ		x67
C	<a href="#">Сумма массива</a>	стандартный ввод/вывод 2 с, 64 МБ		x23
D	<a href="#">Максимальная разность</a>	стандартный ввод/вывод 2 с, 64 МБ		x24
E	<a href="#">Проверка делимости</a>	стандартный ввод/вывод 2 с, 64 МБ		x12
F	<a href="#">Простые числа на интервале</a>	стандартный ввод/вывод 2 с, 64 МБ		x20
G	<a href="#">Венгерская нотация</a>	стандартный ввод/вывод 2 с, 64 МБ		x3
H	<a href="#">Поворот матрицы</a>	стандартный ввод/вывод 2 с, 64 МБ		

Ссылка на контест: <http://codeforces.com/contest/640>

## Задача А - Медвежонок и отображаемые друзья

Последовательность ленивого блинореза определяется следующим образом:  $n$ -й элемент последовательности равен максимальному количеству кусков, на которые можно разрезать выпуклый блин  $n$  прямолинейными разрезами. Формула для подсчёта  $n$ -го элемента:  $C_n = n \cdot (n + 1) / 2 + 1$ . Вам дан номер  $n$ , вычислите  $n$ -й элемент последовательности.

### Входные данные

Единственная строка входных данных содержит целое число  $n$  ( $0 \leq n \leq 100$ ).

### Выходные данные

Выведите  $n$ -й элемент последовательности.

### Примеры

входные данные	
2	
выходные данные	
4	
входные данные	
5	
выходные данные	
16	

## Алгоритм

Сложность задачи в том, что сдавать ее можно было только на неизвестном многим языке J.

## Исходный код

```
1 print =: 1!:2&2
2 read =: 1!:1[3
3
4 in =. (read -.LF) -.CR
5 print 1 + ((1+ ".in)*".in) %2
6
7 exit ,
```

### 3.14 Vekua Cup 2016 Личный этап

#### Результаты

96	Макаров Никита (МАИ)	+ 00:29	—	—	—	—	+1 02:06	—	—	2	176
----	----------------------	------------	---	---	---	---	-------------	---	---	---	-----

Ссылка на контест: <https://official.contest.yandex.ru/contest/2463>

## Задача А - Adjusted Number

Lets say that integer  $a$  is *adjusted* integer  $b$ , if  $a$  contains the same number of zeroes and same number of ones in its binary representation and its minimal possible.

Given an integer, find out its adjusted number.

### Input format

Input contains one integer  $N$  ( $0 \leq N \leq 10^{18}$ ).

### Output format

Print the answer — adjusted number for  $N$ .

#### Sample 1

Input	Output
0	0

#### Sample 2

Input	Output
1	1

### Алгоритм

В этой задаче нужно подсчитать количество ненулевых битов во входном числе, а затем подвинуть их все на младшие позиции и один бит оставить на самой старшей позиции в этом числе. Полученное число будет ответом. Сложность решения  $O(n)$ .

## Исходный код

```
1 #include <iostream>
2 #include <cmath>
3
4 using namespace std;
5
6 int main(int argc, const char * argv[]) {
7
8     long n;
9     cin >> n;
10
11    int zeros = 0;
12    int ones = 0;
13
14    int lim = (int)log2(n);
15
16    for (int i = 0; i <= lim; i++) {
17        if (n & (1L << i)) {
18            ones++;
19        } else {
20            zeros++;
21        }
22    }
23
24    long answer = 0;
25    for (int i = 0; i < (ones - 1); i++) {
26        answer |= (1L << i);
27    }
28
29    if (ones > 0) {
30        answer |= (1L << lim);
31    }
32
33    cout << answer << endl;
34
35    return 0;
36 }
```

## Задача F - Forcecoders

The famous Forcecoders project starts the tournament in paired programming. But at this time, pairs will be formed by random assignment.

$N$  coders, including yourself, are registered to the contest. The rating of  $i$ -th player (aside from yourself) is equal to  $R_i$ . Your rating is equal to  $R$ .

Expected rank of the pair is defined as the number of pairs (including this one) with sum of ratings not less than their sum of ratings.

You are wondering, which is lower (and thus better — first ranked team wins the tournament) expected rank you may have in optimal for you assigment of pairs.

### Input format

First line of the input contains two integers — number of players  $N$  and your rating  $R$  ( $1 \leq R, N \leq 1000$ ,  $N$  is even). Secont line contains  $N-1$  integers  $R_i$  — ratings of the other contestants, given in non-decreasing order ( $1 \leq R_1 \leq R_2 \dots \leq R_{n-1} \leq 1000$ ).

### Output format

Print one integer — the best possible expected rank for your pair.

### Sample

Input	Output
8 20	2
40	
50	
80	
84	
90	
110	
120	

### Алгоритм

В этой задаче нужно минимизировать сумму пары чисел из массива. Для этого отсортируем массив и будем брать элементы с разных концов. Если сумма получается больше собственного счета, значит мы не можем покрыть одно из этих значений и сдвигаем большую границу на 2, иначе сдвигаем обе границы на 1. Ответом является количество больших сумм. Сложность решения  $O(n * \log(n))$ .

## Исходный код

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4 int main(int argc, const char * argv[]) {
5     int n, r;
6     cin >> n >> r;
7     vector<int> ranking(n - 2);
8     for (int i = 0; i < (n - 2); i++) {
9         cin >> ranking[i];
10    }
11    int last;
12    cin >> last;
13    int own = r + last;
14    n = n - 2;
15    int res_ranking = 0;
16    int first_big_pos = -1;
17    for (int i = n - 1; i >= 0; i--) {
18        if (ranking[i] >= own) {
19            res_ranking++;
20            first_big_pos = i;
21        } else {
22            break;
23        }
24    }
25    int begin_pos = 0;
26    int end_pos = 0;
27    if (first_big_pos == -1) {
28        end_pos = n - 1;
29    } else {
30        end_pos = first_big_pos - 1;
31    }
32    if (res_ranking % 2 == 0) {
33        res_ranking /= 2;
34    } else {
35        res_ranking = res_ranking / 2 + 1;
36        end_pos--;
37    }
38    while (begin_pos < end_pos) {
39        if (ranking[begin_pos] + ranking[end_pos] >= own) {
40            res_ranking++;
41            end_pos -= 2;
42        } else {
43            begin_pos++;
44            end_pos--;
45        }
46    }
47    cout << res_ranking + 1 << endl;
48    return 0;
49 }
```

## 4 Журнал по личным контестам Якименко А.В.

### 4.1 Vekua Cup 2016 Личный этап

#### Результаты

67-68	Якименко Антон (МАИ)	+	00:15	?	01:30	-	-	-	+	01:10	-	-	2	86
-------	----------------------	---	-------	---	-------	---	---	---	---	-------	---	---	---	----

Ссылка на контест: <https://official.contest.yandex.ru/contest/2463>

## Задача А - Adjusted Number

Lets say that integer  $a$  is *adjusted* integer  $b$ , if  $a$  contains the same number of zeroes and same number of ones in its binary representation and its minimal possible.

Given an integer, find out its adjusted number.

### Input format

Input contains one integer  $N$  ( $0 \leq N \leq 10^{18}$ ).

### Output format

Print the answer — adjusted number for  $N$ .

#### Sample 1

Input	Output
0	0

#### Sample 2

Input	Output
1	1

### Алгоритм

В этой задаче нужно посчитать количество нулей и единиц в двоичной записи данного числа и затем составить число с тем же количеством нулей и единиц и чтобы оно было минимальным. Сложность  $O(\log_2(n))$ .

## Исходный код

```
1 use bigint;
2 my $a = <>;
3 if ($a == 0) {
4     print "0\n";
5     exit;
6 }
7 my $t = $a;
8 my ($z, $o) = (0, 0);
9 while ($t > 0) {
10    if ($t & 1) {
11        ++$o;
12    }
13    else {
14        ++$z;
15    }
16    $t >>= 1;
17 }
18 my $b = 1;
19 --$o;
20 $b <= $z;
21 while ($o) {
22    $b = ($b << 1) | 1;
23    --$o;
24 }
25 print $b;
```

## Задача F - Forcecoders

The famous Forcecoders project starts the tournament in paired programming. But at this time, pairs will be formed by random assignment.

$N$  coders, including yourself, are registered to the contest. The rating of  $i$ -th player (aside from yourself) is equal to  $R_i$ . Your rating is equal to  $R$ .

Expected rank of the pair is defined as the number of pairs (including this one) with sum of ratings not less than their sum of ratings.

You are wondering, which is lower (and thus better — first ranked team wins the tournament) expected rank you may have in optimal for you assigment of pairs.

### Input format

First line of the input contains two integers — number of players  $N$  and your rating  $R$  ( $1 \leq R, N \leq 1000$ ,  $N$  is even). Secont line contains  $N-1$  integers  $R_i$  — ratings of the other contestants, given in non-decreasing order ( $1 \leq R_1 \leq R_2 \dots \leq R_{n-1} \leq 1000$ ).

### Output format

Print one integer — the best possible expected rank for your pair.

### Sample

Input	Output
8 20	2
40	
50	
80	
84	
90	
110	
120	

### Алгоритм

В этой задаче нужно минимизировать ваш ранг, путём оптимального выбора пар для всех участников. Сложность  $O(n)$ .

## Исходный код

```
1 my ($n, $myr) = split ' ', <>;
2 my @m;
3 for (my $i = 0; $i < $n - 1; ++$i) {
4     my $d = <>;
5     push @m, $d;
6 }
7 my ($l, $r) = (0, $n - 3);
8 my $or = $m[$#m] + $myr;
9 my $res = 1;
10 while ($l < $r) {
11     if ($m[$l] + $m[$r] >= $or) {
12         $res += 1;
13         $r -= 2;
14     }
15     else {
16         --$r;
17         ++$l;
18     }
19 }
20 print $res, "\n";
```