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

1 Личные отчеты **3**

2 Журнал по командным контестам **7**

2.1 OpenCup GrandPrix of Ukraine (4 solved) + (алгоритм в F) . . . 7
2.2 OpenCup GrandPrix of Japan (5 solved) + DONE . . . 13
2.3 OpenCup GrandPrix of Eurasia (4 solved) + (алгоритм в 9, 12) . . . 22
2.4 OpenCup GrandPrix of SPb (2 solved) + (алгоритм в E) . . . 35
2.5 ACM-ICPC Московский четвертьфинал + DONE . . . 41
2.6 OpenCup GrandPrix of Yekaterinburg (4 solved) + (алгоритм в B) . . . 42
2.7 OpenCup GrandPrix of Siberia (6 solved) + (алгоритм в B, K) . . . 50
2.8 OpenCup GrandPrix of Europe (3 solved) + DONE . . . 61
2.9 OpenCup GrandPrix of Peterhof (4 solved) + (алгоритм в A, E, G) . . . 69
2.10 Codeforces ACM-ICPC Южный четвертьфинал . . . 77
2.11 ACM-ICPC Московский четвертьфинал . . . 85
2.12 Codeforces Training S02E07 . . . 86
2.13 Codeforces Crypto Cup 1.0 . . . 90
2.14 OpenCup GrandPrix of Siberia . . . 101
2.15 Codeforces Training S02E08 . . . 108
2.16 Codeforces Training S02E09 . . . 115
2.17 Codeforces Олимпиада школьников Нижегородской обл. . . . 122
2.18 OpenCup GrandPrix of Central Europe . . . 127
2.19 Codeforces Training S02E10 . . . 130
2.20 OpenCup GrandPrix of Europe . . . 136
2.21 OpenCup GrandPrix of Peterhof . . . 149
2.22 OpenCup GrandPrix of Japan . . . 153
2.23 OpenCup Northern GrandPrix . . . 161
2.24 OpenCup GrandPrix of Karelia . . . 168
2.25 OpenCup GrandPrix of Udmurtia . . . 178
2.26 OpenCup GrandPrix of China . . . 192
2.27 OpenCup GrandPrix of Tatarstan . . . 195
2.28 OpenCup GrandPrix of America . . . 198
2.29 Vekua Cup 2015 Командный этап . . . 208
2.30 OpenCup GrandPrix of Ural . . . 209

3 Журнал по личным контестам Макарова Н.А. **214**

3.1 Codeforces Round 267 Div 2 . . . 214
3.2 Codeforces Round 268 Div 2 . . . 218
3.3 Codeforces Отборочный контест СГАУ на четвертьфинал ACM-ICPC . . . 222
3.4 Codeforces Round 270 Div 2 . . . 227
3.5 Codeforces Round 273 Div 2 . . . 230
3.6 Codeforces Round 274 Div 2 . . . 233
3.7 Codeforces Round 275 Div 2 . . . 239
3.8 VK Cup 2015 Квалификация . . . 241
3.9 VK Cup 2015 - Уайлд-кард раунд 1 . . . 246
3.10 Vekua Cup 2015 Личный этап . . . 248
3.11 Mail.ru Russian Code Cup 2015 Квалификация . . . 249

4 Журнал по личным контестам Якименко А.В. 252
4.1 Codeforces Round 267 Div 2 . . . 252
4.2 Codeforces Round 268 Div 2 . . . 256
4.3 Codeforces Отборочный контест СГАУ на четвертьфинал ACM-ICPC . . . 260
4.4 Codeforces Round 269 Div 2 . . . 262
4.5 Codeforces Round 270 Div 2 . . . 265
4.6 Codeforces Round 272 Div 2 . . . 268
4.7 Codeforces Round 273 Div 2 . . . 272
4.8 Codeforces Round 274 Div 2 . . . 275
4.9 Codeforces Round 275 Div 2 . . . 281
4.10 Codeforces Round 276 Div 2 . . . 284
4.11 Codeforces Round 277 Div 2 . . . 290
4.12 Codeforces Round 277.5 Div 2 . . . 295
4.13 VK Cup 2015 Квалификация 2 . . . 304
4.14 VK Cup 2015 - Раунд 1 . . . 309
4.15 VK Cup 2015 - Уайлд-кард раунд 1 . . . 313
4.16 Vekua Cup 2015 Личный этап . . . 316

1 Личные отчеты

Отчет о работе студента Макарова Н.А. по индивидуальному учебному плану в V-VI семестрах 2014-2015 учебного года.

№	Дата	Контест	Место проведения	Кол-во участников	Решено задач	Задач на участника
1	18.09.2014	Codeforces Round 267 Div 2	Дом	1	2	2
2	21.09.2014	Codeforces Round 268 Div 2	Дом	1	2	2
3	22.09.2014	Codeforces Отборочный контест СГАУ на 1/4 ACM-ICPC	Дом	1	3	3
4	25.09.2014	Codeforces Training S02E03	МАИ	3	3	1
5	28.09.2014	Codeforces Round 270 Div 2	Дом	1	2	2
6	02.10.2014	Codeforces Training S02E04	МАИ	3	1	0.33
7	05.10.2014	XV Открытая Всесибирская Олимпиада по Программированию	МАИ	3	1	0.33
8	09.10.2014	Codeforces Training S02E05	МАИ	3	4	1.33
9	16.10.2014	Codeforces Round 273 Div 2	Дом	1	2	2
10	17.10.2014	Codeforces Training S02E06	МАИ	3	0	0
11	18.10.2014	Codeforces Тренировка СПбГУ графы и DFS	Дом	3	2	0.66
12	19.10.2014	OpenCup GP of SPb. Div 2	МАИ	3	1	0.33
13	20.10.2014	Codeforces Round 274 Div 2	Дом	1	3	3
14	23.10.2014	Codeforces Самарский Аэрокосмический Лицей тренировка №1	Дом	2	1	0.5
15	23.10.2014	Codeforces ACM, NEERC, Восточный четвертьфинал	Дом	3	4	1.33
16	24.10.2014	Codeforces Round 275 Div 2	Дом	1	1	1
17	25.10.2014	Codeforces ACM, NEERC, Южный четвертьфинал	Дом	3	3	1
18	26.10.2014	ACM-ICPC 1/4 Final	МГУ	3	3	1
19	30.10.2014	Codeforces Training S02E07	МАИ	3	2	0.66
20	01.11.2014	Codeforces Crypto Cup	Дом	3	9	3
21	02.11.2014	OpenCup GP of Siberia Div 2	МАИ	2	3	1.5
22	06.11.2014	Codeforces Training S02E08	МАИ	3	2	0.66
23	13.11.2014	Codeforces Training S02E09	МАИ	3	3	1
24	15.11.2014	Codeforces Олимпиада школьников Нижегородской области	Дом	2	3	1.5
25	16.11.2014	OpenCup GP of Central Europe. Div 2	МАИ	3	1	0.33
26	20.11.2014	Codeforces Training S02E10	МАИ	3	3	1
27	23.11.2014	OpenCup GP of Europe Div 2	МАИ	3	5	1.66
28	14.12.2014	OpenCup GP of Peterhof Div 2	МАИ	3	1	0.33
29	01.02.2015	OpenCup GP of Japan Div 2	МАИ	3	4	1.33
30	08.02.2015	OpenCup Northern GP Div 2	МАИ	3	2	0.66
31	15.02.2015	OpenCup GP of Karelia Div 2	МАИ	3	4	1.33

Продолжение таблицы.

№	Дата	Контест	Место проведения	Кол-во участников	Решено задач	Задач на участника
32	22.02.2015	OpenCup GP of Udmurtia Div 2	МАИ	3	4	1.33
33	01.03.2015	OpenCup GP of China Div 2	МАИ	3	1	0.33
34	07.03.2015	VK Cup 2015 Квалификация	Дом	1	2	2
35	15.03.2015	OpenCup GP of Tatarstan Div 2	МАИ	3	1	0.33
36	21.03.2015	VK Cup 2015 Раунд 1	Дом	1	2	2
37	29.03.2015	OpenCup Gp of America Div 2	МАИ	3	4	1.33
38	18.04.2015	Vekua Cup Личный этап	МФТИ-1С	1	1	1
39	19.04.2015	Vekua Cup Командный этап	МФТИ-1С	3	3	1
40	26.04.2015	OpenCup GP of Ural Div 2	МАИ	2	2	1
41	31.05.2105	Mail.ru RCC Квалификация	Дом	1	1	1

Итого: 41 контест, 101 решенная задача. Личный вклад ≈49 задач (командные + личные).

Студент:_____ Макаров Н.А. Руководитель:_____ Зайцев В.Е.

Отчет о работе студента Якименко А.В. по индивидуальному учебному плану в V-VI семестрах 2014-2015 учебного года.

№	Дата	Контест	Место проведения	Кол-во участников	Решено задач	Задач на участника
1	18.09.2014	Codeforces Round 267 Div 2	Дом	1	2	2
2	21.09.2014	Codeforces Round 268 Div 2	Дом	1	2	2
3	22.09.2014	Codeforces Отборочный контест СГАУ на 1/4 ACM-ICPC	Дом	1	1	1
4	25.09.2014	Codeforces Training S02E03	МАИ	3	3	1
5	26.09.2014	Codeforces Round 269 Div 2	Дом	1	1	1
6	28.09.2014	Codeforces Round 270 Div 2	Дом	1	2	2
7	02.10.2014	Codeforces Training S02E04	МАИ	3	1	0.33
8	05.10.2014	XV Открытая Всесибирская Олимпиада по Программированию	МАИ	3	1	0.33
9	09.10.2014	Codeforces Training S02E05	МАИ	3	4	1.33
10	12.10.2014	Codeforces Round 272 Div 2	Дом	1	2	2
11	16.10.2014	Codeforces Round 273 Div 2	Дом	1	2	2
12	17.10.2014	Codeforces Training S02E06	МАИ	3	0	0
13	18.10.2014	Codeforces Тренировка СПбГУ графы и DFS	Дом	3	2	0.66
14	19.10.2014	OpenCup GP of SPb. Div 2	МАИ	3	1	0.33
15	20.10.2014	Codeforces Round 274 Div 2	Дом	1	3	3
16	23.10.2014	Codeforces Самарский Аэрокосмический Лицей тренировка №1	Дом	2	1	0.5
17	23.10.2014	Codeforces ACM, NEERC, Восточный четвертьфинал	Дом	3	4	1.33
18	24.10.2014	Codeforces Round 275 Div 2	Дом	1	1	1
19	25.10.2014	Codeforces ACM, NEERC, Южный четвертьфинал	Дом	3	3	1
20	26.10.2014	ACM-ICPC 1/4 Final	МГУ	3	3	1
21	30.10.2014	Codeforces Training S02E07	МАИ	3	2	0.66
22	01.11.2014	Codeforces Crypto Cup	Дом	3	9	3
23	02.11.2014	OpenCup GP of Siberia Div 2	МАИ	2	3	1.5
24	05.11.2014	Codeforces Round 276 Div 2	Дом	1	3	3
25	06.11.2014	Codeforces Training S02E08	МАИ	3	2	0.66
26	11.11.2014	Codeforces Round 277 Div 2	Дом	1	2	2
27	13.11.2014	Codeforces Training S02E09	МАИ	3	3	1
28	15.11.2014	Codeforces Олимпиада школьников Нижегородской области	Дом	2	3	1.5
29	16.11.2014	OpenCup GP of Central Europe. Div 2	МАИ	3	1	0.33
30	17.11.2014	Codeforces Round 277.5 Div 2	Дом	1	4	4
31	20.11.2014	Codeforces Training S02E10	МАИ	3	3	1
32	23.11.2014	OpenCup GP of Europe Div 2	МАИ	3	5	1.66
33	14.12.2014	OpenCup GP of Peterhof Div 2	МАИ	3	1	0.33

Продолжение таблицы.

№	Дата	Контест	Место проведения	Кол-во участников	Решено задач	Задач на участника
34	01.02.2015	OpenCup GP of Japan Div 2	МАИ	3	4	1.33
35	08.02.2015	OpenCup Northern GP Div 2	МАИ	3	2	0.66
36	15.02.2015	OpenCup GP of Karelia Div 2	МАИ	3	4	1.33
37	22.02.2015	OpenCup GP of Udmurtia Div 2	МАИ	3	4	1.33
38	01.03.2015	OpenCup GP of China Div 2	МАИ	3	1	0.33
39	14.03.2015	VK Cup 2015 Квалификация 2	Дом	1	2	2
40	15.03.2015	OpenCup GP of Tatarstan Div 2	МАИ	3	1	0.33
41	21.03.2015	VK Cup 2015 Раунд 1	Дом	1	1	1
42	28.03.2015	VK Cup 2015 - Уайлд-кард раунд 1	Дом	1	2	2
43	29.03.2015	OpenCup Gp of America Div 2	МАИ	3	4	1.33
44	18.04.2015	Vekua Cup Личный этап	МФТИ-1С	1	1	1
45	19.04.2015	Vekua Cup Командный этап	МФТИ-1С	3	3	1
46	26.04.2015	OpenCup GP of Ural Div 2	МАИ	2	2	1

Итого: 46 контестов, 108 решенных задач. Личный вклад ≈55 задач (командные + личные).

Студент:_____ Якименко А.В. Руководитель:_____ Зайцев В.Е.

2 Журнал по командным контестам

2.1 OpenCup GrandPrix of Ukraine (4 solved) + (алгоритм в F)

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

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

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

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 ($3 \le N \le 10^5$, $3 \le K \le 10^5$), 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 окружностиu; $0 \le X_i \le 10^9$). Гарантируется, что $X_N \mod K = 0$. 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

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define ULL unsigned long long
int main() {
    ULL *array = (ULL *)malloc(15500000 * sizeof(ULL));
    ULL p;
    for (p = 0; p < 15000000; p++) {
        array[p] = 0;
    }
    ULL N, K;
    scanf("%lld %lld", &N, &K);
    ULL i;
    int r = 64;
    for (i = 0; i < N; i++) {
        ULL dist;
        scanf("%lld", &dist);
        array[dist/r] |= (1LL << (dist % r));
    }
    ULL max_length;
    scanf("%lld", &max_length);
    const ULL part_size = max_length / K;
    for (i = 0; i < part_size; i++) {
        ULL index = i / r;
        int shift = i % r;
        bool found = true;
        if (array[index] & (1LL << shift)) {
            ULL j;
            for (j = i + part_size; j < max_length; j += part_size) {
                ULL j_index = j / r;
                int j_shift = j % r;
                if (!(array[j_index] & (1LL << j_shift))) {
                    found = false;
                    break;
                }
            }
            if (found) {
                printf("1\n");
                return 0;
            }
        }
    }
    printf("0\n");
    return 0;
}
```

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 ($1 \le a \le 9$, $0 \le b \le 9$) — 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

АНТОН ????.

Исходный код

```
#!/usr/bin/perl
use bigint;
my($a, $b) = split " ", <>;
my $p = 1;
for my $n (0..1000) {
  if (($a eq $b && $p eq $a) || $p =~ m/^$a.*?$b$/) {
    print $n, "\n";
    exit;
  }
  $p <<= 1;
}
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. Т 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, $1 \le K < N \le 10^9$.

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

```
#include <iostream>
using namespace std;
long gcd (long a, long b) {
    while (b) {
        a %= b;
        swap(a, b);
    }
    return a;
}
int main() {
    long n, k;
    cin >> n >> k;
    cout << gcd(n, k);
    return 0;
}
```

2.2 OpenCup GrandPrix of Japan (5 solved) + DONE

Результаты

36.	MAI #6: Makarov, Rik, Yakimenko	+ 1:31	-	-	-	+ 3:04	-	-	+ 0:13	+ 0:30	+ 0:13	5	333	0%	0.11

Ссылка на контест: 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 $1, 2, \ldots, n$, 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 ($1 \le n \le 10^4$) и m ($1 \le m \le 100$) — 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

Выведите два целых числа — западную и восточную префектуры, между которыми проходит граница (в случае однородности выведите "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

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
typedef struct {
    int left;
    int rigth;
    int sum;
} error;
int main(int argc, const char * argv[]) {
    int n, m;
    cin >> n >> m;
    vector< map<string,int> > jag(n);
    for (int i = 0; i < n; i++) {
        jag[i]["w"] = 0;
        jag[i]["e"] = 0;
    }
    vector<error> errors (n + 1);
    errors[0].left = 0;
    errors[0].rigth = 0;
    errors[0].sum = 0;
    for (int i = 0; i < m; i++) {
        string features;
        cin >> features;
        for (size_t j = 0; j < features.size(); j++) {
            if (features[j] == 'W') {
                jag[j]["w"]++;
                errors[0].rigth++; // !!!!!!
            } else {
                jag[j]["e"]++;
            }
        }
    }
    errors[0].sum = errors[0].left + errors[0].rigth;
    int min_error = errors[0].rigth;
    int min_error_index = -1;
    for (int i = 1; i < n + 1; i++) {
        errors[i].left = 0;
        errors[i].rigth = 0;
        errors[i].sum = 0;
        errors[i].left = errors[i - 1].left + jag[i - 1]["e"];
        errors[i].rigth = errors[i - 1].rigth - jag[i - 1]["w"];
        errors[i].sum = errors[i].left + errors[i].rigth;
        if (errors[i].sum < min_error) {
            min_error = errors[i].sum;
            min_error_index = i;
        }
    }
    if (min_error_index == -1) {
        cout << "0 1" << endl;
    } else {
        cout << min_error_index << " " << min_error_index + 1 << endl;
    }
    return 0;
}
```

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) $(0 \le x \le A-1, 0 \le y \le B-1, 0 \le z \le C-1)$. 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 несвязных 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 $(1 \le A, B, C \le 10^8$, $0 \le N \le \min\{1,000, A \cdot B \cdot C - 1\})$.

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 убирания кубика на заданных координатах. Сложность $O(N^2)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;
#define ll long long
class Point {
public:
  ll x, y, z;
  Point() {}
  Point(ll a, ll b, ll c) {
    x = a;
    y = b;
    z = c;
  }
  void setPoints(ll a, ll b, ll c) {
    x = a;
    y = b;
    z = c;
  }
  bool eql(ll a, ll b, ll c) {
    return a == x && b == y && c == z;
  }
};
int main() {
  ll a, b, c, n;
  cin >> a >> b >> c >> n;
  vector<Point> points(n);
  ll square = 2*(a*b + b*c + a*c);
  for (int i = 0; i < n; ++i) {
    ll x, y, z;
    cin >> x >> y >> z;
    points[i].setPoints(x, y, z);
    ll s = 0;
    if (x == 0) ++s;
    if (x == a-1) ++s;
    if (y == 0) ++s;
    if (y == b-1) ++s;
    if (z == 0) ++s;
    if (z == c-1) ++s;
    ll p = 0;
    for (int j = 0; j < i; ++j) {
      if (points[j].eql(x-1, y, z)) ++p;
      if (points[j].eql(x+1, y, z)) ++p;
      if (points[j].eql(x, y-1, z)) ++p;
      if (points[j].eql(x, y+1, z)) ++p;
      if (points[j].eql(x, y, z-1)) ++p;
      if (points[j].eql(x, y, z+1)) ++p;
    }
    square += 6-2*(s+p);
  }
  cout << square << "\n";
  return 0;
}
```

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 грустный смайликов и вывести в ответе каких больше. Решается поиском подстроки в строке. Сложность $O(N * M)$.

Исходный код

```
str = gets
str = str + "$"
happy = str.split(":-)")
sad = str.split(":-(")
if happy.size > sad.size
  puts "happy"
elsif happy.size < sad.size
  puts "sad"
else
  puts "neutral"
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	opoqenopcadupoq
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

```
#!/usr/bin/perl
my %h = (
     a => "a", b => "bac",c => "cad",e => "e",f => "feg",
     g => "geh",h => "hij",i => "i",j => "jik",k => "kil",
     l => "lim",m => "mon",n => "nop",o => "o",p => "poq",
     q => "qor",r => "ros",s => "sut",t => "tuv",u => "u",
     v => "vuw",w => "wux",x => "xuy",y => "yuz",z => "zuz");
my $a = <>;
chomp $a;
my $res = "";
for my $i (split "", $a) {
  $res .= $h{$i};
}
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 ($1 \le N \le 100$) — 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

```
#!/usr/bin/perl
my %h = (paper => 0, scissors => 1, rock => 2);
my $n = <>;
my $A = <>;
my $B = <>;
chomp $A;
chomp $B;
my @A = split " ", $A;
my @B = split " ", $B;
my ($ac, $bc) = (0, 0);
for (my $i = 0; $i < $n; ++$i) {
  my ($a, $b) = ($A[$i], $B[$i]);
  if ($a eq "paper" && $b eq "rock") {
    ++$ac;
  }
  elsif ($b eq "paper" && $a eq "rock") {
    ++$bc;
  }
  elsif ($h{$a} > $h{$b}) {
    ++$ac;
  }
  elsif ($h{$a} < $h{$b}) {
    ++$bc;
  }
}
print "$ac $bc\n";
```

2.3 OpenCup GrandPrix of Eurasia (4 solved) + (алгоритм в 9, 12)

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

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 ($1 \le N \le 18$).

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 ($1 \le Q \le 10^5$).

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

```
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <fstream>
using namespace std;
typedef struct {
    string name;
    vector<string> losers;
} Team;
int main(int argc, const char * argv[]) {
    ifstream in("input.txt");
    ofstream out("output.txt");
    int n;
    in >> n;
    int t = 1;
    for (int i = 0; i < n; i++) {
        t *= 2;
    }
    map<int, vector<int> > teams; // first=team_name, second=losers_vector
    map<string, int> index_for_name;
    vector<pair<int, int> > pairs;
    vector<pair<int, int> > prev_pairs;
    vector<char> log(t - 1);
    for (int i = 0; i < t; i = i + 2) {
        string team_name_1, team_name_2;
        in >> team_name_1 >> team_name_2;
        index_for_name[team_name_1] = i;
        index_for_name[team_name_2] = i + 1;
        teams[i] = vector<int> ();
        teams[i + 1] = vector<int> ();
        pairs.push_back(make_pair(i, i + 1));
    }
    in.get();
    for (int i = 0; i < t - 1; i++) {
        char result = in.get();
        log[i] = result;
    }
    in.get();
    prev_pairs = pairs;
    pairs.clear();
    bool end = false;
    while (!end) {
        for (int i = 0; i < t / 2; i += 2) {
            int first_winner, first_looser;
            int second_winner, second_looser;
            if (log[i] == 'W') {
                first_winner = prev_pairs[i].first;
                first_looser = prev_pairs[i].second;
                teams[first_winner].push_back(first_looser);
                teams[first_winner].insert(teams[first_winner].end(), teams[first_looser].begin(), teams[first_looser].end());
            } else {
                first_winner = prev_pairs[i].second;
                first_looser = prev_pairs[i].first;
                teams[first_winner].push_back(first_looser);
                teams[first_winner].insert(teams[first_winner].end(), teams[first_looser].begin(), teams[first_looser].end());
```

```
            }
            if (i == t - 2) {
                end = true;
                break;
            }
            if (log[i + 1] == 'W') {
                second_winner = prev_pairs[i + 1].first;
                second_looser = prev_pairs[i + 1].second;
                teams[second_winner].push_back(second_looser);
                teams[second_winner].insert(teams[second_winner].end(), teams[second_looser].begin(), teams[second_looser].end());
            } else {
                second_winner = prev_pairs[i + 1].second;
                second_looser = prev_pairs[i + 1].first;
                teams[second_winner].push_back(second_looser);
                teams[second_winner].insert(teams[second_winner].end(), teams[second_looser].begin(), teams[second_looser].end());
            }
            pairs.push_back(make_pair(first_winner, second_winner));
        }
        log.erase(log.begin(), log.begin() + t / 2);
        prev_pairs = pairs;
        pairs.clear();
        t /= 2;
    }
    int q;
    in >> q;
    for (int i = 0; i < q; i++) {
        string t1, t2;
        in >> t1 >> t2;
        int index_1 = index_for_name[t1];
        int index_2 = index_for_name[t2];
        if (find(teams[index_1].begin(), teams[index_1].end(), index_2) != teams[index_1].end()) {
            out << "Win" << endl;
        } else if (find(teams[index_2].begin(), teams[index_2].end(), index_1) != teams[index_2].end()) {
            out << "Lose" << endl;
        } else {
            out << "Unknown" << endl;
        }
    }
    in.close();
    out.clear();
    return 0;
}
```

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 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 << b` — операция сдвига битов числа a на b разрядов влево. При этом младшие b разрядов числа заполняются нулями, а старшие b битов отбрасываются.
- `a >> 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 обращать 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 614278301 1228622139 1841720774 2457244278	1 2 3 4

Алгоритм

АНТОН ????.

Исходный код

```
#include <iostream>
#include <cinttypes>
using namespace std;
#define ll long long
#define ull unsigned long long

uint32_t dehash(uint32_t value) {
    uint32_t hash = value;
    hash = hash - (hash<<16);
    uint32_t t = hash;
  hash = (hash xor (hash >> 11)) & ~((1ULL<<10)-1);
  for (int i = 0; i < 1023; ++i) {
    if ((hash xor (hash >> 11)) == t) {
      break;
    }
    ++hash;
  }
  ull h = hash;
  for (ull i = 0;; ++i) {
    if (((h|(i<<32)) % 9) == 0) {
      hash = (h|(i<<32))/9;
      break;
    }
  }
  t = hash;
  hash = (hash xor (hash >> 6)) & ~((1ULL<<20)-1);
  for (int i = 0; i < 1048575; ++i) {
    if ((hash xor (hash >> 6)) == t) {
      break;
    }
    ++hash;
  }
  h = hash;
  for (ull i = 0;; ++i) {
    if (((h|(i<<32)) % 1025) == 0) {
      hash = (h|(i<<32))/1025;
      break;
    }
  }
    return hash;
}

int main() {
  uint32_t Q;
  cin >> Q;
  for (int i = 0; i < Q; ++i) {
    ull d;
    cin >> d;
    cout << dehash(d) << "\n";
  }
  return 0;
}
```

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 ($1 \le N \le 10^5$, $1 \le S \le 10^6$).

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

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

```
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>

using namespace std;

bool pairMinMax(const pair<int, int> &a, const pair<int, int> &b) {
  return a.second < b.second;
}


int main() {

  ifstream in("input.txt");
  ofstream out("output.txt");


  int N;
  long long S;

  in >> N >> S;

  vector <pair <int, int> > element(N);
  vector <bool> bought(N+1, false);

  pair <int, int> temp;
  for (int i = 0; i < N; i++) {
    in >> temp.first >> temp.second;
    element[i] = temp;
  }

  sort(element.begin(), element.end(), pairMinMax);

  int count = 0;
  for (int i = 0; i < N; i++) {
    if (element[i].second > S) {
      break;
    }
    if (element[i].second <= S && bought[element[i].first] == false) {
      S -= element[i].second;
      bought[element[i].first] = true;
      count++;
    }
  }

  out << count << endl;

  in.close();
  out.close();

  return 0;
}
```

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 15×15.

В начале игры каждый получает по 7 фишек с буквам 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 фишки, 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 фишки, то есть, если это не первое слово, которое составлено за всю игру. Назовём фишками, образующими слово, все фишки, которые игрок поместил на поле, и те, на которые он указал. Фишки, образующие слово, должны находиться подряд либо на одной вертикали, либо на одной горизонтали. Если эти фишки 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 фишки, 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 фишками, 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. Дана 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 ($2 \le N \le 4$, $1 \le M \le 130$).

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

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 ($2 \le L \le 15$, $1 \le X, Y \le 15$). 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 совершённых ходов.

Пример

input.txt	output.txt
2 6 2 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	121 102

Алгоритм

АНТОН ????.

Исходный код

```
#!/usr/bin/perl
use bigint;
my @cmap = qw(   rwwgwwwrwwwgwwr
            wbwwwywwwywwwbw
            wwbwwwgwgwwwbww
            gwwbwwwgwwwbwwg
            wwwwbwwwwwbwwww
            wywwwwwwwwwwwyw
            wwgwwwgwgwwwgww
            rwwgwwwswwwgwwr
            wwgwwwgwgwwwgww
            wywwwwwwwwwwwyw
            wwwwbwwwwwbwwww
            gwwbwwwgwwwbwwg
            wwbwwwgwgwwwbww
            wbwwwywwwywwwbw
            rwwgwwwrwwwgwwr);
my @map;
for (my $i = 0; $i < 16; ++$i) {
  for (my $j = 0; $j < 16; ++$j) {
    $map[$i][$j] = 0;
  }
}
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);

my %cl =
    (w => 1,
     s => 1,
     g => 2,
     y => 3,
     b => 1,
     r => 1);

my %cw =
    (w => 1,
     s => 1,
     g => 1,
     y => 1,
     b => 2,
     r => 3);

sub getColor {
  my ($x, $y) = @_;
  substr $cmap[$y-1], $x-1, 1;
}
my $str;
$str = <ARGV>;
chomp $str;
my ($n, $m) = split(" ", $str);
my @scores;
for (my $i = 0; $i < $n; ++$i) {
  $scores[$i] = 0;
}
for (my $si = 0; $si < $m; ++$si) {
  my $str = <ARGV>;
  chomp $str;
  my $w = $str;
  my $count = 0;
```

```
  my $score = 0;
  for (my $i = 0; $i < $w; ++$i) {
    $str = <ARGV>;
    chomp $str;
    my @A = split(" ", $str);
    my $l = shift @A;
    my $dir = shift @A;
    my $x = shift @A;
    my $y = shift @A;
    my $scoreL = 0;
    my $scoreW = 1;
    for my $char (@A) {
      my $color = getColor($x, $y);
      unless ($map[$x][$y]) {
        $map[$x][$y] = 1;
        ++$count;
      }

      $scoreL += $cl{$color} * $alp[$char-1];
      $scoreW *= $cw{$color};
      if ($dir eq 'v') {
        ++$y;
      }
      else {
        ++$x;
      }
    }
    $score += $scoreW * $scoreL;
  }
  if ($count == 7) {
    $score += 15;
  }
  $scores[$si%$n] += $score;
}
print join("\n", @scores), "\n";
```

2.4 OpenCup GrandPrix of SPb (2 solved) + (алгоритм в E)

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

Задача E - 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 ($K \leqslant 10^5$, $K \leqslant N$, $1 \leqslant N \leqslant 10^9$). Во второй строке даны K целых чисел A_i — слагаемые разбиения ($1 \leqslant A_i \leqslant N$).

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

АНТОН ????.

Исходный код

```
#include <iostream>
using namespace std;
bool pairMinMax(const pair<int, int> &a, const pair<int, int> &b) {
  return a.second < b.second;
}
int main() {
  int input;
  ifstream in("next-partition.in");
  ofstream out("next-partition.out");
  int K;
  in >> K;
  vector <int> A(K);
  int N = 0;
  int temp;
  for (int i = 0; i < K; i++) {
    in >> temp;
    A[i] = temp;
    N += temp;
  }
  bool flag1 = false;
  bool flag2 = false;
  int sum = 0;
  sort(A.begin(), A.end());
  int iMarked = 0;
  for (int i = 0; i < K - 1; i++) {
    sum += A[i];
    if (A[i + 1] - A[i] >= 2) {
      iMarked = i + 1;
```

```
      A[i + 1]--;
      sum++;
      flag1 = true;
      break;
    }
  }
  if (!flag1) {
    for (int i = 0; i < K - 2; i++) {
      for (int j = i + 2; j < K; j++) {
        if (A[i] + 1 > A[i + 1]) {
          break;
        }
        if (A[j] - A[i] > 2) {
          break;
        }
        if (A[j] - A[i] == 2 && A[j] - 1 >= A[j - 1]) {
          A[j]--;
          A[i]++;
          flag2 = true;
          break;
        }
      }
    }
  }
  if (flag1) {
    sum -= iMarked;
    for (int i = iMarked - 1; i >= 0; i--) {
      if (sum > 0) {
        temp = A[iMarked];
        if (sum+1 >= temp) {
          A[i] = temp;
          sum = sum - temp + 1;
        }
        else {
          A[i] = sum + 1;
          sum = 0;
        }
      }
      else A[i] = 1;
    }
    out << K << endl;
    for (int i = K - 1; i >= 0; i--) {
      out << A[i] << " ";
    }
  }
  else if (flag2) {
    out << K << endl;
    for (int i = K - 1; i >= 0; i--) {
      out << A[i] << " ";
    }
  } else {
    out << -1;
  }
  in.close();
  out.close();
  return 0;
}
```

Задача G - Головоломка Обмен

Головоломка «Обмен» выглядит как доска 4×4, 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 05 10 13 08 09 12 11 06 07 14 15 16	4 b2-c2 a1-a2 c4-b2 b3-d1

Алгоритм

Так как доска всегда имеет размеры 4x4, 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

```
#include <iostream>
#include <sstream>
#include <vector>
#include <fstream>

using namespace std;

bool desk_is_correct(int desk[4][4])
{
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            int targer_i = (desk[i][j] - 1) / 4;
            int targer_j = (desk[i][j] - 1) % 4;
            if (i != targer_i || j != targer_j) {
                return false;
            }
        }
    }
    return true;
}

char character_for_num(int num)
{
    return char('a' + num);
}

int main(int argc, const char * argv[])
{
    stringstream ss;

    ifstream in("puzzle-swap.in");
    ofstream out("puzzle-swap.out");

    int desk[4][4] = {0};

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            in >> desk[i][j];
        }
    }

    int moves = 0;
    vector<string> log;

    // find all wrong pairs
    for (int i1 = 0; i1 < 4; i1++) {
        for (int j1 = 0; j1 < 4; j1++) {
            for (int i2 = 0; i2 < 4; i2++) {
                for (int j2 = 0; j2 < 4; j2++) {
                    if (i1 != i2 || j1 != j2) {
                        int targer_i1 = (desk[i1][j1] - 1) / 4;
                        int targer_j1 = (desk[i1][j1] - 1) % 4;
                        int targer_i2 = (desk[i2][j2] - 1) / 4;
                        int targer_j2 = (desk[i2][j2] - 1) % 4;
                        if (targer_i1 == i2 && targer_j1 == j2 &&
                            targer_i2 == i1 && targer_j2 == j1) {
                            moves++;
                            swap(desk[i1][j1], desk[i2][j2]);
```

```
                                ss << character_for_num(i1) << j1 + 1 << "-" << character_for_num(i2) << j2 + 1 << endl;
                                string s;
                                ss >> s;
                                log.push_back(s);
                            }
                        }
                    }
                }
            }
        }

    while (!desk_is_correct(desk)) {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                int targer_i = (desk[i][j] - 1) / 4;
                int targer_j = (desk[i][j] - 1) % 4;
                if (i != targer_i || j != targer_j) {
                    moves++;
                    swap(desk[i][j], desk[targer_i][targer_j]);
                    ss << character_for_num(i) << j + 1 << "-" << character_for_num(targer_i) << targer_j + 1 << endl;
                    string s;
                    ss >> s;
                    log.push_back(s);
                }
            }
        }
    }

    out << moves << endl;
    for (size_t i = 0; i < log.size(); i++) {
        out << log[i] << endl;
    }

    in.close();
    out.close();

    return 0;
}
```

2.5 ACM-ICPC Московский четвертьфинал + DONE

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

2.6 OpenCup GrandPrix of Yekaterinburg (4 solved) + (алгоритм в B)

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

Задача B - 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 оснащён 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 ($1 \leqslant N \leqslant 1000$ — 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 ($1 \leqslant X \leqslant 10^9$ — 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

АНТОН ????.

Исходный код

```
#include <iostream>
#include <set>
#include <vector>
using namespace std;
#define ll long long
#define ull unsigned long long

int main() {
  ull n;
  cin >> n;
  set<ull> v;
  vector<ull> m;
  ull maxV = 0;
  ull a = 1;
  for (int i = 0; i < n; ++i) {
    ull d;
    cin >> d;
    if (d > maxV)
      maxV = d;
    v.insert(d);
    m.push_back(d);
  }
  bool f = 1;
  while (1) {
    ++a;
    bool f = 1;
    for (int i = 0; i < n; ++i) {
      if (m[i] >= a && m[i]%a == 0) {
        f = 0;
        break;
      }
    }
    if (f) {
      break;
    }
  }
  while (1) {
    f = 1;
    for (ull x = 0; x < maxV+1; x += a) {
      if (v.find(x) != v.end()) {
        f = 0;
        break;
      }
    }
    if (f) {
      cout << a << endl;
      break;
    }
    while (1) {
      ++a;
      bool f = 1;
      for (int i = 0; i < n; ++i) {
        if (m[i] >= a && m[i]%a == 0) {
          f = 0;
          break;
        }
      }
      if (f) {
```

```
            break;
        }
    }
  }
  return 0;
}
```

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. ($1 \leqslant N \leqslant 10^5$)

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

```
n = gets.to_i
m = (1e+9 + 7).to_i

amount = Array.new(1e+5 + 1)
sum_for_pos = Array.new(1e+5 + 1)

k = 5

amount[1] = 1
amount[2] = 2
amount[3] = 2
amount[4] = 4

```

```
sum_for_pos[1] = 1
sum_for_pos[2] = 3
sum_for_pos[3] = 5
sum_for_pos[4] = 9

while k <= n do
    max_next_k = k / 2
    amount[k] = (1 + sum_for_pos[max_next_k])
    sum_for_pos[k] = sum_for_pos[k - 1] + amount[k]
    k += 1
end

answer = (amount[n] % m)

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!!!!!!	Wooooooow

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

```
#!/usr/bin/perl
my $str =  <>;
my @a = $str =~ /\!/g;
my $n = scalar(@a);
if ($n == 0) {
  print "Pfff";
}
else {
  print "W"."o"x$n."w";
}
```

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

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

2.7 OpenCup GrandPrix of Siberia (6 solved) + (алгоритм в B, K)

Результаты

37.	MAI #6: Makarov, Rik, Yakimenko	+ 0:08	+ 2:21	-	+ 1:24	-25 4:59	-	+ 0:15	-	-	-	+ 1:54	+2 1:04	6	467	25%	0.15

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10281

Задача A - 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 reguar 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

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

Задача B - Painting Tracks

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

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 ($1 \le N, M \le 10^{17}$, $|N - M| \le 55$).

Output

Print one integer — number of different paintings.

Examples

standard input	standard output
2 2	6
3 1	12

Алгоритм

АНТОН ????.

Исходный код

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <cmath>
#include <set>

#define LL long long
#define ULL  unsigned long long


using namespace std;

ULL myAbs(ULL a, ULL b) {
  if(a > b) {
    return a - b;
  } else {
    return b - a + 1;
  }
}

```

```
int main() {

  int temp;
  ULL N, M;
  cin >> N >> M;
  if(N == M) {
    cout << 6 << endl;
    return 0;
  }
  else {
    ULL answ = 3;
    for(int i = 0; i < myAbs(N, M); i++) {
      answ *= 2;
    }
    cout << answ << endl;
  }

  //cin >> temp;

  return 0;

}
```

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 ($1 \le N \le 2 \cdot 10^5$). Second line contains N integers G_i ($1 \le G_i \le 10^5$) — strengths of GTU teams. Third line contains N integers T_i ($1 \le T_i \le 10^5$) — strengths of TSU teams. It is guaranteed that $T_i \ne G_j$ for any $1 \le i, j \le N$.

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

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>

using namespace std;

int main(int argc, const char * argv[]) {

    int n;
    cin >> n;

    vector<int> gtu(n);
    vector<int> tsu(n);

    int val;

    for (int i = 0; i < n; i++) {
        cin >> val;
        gtu[i] = val;
    }

    for (int i = 0; i < n; i++) {
        cin >> val;
        tsu[i] = val;
    }

    sort(gtu.begin(), gtu.end(), greater<int>());
    sort(tsu.begin(), tsu.end(), greater<int>());

    int gtu_score = 0;
    int tsu_score = 0;

    vector<int> tsu_search = tsu;

    int next_start_pos = 0;

    for (int i = 0; i < n; i++) {
        bool found = false;
        for (int j = next_start_pos; j < n; j++) {
            if (gtu[i] > tsu_search[j]) {
                gtu_score++;
                tsu_search[j] = INT_MAX;
                found = true;
                next_start_pos = j + 1;
                break;
            }
        }
        if (!found) {
            break;
        }
    }

    next_start_pos = 0;

    for (int i = 0; i < n; i++) {
        bool found = false;
        for (int j = next_start_pos; j < n; j++) {
```

```
            if (tsu[i] > gtu[j]) {
                tsu_score++;
                gtu[j] = INT_MAX;
                found = true;
                next_start_pos = j + 1;
                break;
            }
        }
        if (!found) {
            break;
        }
    }

    cout << gtu_score << " " << tsu_score << endl;

    return 0;
}
```

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

```
#!/usr/bin/perl
<>;
my %h;
my $i;
my @keys;
while (my $s = <>) {
  chomp $s;
  my ($suf) = $s =~ /\.(.+?)$/;
  push @keys, $suf unless exists $h{$suf};
  ++$h{$suf};
}
foreach my $key (@keys) {
  print $key, ": ", $h{$key}, "\n";
}
```

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 ($1 \le N \le 10^4; |X|, |Y| \le 10^4$). 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 ($|x|, |y| \le 10^4$). 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 4 1 4 2	5 0 6 0
1 0 0 1 1	2 -1

Алгоритм

АНТОН ????.

Исходный код

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

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 ($1 \le N, K, M \le 10^5; K \le N$). 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 ($1 \le a, b \le N$).

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 3 1 1 2 3 3 2 2 1 3 1	1 2 2 2 2

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

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
int main(int argc, const char * argv[]) {
    int n, k, m;
    cin >> n >> k >> m;
    vector< vector<int> > g(n + 1, vector<int>());
    vector<bool> side_effect(n + 1, false);
    int func_num;
    for (int i = 0; i < k; i++) {
        cin >> func_num;
        side_effect[func_num] = true;
    }
    int a, b;
    int count = k;
    for (int i = 0; i < m; i++) {
        cin >> a >> b;
        g[b].push_back(a);
        if (!side_effect[a] && side_effect[b]) {
            side_effect[a] = true;
            count++;
            queue<int> q;
            q.push(a);
            vector<bool> used(n + 1);
            used[a] = true;
            while (!q.empty()) {
                int v = q.front();
                q.pop();
                for (size_t i = 0; i < g[v].size(); ++i) {
                    int to = g[v][i];
                    if (!used[to]) {
                        used[to] = true;
                        if (!side_effect[to]) {
                            side_effect[to] = true;
                            count++;
                            q.push(to);
                        }

                    }
                }
            }
        }
        cout << count << endl;
    }
    return 0;
}
```

2.8 OpenCup GrandPrix of Europe (3 solved) + DONE

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

Задача A - 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 7×5, 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 folows:

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

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

Алгоритм

Задача просто на реализацию.

Исходный код

```
my %digits = (
  "xxxxx".
  "x...x".
  "x...x".
  "x...x".
  "x...x".
  "x...x".
  "xxxxx" => 0,
  "....x".
  "....x".
  "....x".
  "....x".
  "....x".
  "....x".
  "....x" => 1,

  "xxxxx".
  "....x".
  "....x".
  "xxxxx".
  "x....".
  "x....".
  "xxxxx" => 2,

  "xxxxx".
  "....x".
  "....x".
  "xxxxx".
  "....x".
  "....x".
  "xxxxx" => 3,

  "x...x".
  "x...x".
  "x...x".
  "xxxxx".
  "....x".
  "....x".
  "....x" => 4,

  "xxxxx".
  "x....".
  "x....".
  "xxxxx".
  "....x".
  "....x".
  "xxxxx" => 5,

  "xxxxx".
  "x....".
  "x....".
  "xxxxx".
  "x...x".
  "x...x".
  "xxxxx" => 6,

  "xxxxx".
  "....x".
```

```
  "....x".
  "....x".
  "....x".
  "....x".
  "....x" => 7,

  "xxxxx".
  "x...x".
  "x...x".
  "xxxxx".
  "x...x".
  "x...x".
  "xxxxx" => 8,

  "xxxxx".
  "x...x".
  "x...x".
  "xxxxx".
  "....x".
  "....x".
  "xxxxx" => 9,

  ".....".
  "..x..".
  "..x..".
  "xxxxx".
  "..x..".
  "..x..".
  "....." => "+"
  );
my %dig;
foreach my $key (keys %digits) {
  $dig{$digits{$key}} = $key;
}
my @screen;
for (my $i = 0; $i < 7; ++$i) {
  my $str = <>;
  chomp $str;
  my $n = length($str)+1;
  my $dig_cnt = $n/6;
  for (my $j = 0; $j < $dig_cnt; ++$j) {
    my $a = substr substr($str, $j*6, 6), 0, 5;
    $screen[$j] .= $a;
  }
}
my $expr = "";
foreach my $s (@screen) {
  $expr .= $digits{$s};
}
my $res = eval $expr;
for (my $i = 0; $i < 7; ++$i) {
  my $str = "";
  foreach my $d (split "", $res) {
    my $a = substr $dig{$d}, $i*5, 5;
    $str .= $a.".";
  }
  chop $str;
  print $str, "\n";
}
```

Задача C - 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 городо 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 командировки. 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 7 zagreb krakow warsaw krakow prague zagreb krakow 3 barnaul barnaul barnaul	4 1

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

```
#include <iostream>
#include <set>
#include <string>

using namespace std;

int main(int argc, const char * argv[]) {

    int t;
    cin >> t;
    set<string> s;
    while (t--) {
        int n;
        cin >> n;
        string name;
        for (int i = 0; i < n; i++) {
            cin >> name;
            s.insert(name);
        }
        cout << s.size() << endl;
        s.clear();
    }

    return 0;
}
```

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 прмера 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 3 10 21 10 3 20 10 10 3 10 10 10 4 15 15 15 45	Victory 2 Leader 1 Tie Leader 4

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

```
#include <iostream>
#include <algorithm>
#include <vector>
typedef struct {
    int number;
    int votes;
} Person;
bool cmp(const Person &a, const Person &b) {
    return a.votes > b.votes;
}
using namespace std;
int main(int argc, const char * argv[]) {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        int total_votes = 0;
        vector<Person> persons(n);
        for (int i = 0; i < n; i++) {
            persons[i].number = i + 1;
            cin >> persons[i].votes;
            total_votes += persons[i].votes;
        }
        sort(persons.begin(), persons.end(), cmp);
        int top_votes = persons[0].votes;
        int half_votes = total_votes / 2;
        if (t == 1) {
            cout << " " << endl;
        }
        if (top_votes > half_votes) {
            if (persons[1].votes != top_votes) {
                cout << "Victory " << persons[0].number << endl;
            } else {
                cout << "Tie" << endl;
            }
        } else {
            if (persons[1].votes != top_votes) {
                cout << "Leader " << persons[0].number << endl;
            } else {
                cout << "Tie" << endl;
            }
        }
    }
    return 0;
}
```

2.9 OpenCup GrandPrix of Peterhof (4 solved) + (алгоритм в A, E, G)

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

Задача A - (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 $2^{3\cdot3\cdot3\cdot3} = 2^{81} = 2\,417\,851\,639\,229\,258\,349\,412\,352$.

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

$$a^{(a+1)^{\cdot^{\cdot^{\cdot^{(b-1)^b}}}}}.$$

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 ($1 \leqslant a \leqslant b \leqslant 10$).

Формат выходных данных

Выведите одно число — последнюю десятичную цифру числа

$$a^{(a+1)^{\cdot^{\cdot^{\cdot^{(b-1)^b}}}}}.$$

Примеры

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

Алгоритм

АНТОН ????.

Исходный код

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

Задача C - 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 ($2 \leqslant n \leqslant 4096$). 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 подрядке. 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 0101010101 101010110101011 0101 0000 111	4 1 2 3 4 5

Алгоритм

Задача на реализацию. Необходимо найти классы эквивалентности.

Исходный код

```
n = gets.to_i

data = Array.new

for i in 0...n
    s = gets.chomp
    data.push(s)
end

data.each do |s|
    begin
        s_copy = s.clone
        s.gsub!(/0{2}/,'')
```

```
        s.gsub!(/1{3}/,'')
    end until s == s_copy
end

h = Hash.new()

data.each_index do |i|
    s = data[i]
    if h[s] == nil
        h[s] = Array.new
    end
    h[s].push(i + 1)
end

res = Array.new

h.each_value do |v|
    res.push(v)
end

res.sort! do |a, b|
    a.first <=> b.first
end

puts "#{res.size}"

res.each do |l|
    l.each do |x|
        print "#{x} "
    end
    puts
end
```

Задача E - 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 команд 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 ($1 \leqslant N \leqslant 100$). Вторая строка содержит N целых чисел r_i ($-10^6 \leqslant r_i \leqslant 10^6$). Третья строка содержит N целых чисел c_i ($-10^6 \leqslant c_i \leqslant 10^6$). 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 $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-9}$.

Пример

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

Алгоритм

АНТОН ????.

Исходный код

```
#!/usr/bin/env ruby
def calc_sum(m, i)
  s = 0
  for a in m do
    s += Math.sqrt((a-i)*(a-i) + 1)
    i += 1
  end
  return s
end
n = gets.to_i
r = gets.split.map{|x| x.to_i}
c = gets.split.map{|x| x.to_i}
m = (r + c).sort
x = 0
step = 1000000
sum_curr = calc_sum(m, x)
sum_prev = sum_curr + 1
sign = 1
x += step
while ((sum_prev-sum_curr).abs > 0.0000000001)
  sum_prev = sum_curr
  sum_curr = calc_sum(m, x)
  if sum_curr > sum_prev && sign == 1
    sign = -1
    step /= 2.0
    sign_counter = 0
  elsif sum_curr > sum_prev && sign == -1
     sign = 1
     step /= 2.0
     sign_counter = 0
  end
  x += step*sign
end
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 тесте 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

АНТОН ????.

Исходный код

```
#!/usr/bin/env ruby
def send_cmd(cmd)
  puts cmd
  STDOUT.flush
end
a = 1
while (true) do
  send_cmd a
  case gets
  when /yes/
    exit if a == 1
    a = 1
  when /no/
    a = rand(1<<31)
  end
end
```

2.10 Codeforces ACM-ICPC Южный четвертьфинал

Результаты

Задачи

№	Название				
A	Nasta Rabbara	стандартный ввод/вывод 10 с, 512 МБ			x41
B	Colored Blankets	стандартный ввод/вывод 1 с, 512 МБ			x167
C	Component Tree	стандартный ввод/вывод 6 с, 512 МБ			x148
D	Data Center	стандартный ввод/вывод 2 с, 512 МБ			x682
E	Election of a Mayor	стандартный ввод/вывод 2 с, 512 МБ			x452
F	Ilya Muromets	стандартный ввод/вывод 1 с, 512 МБ			x745
G	FacePalm Accounting	стандартный ввод/вывод 1 с, 512 МБ			x553
H	Minimal Agapov Code	стандартный ввод/вывод 8 с, 512 МБ			x18
I	Sale in GameStore	стандартный ввод/вывод 2 с, 512 МБ			x1312
J	Getting Ready for VIPC	стандартный ввод/вывод 1,5 с, 512 МБ			x28
K	Treeland	стандартный ввод/вывод 2 с, 512 МБ			x304
L	Useful Roads	стандартный ввод/вывод 4 с, 512 МБ			x10
M	Variable Shadowing	стандартный ввод/вывод 2 с, 512 МБ			x649

Ссылка на контест: http://codeforces.com/gym/100513

Задача D - Data Center

The startup "Booble" has shown explosive growth and now it needs a new data center with the capacity of m petabytes. Booble can buy servers, there are n servers available for purchase: they have equal price but different capacities. The i-th server can store a_i petabytes of data. Also they have different energy consumption — some servers are *low voltage* and other servers are not.

Booble wants to buy the minimum number of servers with the total capacity of at least m petabytes. If there are many ways to do it Booble wants to choose a way to maximize the number of *low voltage* servers. Booble doesn't care about exact total capacity, the only requirement is to make it at least m petabytes.

Input

The first line contains two integer numbers n and m ($1 \le n \le 2 \cdot 10^5, 1 \le m \le 2 \cdot 10^{15}$) — the number of servers and the required total capacity.

The following n lines describe the servers, one server per line. The i-th line contains two integers a_i, l_i ($1 \le a_i \le 10^{10}, 0 \le l_i \le 1$), where a_i is the capacity, $l_i = 1$ if server is *low voltage* and $l_i = 0$ in the opposite case.

It is guaranteed that the sum of all a_i is at least m.

Output

Print two integers r and w on the first line — the minimum number of servers needed to satisfy the capacity requirement and maximum number of *low voltage* servers that can be bought in an optimal r servers set.

Print on the second line r distinct integers between 1 and n — the indices of servers to buy. You may print the indices in any order. If there are many solutions, print any of them.

Examples

standard input	standard output
4 10 3 1 7 0 5 1 4 1	2 1 4 2
3 13 6 1 6 1 6 1	3 3 1 2 3

Алгоритм

Сначала отсортируем сервера по объему памяти. Наберем необходимое количество серверов и сравним набранную память с минимальным объемом. Если разница равна 0, то ответ найден. Иначе будем пытаться поменять сервера с высоким напряжением на сервера с низким напряжением. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <fstream>

using namespace std;

typedef struct {
    LL num;
    ULL cap;
    short low;
} server;

bool cmp_low(const server &a, const server &b) {
    return a.low > b.low;
}

bool cmp_cap(const server &a, const server &b) {
    if (a.cap == b.cap) return a.low > b.low;
    return a.cap > b.cap;
}

int main() {
    LL n;
    ULL m;
    cin >> n >> m;
    vector<server> s(n);
    for (LL i = 0; i < n; i++) {
        s[i].num = i + 1;
        cin >> s[i].cap;
        cin >> s[i].low;
    }

    sort(s.begin(), s.end(), cmp_cap);
    LL ind = 0;
    ULL curr_cap = 0;
    LL count = 0;

    while (curr_cap < m) {
        curr_cap += s[ind].cap;
        if (s[ind].low) count++;
        ind++;
    }

    ULL rem = curr_cap - m;

    if (rem == 0) {
        cout << ind << " ";
        cout << count << endl;
        for (LL i = 0; i < ind; i++) {
            cout << s[i].num << " ";
        }
        cout << endl;
        return 0;
    }

    for (LL i = ind - 1; i >= 0; i--) {
        if (!s[i].low && rem > 0) {
```

```
            bool azaza = false;
            for (LL j = ind; j < n; j++) {
                if (s[j].low && (s[j].cap + rem >= s[i].cap)) {
                    rem -= s[i].cap - s[j].cap;
                    swap(s[i], s[j]);
                    count++;
                    azaza = true;
                    break;
                }
            }
            if (!azaza) break;
        }
        else if (rem == 0) break;
    }

    cout << ind << " " << count << endl;
    for (LL i = 0; i < ind; i++) {
        cout << s[i].num << " ";
    }
    cout << endl;;
  return 0;
}
```

Задача I - Sales in GameStore

A well-known Berland online games store has announced a great sale! Buy any game today, and you can download more games for free! The only constraint is that the total price of the games downloaded for free can't exceed the price of the bought game.

When Polycarp found out about the sale, he remembered that his friends promised him to cover any single purchase in GameStore. They presented their promise as a gift for Polycarp's birthday.

There are n games in GameStore, the price of the i-th game is p_i. What is the maximum number of games Polycarp can get today, if his friends agree to cover the expenses for any single purchase in GameStore?

Input

The first line of the input contains a single integer number n ($1 \le n \le 2000$) — the number of games in GameStore. The second line contains n integer numbers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le 10^5$), where p_i is the price of the i-th game.

Output

Print the maximum number of games Polycarp can get today.

Examples

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

Алгоритм

Отсортируем цены по возрастанию. Затем будем складывать цены по порядку из возрастания и будем считать количество сложений, пока сумма не станет больше максимальной цены. В ответе выведем значение счётчика сложений. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
bool compare(const int &a, const int &b) {
    return a<b;
}
int main(int argc, const char * argv[]) {
    vector<int> p(2001);
    int n;
    cin >> n;
    for(int i=0; i<n; ++i)
        cin >> p[i];
    sort(p.begin(), p.begin()+n);
    int sum = 0;
    int i=0;
    while(i<n-1 && sum+p[i]<=p[n-1])
        sum += p[i++];
    cout << i+1;
    return 0;
}
```

Задача M - Variable Shadowing

In computer programming, *variable shadowing* occurs when a variable declared within a certain scope has the same name as a variable declared in an outer scope. The outer variable is said to be shadowed by the inner variable, and this can lead to a confusion. If multiple outer scopes contain variables with the same name, the variable in the nearest scope will be shadowed.

Formally, a declared variable shadows another declared variable if the following conditions are met simultaneously:

- the other variable is declared in outer scope and before (in terms of position in program source code) the declaration of the first variable,
- the other variable is nearest among all variables satisfying the condition above.

Here is an example containing exactly one variable shadowing:

```
/* Prints a+max(b,c) */
int main() {
    int a, b, c;
    cin >> a >> b >> c;
    if (b > c) {
        int a = b; // <-- variable 'a' shadows outer 'a'
        int x = c;
        b = x;
        c = a;
    }
    int x = a + c; // <-- no shadowing here
    cout << x << endl;
}
```

Variable shadowing is permitted in many modern programming languages including C++, but compilers can warn a programmer about variable shadowing to avoid possible mistakes in a code.

Consider a trivial programming language that consists only of scopes and variable declarations. The program consists of lines, each line contains only characters '{', '}', 'a' ... 'z' separated by one or more spaces.

- *Scopes.* A scope (excluding global) is bounded with a pair of matching curly brackets '{' and '}'. A scope is an inner scope relative to another scope if brackets of the first scope are enclosed by brackets of the second scope.
- *Variables.* A variable declaration in this language is written just as a name of the variable. In addition all variables are lowercase Latin letters from 'a' to 'z' inclusive (so there are at most 26 variable names). A variable is declared in each scope at most once.

Given a syntactically correct program (i.e. curly brackets form a *regular bracket sequence*), write an analyzer to warn about each fact of variable shadowing. Warnings should include exact positions of shadowing and shadowed variables. Your output should follow the format shown in the examples below.

Input

The first line contains integer n ($1 \le n \le 50$) — the number of lines in the program. The following n lines contain the program. Each program line consists of tokens '{', '}', 'a' ... 'z' separated by one or more spaces. The length of each line is between 1 and 50 characters. Each program line contains at least one non-space character.

The curly brackets in the program form a *regular bracket sequence*, so each opening bracket '{' has uniquely defined matching closing bracket '}' and vice versa. A variable is declared in a scope at most once. Any scope (including global) can be empty, i.e. can contain no variable declarations.

Output

For each fact of shadowing write a line in form "`r1:c1: warning: shadowed declaration of ?, the shadowed position is r2:c2`", where "`r1:c1`" is the number of line and position in line of shadowing declaration and "`r2:c2`" is the number of line and position in line of shadowed declaration. Replace '?' with the letter `'a' ... 'z'` — the name of shadowing/shadowed variable. If multiple outer scopes have variables named as the shadowing variable, the variable in the nearest outer scope is shadowed.

Print warnings in increasing order of `r1`, or in increasing order of `c1` if values `r1` are equal. Leave the output empty if there are no variable shadowings.

Examples

standard input
1 { a { b { a } } } b
standard output
1:11: warning: shadowed declaration of a, the shadowed position is 1:3

standard input
1 { a { a { a } } }
standard output
1:7: warning: shadowed declaration of a, the shadowed position is 1:3 1:11: warning: shadowed declaration of a, the shadowed position is 1:7

Алгоритм

Задачу можно решить с помощью стека и вектора стеков. Предупреждение нужно выводить когда новая переменная пытается попасть в стек, который не пуст, это значит уже была объявлена переменная с таким же именем. Когда появляется закрывающая скобка убираем все до открывающей.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
typedef struct {
    char ch;
    int line;
    int sym;
} var;
int main() {
    int f,n;
    cin >> n;
    vector <stack <var> > all_alph(26);
    stack <var> curr;
    char temp;
    int symbol = 0;
    var to_put;
    temp = cin.get();
    for (int i = 1; i <= n; i++){
        temp = cin.get();
        symbol = 1;
        while(temp != '\n') {
            if(temp == '}') {
                to_put = curr.top();
                curr.pop();
                while (to_put.ch != '{') {
                    all_alph[to_put.ch - 97].pop();
                    to_put = curr.top();
                    curr.pop();
                }
            }
            else if(temp == '{') {
                to_put.ch = temp;
                to_put.sym = symbol;
                to_put.line = i;
                curr.push(to_put);
            }
            else if (temp != ' '){
                to_put.sym = symbol;
                to_put.line = i;
                to_put.ch = temp;
                curr.push(to_put);
                if(all_alph[temp - 97].size() != 0)
                    cout << to_put.line << ":" << to_put.sym
                    << ": warning: shadowed declaration of "<< to_put.ch
                    << ", the shadowed position is " << all_alph[temp - 97].top().line
                    << ":" << all_alph[temp - 97].top().sym << endl;
                all_alph[temp - 97].push(to_put);
            }

            temp = cin.get();
            symbol++;
        }
    }
    return 0;
}
```

2.11 ACM-ICPC Московский четвертьфинал

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

2.12 Codeforces Training S02E07

Результаты

Задачи

№	Название			
A	Arithmetic Rectangle	стандартный ввод/вывод 5 с, 256 МБ		x51
B	Bytean Road Race	стандартный ввод/вывод 4 с, 256 МБ		x13
C	Will It Stop?	стандартный ввод/вывод 1 с, 256 МБ		x410
D	Ants	стандартный ввод/вывод 15 с, 20 МБ		x4
E	Gophers	стандартный ввод/вывод 8 с, 256 МБ		x177
F	Laundry	стандартный ввод/вывод 5 с, 256 МБ		x159
G	Bits Generator	стандартный ввод/вывод 3 с, 256 МБ		x57
H	Afternoon Tea	стандартный ввод/вывод 1 с, 256 МБ		x254
I	Intelligence Quotient	стандартный ввод/вывод 6 с, 256 МБ		x46
J	Cave	стандартный ввод/вывод 30 с, 256 МБ		x113
K	Cross Spider	стандартный ввод/вывод 1 с, 256 МБ		x249

Ссылка на контест: http://codeforces.com/gym/100523

Задача C - Will It Stop?

Byteasar was wandering around the library of the University of Warsaw and at one of its facades he noticed a piece of a program with an inscription "Will it stop?". The question seemed interesting, so Byteasar tried to tackle it after returning home. Unfortunately, when he was writing down the piece of code he made a mistake and noted:

$$\begin{array}{l} \text{while } n > 1 \text{ do} \\ \quad \text{if } n \bmod 2 = 0 \text{ then} \\ \quad\quad n := n/2 \\ \quad \text{else} \\ \quad\quad n := 3 \cdot n + 3 \end{array}$$

Byteasar is now trying to figure out, for which initial values of the variable n the program he wrote down stops. We assume that the variable n has an unbounded size, i.e., it may attain arbitrarily large values.

Input

The first and only line of input contains one integer n ($2 \leqslant n \leqslant 10^{14}$).

Output

In the first and only line of output you should write a single word `TAK` (i.e., *yes* in Polish), if the program stops for the given value of n, or `NIE` (*no* in Polish) otherwise.

Example

For the input data:

```
4
```

the correct result is:

```
TAK
```

Алгоритм

Делим число n на 2 до тех пор пока оно не перестанет делиться на 2. Если получившееся число равно единице, то выводим TAK, если нет, то NIE.

Исходный код

```
#include <iostream>
using namespace std;
int main()
{
    unsigned long long a;
    cin >> a;
    while(!(a%2))a/=2;
    if(a==1)
        cout << "TAK";
    else
        cout << "NIE";
    return 0;
}
```

Задача H - Afternoon Tea

During his visit at Bytic Islands Byteasar really enjoyed the national beverage of Byteans, that is, tea with milk. This drink is always prepared in a strictly determined manner, which is as follows. Firstly the teacup is filled with tea mixed half and half with milk. Then, an n-letter *ceremonial word* consisting of letters `H` and `M` is chosen. Now, for $i = 1, 2, \ldots, n$, the following action is performed: if the i-th letter of the ceremonial word is `H`, one should drink half of the teacup, add tea until the teacup is full and stir. On the other hand, if the i-th letter of the word is `M`, one should perform a similar action, however milk should be added instead of tea. After such action is performed for each letter of the ceremonial word, the remaining liquid is disposed of.

Each time Byteasar performs the ceremony, he wonders which of the ingredients he has drunk more: tea or milk. Help Byteasar answer this question.

Input

The first line of input holds an integer n ($1 \leqslant n \leqslant 100\,000$). The second line contains an n-letter word consisting of letters `H` and `M`; this is the ceremonial word used by Byteasar.

Output

Your program should output a single letter `H` if Byteasar has drunk more tea than milk; a single letter `M` if he has drunk more milk than tea; or the word `HM` if he has drunk equal amounts of tea and milk.

Example

For the input data:	the correct result is:
5 HMHHM	H

Explanation of the example: Byteasar has drunk $1\frac{37}{64}$ teacups of tea and $\frac{59}{64}$ teacups of milk in total.

Алгоритм

Создадим две переменные одну для чая, другую для молока и будем последовательно, для каждой буквы, добавлять к ним значение зависящее от позиции текущей буквы, и посчитанное по выведенной формуле. Сложность O(n).

Исходный код

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n;
    cin >> n;
    if(n==1)
    {
        cout << "HM";
        return 0;
    }
    cin.get();
    char c;
    long double hDrunked = 0, mDrunked = 0;
    hDrunked = mDrunked = (1-pow(0.5, n))*0.5;
    for(int i=0; i<n-1; ++i)
    {
        c=cin.get();
        if(c=='H')
            hDrunked += (1-pow(0.5, n-i-1))*0.5;
        else
            mDrunked += (1-pow(0.5, n-i-1))*0.5;
    }
    if(hDrunked>mDrunked)
        cout << "H";
    else if(hDrunked<mDrunked)
        cout << "M";
    return 0;
}
```

2.13 Codeforces Crypto Cup 1.0

Результаты

Задачи

№	Название			
A	Bank	стандартный ввод/вывод 2 с, 64 МБ		x81
B	:-P	стандартный ввод/вывод 1 с, 256 МБ		x73
C	Pgkpxumgs	стандартный ввод/вывод 1 с, 256 МБ		x131
D	13lk	стандартный ввод/вывод 1 с, 256 МБ		x27
E	Peace of AmericanWedding	стандартный ввод/вывод 1 с, 256 МБ		x62
F	2715	стандартный ввод/вывод 1 с, 256 МБ		x19
G	uehSlff	стандартный ввод/вывод 2 с, 64 МБ		x20
H	Peace of AmericaReunion	стандартный ввод/вывод 1 с, 256 МБ		x171
I	Peace of AmericanPie	стандартный ввод/вывод 1 с, 256 МБ		x279
J	Common	стандартный ввод/вывод 1 с, 256 МБ		x236
K	Crap	стандартный ввод/вывод 1 с, 256 МБ		x104
L	Key	стандартный ввод/вывод 1 с, 256 МБ		x22
M	oPlus	стандартный ввод/вывод 2 с, 64 МБ		x216
N	tirnaoeumPt	стандартный ввод/вывод 1 с, 256 МБ		x95
O	0x	стандартный ввод/вывод 1 с, 256 МБ		x75
P	Proooooooooooooofer	стандартный ввод/вывод 1 с, 256 МБ		x11
Q	Peace of bzjd	стандартный ввод/вывод 1 с, 256 МБ		x337
R	6227020800	стандартный ввод/вывод 1 с, 256 МБ		x331

Ссылка на контест: http://codeforces.com/gym/100514

Задача B - :-P

You are given an encrypted string, encrypted using a certain algorithm. Decrypt it !

Input
The first line of input contains a string s, which each of it's characters is a lower case English letter. ($1 \le |s| \le 10^5$)

The second line contains single integer p. ($1 \le p \le |s|$)

Output
Print the original string.

Sample test(s)

input
crhhzae 3
output
charzeh

Исходный код

```
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;
int main(int argc, const char * argv[]) {
    char str[100001];
    long p, len;
    while((str[len]=cin.get())!='\n') ++len;
    str[len] = '\0';
    cin >> p;
    vector< vector <char> > v(p);
    long vSize = len/p, r = len%p;
    for(long i=r; i<p; ++i) {
        v[i].resize(vSize);
    }
    for(long i=0; i<r; ++i) {
        v[i].resize(vSize+1);
    }
    for(long i=0, k=0; i<p; ++i) {
        for(long j=0; j<v[i].size(); ++j, ++k) {
            v[i][j] = str[k];
        }
    }
    for(long i=0; i<len; ++i) {
        cout.put(v[i%p][i/p]);
    }
    return 0;
}
```

Задача C - Pgkpxumgs

You are given an encrypted string, encrypted using a certain algorithm.Decrypt it !

Input
The first and single line of input contains a string s, which each of it's characters is a lower case English letter. ($1 \le |s| \le 10^5$)

Output
Print the original string.

Sample test(s)

input
cjjazdk
output
charzeh

Исходный код

```
#include <iostream>
#include <cstdio>
using namespace std;
int main() {
  char cur, prev;
  cout.put(prev = cin.get());
  while ((cur = cin.get()) != '\n') {
    if ((int)(cur - prev) < 0) cout << (char)(cur - prev + '{');
    else cout << (char)(cur - prev + 'a');
    prev = cur;
  }
  return 0;
}
```

Задача H - Peace of AmericaReunion

You are given an encrypted string, encrypted using a certain algorithm. Decrypt it !

Encryption algorithm's output is more than one string.

Input

The first line of input contains 26 integers $a_1, ..., a_{26}$ separated by space ($1 \le a_i \le 10^5$).

Next $\sum_{i=1}^{26} a_i$ lines, each line contains an integer between 1 and 10^5 inclusive.

Output

Print a single string in a single line.

Sample test(s)

input

```
1 0 1 0 1 0 0 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
3
1
6
2
7
4
5
```

output

```
charzeh
```

Исходный код

```cpp
#include <iostream>
using namespace std;
int main() {
  vector<long> v(26);
  long length = 0;
  for (int i = 0; i < 26; i++) {
    cin >> v[i];
    length += v[i];
  }
  vector<char> answer(length);
  long pos;
  int nextSymb = -1;
  for (int i = 0; i < 26; i++) {
    if (v[i]) {
      nextSymb = i;
      break;
    }
  }
  for (long i = 0; i < length; i++) {
    cin >> pos;
    if (!v[nextSymb]) {
      for (int i = nextSymb; i < 26; i++) {
        if (v[i]) {
          nextSymb = i;
          break;
        }
      }
    }
    answer[--pos] = (char)(nextSymb + 'a');
    v[nextSymb]--;
  }
  for (auto n : answer) cout << n;
  return 0;
}
```

Задача I - Peace of AmericanPie

You are given an encrypted string, encrypted using a certain algorithm. Decrypt it !

Input
The first and only line of input contains a string s,each character of s is either 0 or 1 . ($8 \leq |s| \leq 8 \times 10^4$)

Output
Print the original string.

Sample test(s)

input
01100011011010000110000101110010011110100110010101101000
output
charzeh

Исходный код

```
#include <iostream>
using namespace std;
int main() {
  int byte = 0;
  int spow = 256;
  int collected = 0;
  char currentBit;
  while ((currentBit = cin.get()) != '\n') {
    cin.unget();
    while (collected < 8) {
      currentBit = cin.get();
      byte += (currentBit - '0') * spow;
      spow /= 2;
      collected++;
    }
    cout << (char)(byte / 2);
    byte = 0;
    spow = 256;
    collected = 0;
  }
  return 0;
}
```

Задача J - Common

You are given an encrypted string, encrypted using a certain algorithm. Decrypt it !

Input
The first and single line of input contains a string s, which each of it's characters is a lower case English letter. ($1 \leq |s| \leq 10^5$)

Output
Print the original string.

Sample test(s)

input
wbpctfb
output
charzeh

input
ghnxfuv
output
yousefi

Исходный код

```
#include <iostream>
using namespace std;
int main() {
  char t;
  while ((t = cin.get()) != '\n') {
    switch (t) {
      case 'a':
        cout << "n";
        break;
      case 'b':
        cout << "h";
        break;
      case 'c':
        cout << "r";
        break;
      case 'd':
        cout << "x";
        break;
      case 'e':
        cout << "k";
        break;
      case 'f':
        cout << "e";
        break;
      case 'g':
        cout << "y";
        break;
      case 'h':
        cout << "o";
        break;
      case 'i':
        cout << "q";
        break;
      case 'j':
        cout << "m";
        break;
```

```
37        case 'k':
38          cout << "j";
39          break;
40        case 'l':
41          cout << "b";
42          break;
43        case 'm':
44          cout << "d";
45          break;
46        case 'n':
47          cout << "u";
48          break;
49        case 'o':
50          cout << "v";
51          break;
52        case 'p':
53          cout << "a";
54          break;
55        case 'q':
56          cout << "p";
57          break;
58        case 'r':
59          cout << "w";
60          break;
61        case 's':
62          cout << "g";
63          break;
64        case 't':
65          cout << "z";
66          break;
67        case 'u':
68          cout << "f";
69          break;
70        case 'v':
71          cout << "i";
72          break;
73        case 'w':
74          cout << "c";
75          break;
76        case 'x':
77          cout << "s";
78          break;
79        case 'y':
80          cout << "t";
81          break;
82        case 'z':
83          cout << "l";
84          break;
85      }
86    }
87    return 0;
88  }
```

Задача M - oPlus

You are given an encrypted string, encrypted using a certain algorithm.Decrypt it !

Each character of the encrypted string has ASCII code between 0 and 255 inclusive.So you're given the ASCII code of each character.It's guaranteed that the original string is made of lower case English letters.

Input

The first line of input contains integer n, the size of the encrypted string. ($1 \le n \le 10^5$).

The second line contains n integers between 0 and 255 inclusive, speared by space.

Output

Print the original string.

Sample test(s)

input
7 189 182 191 172 164 187 182
output
charzeh

Исходный код

```
#include <iostream>
using namespace std;
int main() {
  long n;
  int curr, sum;
  cin >> n;
  for (long i = 0; i < n; i++) {
    cin >> curr;
    if (curr % 2) sum = 400;
    else sum = 398;
    cout << (char)(sum - curr - 112);
  }
  return 0;
}
```

Задача N - tirnaoeumPt

You are given an encrypted string, encrypted using a certain algorithm.Decrypt it !

Each character of the encrypted string has ASCII code between 0 and 255 inclusive.So you're given the ASCII code of each character.It's guaranteed that the original string is made of lower case English letters.

Input

The first line of input contains integer n, the size of the encrypted string. ($1 \le n \le 10^5$).

The second line contains n integers between 0 and 255 inclusive, speared by space.

Output

Print the original string.

Sample test(s)

input
7 8 25 0 3 7 16 25
output
charzeh

Исходный код

```
#include <iostream>
using namespace std;

int main()
{
    int m[] = {0, 1, 16, 17, 8, 9, 24, 25, 2, 3, 18, 19, 10, 11, 22, 23, 4, 5, 20, 21, 12, 13, 22, 23, 6, 7, 22, 23, 14, 15};
    int n, d;
    cin >> n;
    for(int i=0; i<n; ++i)
    {
        cin >> d;
        cout.put(m[d]+'a');
    }
    return 0;
}
```

Задача Q - Peace of bzjd

You are given an encrypted string, encrypted using a certain algorithm. Decrypt it !

Input
The first and single line of input contains a string s, which each of it's characters is a lower case English letter. ($1 \leq |s| \leq 10^5$)

Output
Print the original string.

Sample test(s)

input
bgzqydg
output
charzeh

input
xntrdeh
output
yousefi

Исходный код

```
#include <iostream>

using namespace std;

int main()
{
    char temp;
    temp = cin.get();
    while (temp != '\n' && temp != EOF) {
        if (temp == 'z')
            temp = 'a';
        else temp++;
        cout << temp;
        temp = cin.get();
    }
    return 0;
}
```

Задача R - 6227020800

You are given an encrypted string, encrypted using a certain algorithm.Decrypt it !

Input
The first and single line of input contains a string s, which each of it's characters is a lower case English letter. ($1 \leq |s| \leq 10^5$)

Output
Print the original string.

Sample test(s)

input
punemru
output
charzeh

input
lbhfrsv
output
yousefi

Исходный код

```
#include <iostream>
#include <algorithm>
#include <deque>
#include <cstdio>
using namespace std;
void p(string s) {
  cout << s << endl;
}
int gcd(int a, int b){
    if (b == 0)
        return a;
    return gcd(b, a%b);
}
int main() {
  char t;
  while ((t = cin.get()) != '\n') {
    t -= 13;
    if (t < 'a') {
      cout << (char)('z' - ('a' - t) + 1);
    }
    else {
      cout << t;
    }
  }
  return 0;
}
```

2.14 OpenCup GrandPrix of Siberia

Результаты

26.	MAI #11: Makarov, Rik, Yakimenko	-	-10 4:49	-	-	-	-	-	-	-	+ 0:00	+4 2:12	+4 1:39	3	392	72%	0.39

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10282

Задача 12 - Construction of Chand Baori

The Indian stepped well Chand Baori is, with some simplifications, a pyramid tapering down. Faces of the pyramid are made up of rows of trapezoid stairs which can be used to descend to reach the water. The first (lowest) level is a single stairway with two flights of stairs on the left and on the right. The second level has two such stairways, etc.

To visit the "Harshat Mata" shrine dedicated to the goddess of happiness and joy pilgrims need to cleanse themselves in the waters of the well, that is, to descend to the very bottom. Pilgrims on their way to the water can only descend or walk horizontally along the current level, but not climb up.

The builders of the well want to construct it in such a way that the number of possible ways of descend would be no less than the number of pilgrims. Two paths are considered different if there exists a level with a different stairway used for descending, or different flight of the same stairway used.

Input

The first line of the input file contains two integers N and M – the number of levels in the well and the number of pilgrims descending along the single face, respectively ($1 \leq N \leq 20$, $0 \leq M \leq 1.5 \cdot 10^{18}$).

Output

The output file must contain "`Harshat Mata`", if the number of possible ways to descend along one face of the well is less than the given number of pilgrims (also for one face), otherwise it must contain "`Nope`".

Examples

input.txt	output.txt
1 2	Harshat Mata
1 3	Nope
2 9	Nope
2 8	Harshat Mata

Алгоритм

Если $n! > m$, то вывести $HarshatMata$, если нет, то $Nope$. Сложность $O(n)$.

Исходный код

```
#include <iostream>
using namespace std;
#define ULL unsigned long long
int main(int argc, const char * argv[]) {
    ULL n, m;
    cin >> n >> m;
    ULL res = 1;
    for(int i=2; i<=n*2; i+=2)
    {
        res *= i;
    }
    if(res<m)
        cout << "Nope";
    else
        cout << "Harshat Mata";
    return 0;
}
```

Задача 13 - Sum

Даны три целых числа A, K и P. Вычислите следующую сумму:

$$\sum_{i=1}^{K} A^i \ mod \ P$$

Input

Первая и единственная строка входного файла содержит три целых числа A ($0 \le A \le 10^8$), K ($1 \le K \le 10^{16}$) и P ($1 \le P \le 10^8$), разделённые пробелами.

Output

Выведите одно число — значение требуемой суммы.

Examples

input.txt	output.txt
3 4 101	120

Алгоритм

Для того, чтобы подсчитать требуемую сумму можно воспользоваться рекуррентной формулой $a_i = (A * a_{i-1}) \ mod \ P$, где $i \in [2; K]$ и $a_1 = 1$. Сложность $O(K)$.

Исходный код

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
using namespace std;
#define ULL unsigned long long
#define LL long long
int main(int argc, const char * argv[]) {
    ifstream in("input.txt");
    ofstream out("output.txt");
    ULL a, k, p;
    cin >> a >> k >> p;
    ULL sum = 0, prev = 1;
    for(ULL i = 0; i<k; ++i)
    {
        prev = (prev*a)%p;
        sum += prev;
    }
    cout << sum;
    out.close();
    in.close();
    return 0;
}
```

Задача 14 - Coinquerors

Правила старинной игры “Coinquerors” заключаются в следующем.

Каждый игрок участвует со своей монетой. Монета должна представлять собой окружность целого радиуса. Далее игроки бросают свои монеты так, что центр каждой монеты оказывается в точке с целыми координатами. После броска каждого игрока накрытый его монетой участок отмечается.

По завершении игры рассматриваются все отмеченные участки. Если у двух участников отмеченные участки пересекаются, то они объявляются союзниками. При этом гарантируется, что никакие два участка не имеют ровно одну общую точку (то есть случай касания соответствующих окружностей невозможен). Участник, набравший как можно больше союзников, и объявляется победителем.

Ваша задача — по координатам центров упавших монет и их радиусам определить победителя или же сказать, что игра завершилась вничью.

Input

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

Далее задаются тестовые примеры. Каждый тестовый пример начинается строкой, содержащей одно целое число N ($2 \le N \le 100$).

Каждая из последующих N строк содержит имя игрока, состоящее из не менее, чем двух и не более, чем из 255 строчных латинских букв, и три целых числа X, Y и R, задающих x и y координаты центра брошенной игроком монеты и её радиус ($-100 \le X, Y \le 100$, $1 \le R \le 10$).

Output

Для каждого тестового примера выведите одну строку с имеем победившего игрока. В случае, если победителей несколько, выведите строку “TIE”.

Example

input.txt	output.txt
3 3 gennady 0 0 3 tomek 1 1 1 petr -1 -1 1 2 alice -100 -100 1 bob 100 100 100 3 john 0 0 10 john 2 2 3 jack -5 0 1	gennady TIE john

Алгоритм

Просто подсчитываем количество пересечений у каждой окружности с остальными и находим ту, у которой это количество максимально. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
using namespace std;
#define ULL unsigned long long
#define LL long long
#define eps 0.0001
struct player
{
    char name[256];
    LL x, y, r;
};
int main(int argc, const char * argv[]) {
    ifstream in("input.txt");
    ofstream out("output.txt");
    ULL T;
    in >> T;
    double pi81 = M_PI/81, pi2 = M_PI*2;
    for(ULL TT = 0; TT<T; ++TT)
    {
        ULL n;
        in >> n;
        vector<player> pl(n);
        for(int i=0; i<n; ++i)
        {
            in >> pl[i].name >> pl[i].x >> pl[i].y >> pl[i].r;
        }
        ULL maxInd = -1, max = 0, forTie = -1;
        for(int i=0; i<n; ++i)
        {
            ULL count = 0;
            LL rr = pl[i].r*pl[i].r;
            for(int j=0; j<n; ++j)
            {
                for(double pi = 0; pi<=pi2; pi += pi81)
                {
                    double x = (pl[j].r-eps)*cos(pi);
                    double y = (pl[j].r-eps)*sin(pi);
                    if((pl[j].x-pl[i].x+x)*(pl[j].x-pl[i].x+x)+(pl[j].y-pl[i].y+y)*(pl[j].y-pl[i].y+y)<=rr)
                    {
                        ++count;
                        break;
                    }
                }
            }
            if(count > max)
            {
                max = count;
                maxInd = i;
                forTie = -1;
            }
```

```
            else if(count == max)
            {
                max = count;
                forTie = maxInd;
                maxInd = i;
            }
        }
        if(maxInd == -1 || forTie != -1)
            out << "TIE" << '\n';
        else
            out << pl[maxInd].name << '\n';
    }
    out.close();
    in.close();
    return 0;
}
```

2.15 Codeforces Training S02E08

Результаты

Задачи

№	Название			
A	Avoiding the Apocalypse	стандартный ввод/вывод 6 с, 256 МБ		x72
B	Button Bashing	стандартный ввод/вывод 1 с, 256 МБ		x189
C	Citadel Construction	стандартный ввод/вывод 6 с, 256 МБ		x94
D	Dropping Directions	стандартный ввод/вывод 3 с, 256 МБ		x71
E	Excellent Engineers	стандартный ввод/вывод 3 с, 256 МБ		x163
F	Floating Formation	стандартный ввод/вывод 1 с, 256 МБ		x28
G	Growling Gears	стандартный ввод/вывод 1 с, 256 МБ		x235
H	Highway Hassle	стандартный ввод/вывод 3 с, 256 МБ		x17
I	Interesting Integers	стандартный ввод/вывод 1 с, 256 МБ		x135
J	Jury Jeopardy	стандартный ввод/вывод 1 с, 256 МБ		x173
K	Key to Knowledge	стандартный ввод/вывод 3 с, 256 МБ		x107

Ссылка на контест: http://codeforces.com/gym/100526

Задача G - Growling Gears

The *Best Acceleration Production Company* specializes in multi-gear engines. The performance of an engine in a certain gear, measured in the amount of torque produced, is not constant: the amount of torque depends on the RPM of the engine. This relationship can be described using a *torque-RPM curve*.

The torque-RPM curve of the gears given in the second sample input. The second gear can produce the highest torque.

For the latest line of engines, the torque-RPM curve of all gears in the engine is a parabola of the form $T = -aR^2 + bR + c$, where R is the RPM of the engine, and T is the resulting torque.

Given the parabolas describing all gears in an engine, determine the gear in which the highest torque is produced. The first gear is gear 1, the second gear is gear 2, etc. There will be only one gear that produces the highest torque: all test cases are such that the maximum torque is at least 1 higher than the maximum torque in all the other gears.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with a single integer n ($1 \leq n \leq 10$): the number of gears in the engine.
- n lines, each with three space-separated integers a, b and c ($1 \leq a, b, c \leq 10\,000$): the parameters of the parabola $T = -aR^2 + bR + c$ describing the torque-RPM curve of each engine.

Output

Per test case:

- one line with a single integer: the gear in which the maximum torque is generated.

Sample in- and output

Input	Output
3 1 1 4 2 2 3 126 1400 2 152 208 2 3 127 1400 2 154 208	1 2 2

Алгоритм

В задаче нужно найти передачу с максимальной скоростью вращения. Для этого мы будем применять формулу $(b * b)/(4 * a) + c$ для каждой передачи. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <algorithm>

#include <fstream>

using namespace std;

int main() {
    int k;
    cin >> k;
    int n, a, b, c;
    double max_T = -1000000;
    int max_T_num;
    double temp;

    for(int i = 0; i < k; i++) {
        cin >> n;
        for(int j = 1; j <= n; j++) {
            cin >> a >> b >> c;
            temp = (b * b) / (4 * a) + c;
            if(temp > max_T) {
                max_T = temp;
                max_T_num = j;
            }
        }
        cout << max_T_num << endl;
        max_T = -1000000;
    }
  int q;
  cin >> q;
    return 0;
}
```

Задача J - Jury Jeopardy

What would a programming contest be without a problem featuring an ASCII-maze? Do not despair: one of the judges has designed such a problem.

The problem is about a maze that has exactly one entrance/exit, contains no cycles and has no empty space that is completely enclosed by walls. A robot is sent in to explore the entire maze. The robot always faces the direction it travels in. At every step, the robot will try to turn right. If there is a wall there, it will attempt to go forward instead. If that is not possible, it will try to turn left. If all three directions are unavailable, it will turn back.

source: xkcd.com/246

The challenge for the contestants is to write a program that describes the path of the robot, starting from the entrance/exit square until it finally comes back to it. The movements are described by a single letter: 'F' means forward, 'L' is left, 'R' is right and 'B' stands for backward. Each of 'L', 'R' and 'B' does not only describe the change in orientation of the robot, but also the advancement of one square in that direction. The robot's initial direction is East. In addition, the path of the robot always ends at the entrance/exit square.

The judge responsible for the problem had completed all the samples and testdata, when disaster struck: the input file got deleted and there is no way to recover it! Fortunately the output and the samples are still there. Can you reconstruct the input from the output? For your convenience, he has manually added the number of test cases to both the sample output and the testdata output.

Input

On the first line one positive number: the number of test cases. After that per test case:

- one line with a single string: the movements of the robot through the maze.

Output

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers h and w ($3 \leq h, w \leq 100$): the height and width of the maze, respectively.
- h lines, each with w characters, describing the maze: a '#' indicates a wall and a '.' represents an empty square.

The entire contour of the maze consists of walls, with the exception of one square on the left: this is the entrance. The maze contains no cycles (i.e. paths that would lead the robot back to a square it had left in another direction) and no empty squares that cannot be reached from the entrance. Every row or column – with the exception of the top row, bottom row and right column – contains at least one empty square.

Sample in- and output

Input	Output
3 FFRBLF FFRFRBRFBFRBRFLF FRLFFFLBRFFFRFFFRFRFBRFLBRFRLFLFFR	3 4 4 #### ...# ##.# #### 7 5 ##### ...## ##.## #...# ##.## ##.## ##### 7 7 ####### #...#.# #.#...# #.#.### ..###.# #.....# #######

Алгоритм

Создаём матрицу, изначально заполненную решётками. Воспроизводим путь робота согласно входным символам и отмечаем путь точками, в процессе запоминая максимальную и минимальную координаты. Используя найденные граничные координаты, вычисляем размер поля и выводим само поле. Сложность $O(n)$.

Исходный код

```
#include <iostream>
using namespace std;
int main(int argc, const char * argv[]) {
    long T;
    cin >> T;
    cout << T << '\n';
    char map[201][201];
    cin.get();
    while(T--)
    {
        for(long i=0; i<201; ++i)
            for(long j=0; j<201; ++j)
                map[i][j] = '#';
        char c;
        long x, y, minX, minY, maxX, maxY;
        maxX = maxY = minX = minY = x = y = 100;
        int dir = 0;
        while((c=cin.get())!='\n')
        {
            switch(dir)
            {
                case 0:
                    if(c=='R')
                        dir = 3;
                    else if(c=='L')
                        dir = 1;
                    else if(c=='B')
                        dir = 2;
                    break;
                case 1:
                    if(c=='R')
                        dir = 0;
                    else if(c=='L')
                        dir = 2;
                    else if(c=='B')
                        dir = 3;
                    break;
                case 2:
                    if(c=='R')
                        dir = 1;
                    else if(c=='L')
                        dir = 3;
                    else if(c=='B')
                        dir = 0;
                    break;
                case 3:
                    if(c=='R')
                        dir = 2;
                    else if(c=='L')
                        dir = 0;
                    else if(c=='B')
                        dir = 1;
                    break;
            }
            switch(dir)
            {
                case 0:
                    ++x;
```

```
                    if(x>maxX)
                        maxX = x;
                    break;
                case 1:
                    --y;
                    if(y<minY)
                        minY = y;
                    break;
                case 2:
                    --x;
                    if(x<minX)
                        minX = x;
                    break;
                case 3:
                    ++y;
                    if(y>maxY)
                        maxY = y;
                    break;
            }
            map[x][y] = '.';
        }
        cout << maxY-minY+3 << ' ' << maxX-minX+2 << '\n';
        for(long i = minY-1; i<=maxY+1; ++i)
        {
            for(long j = minX; j<=maxX+1; ++j)
                cout.put(map[j][i]);
            cout.put('\n');
        }
    }
    return 0;
}
```

2.16 Codeforces Training S02E09

Результаты

Задачи

№	Название			
A	ASCII Art	ascii.in / ascii.out 3 с, 256 МБ		x38
B	Billing Tables	billing.in / billing.out 1 с, 256 МБ		x17
C	Cellular Automaton	cell.in / cell.out 2 с, 256 МБ		x113
D	Driving Directions	driving.in / driving.out 1 с, 256 МБ		
E	Exchange	exchange.in / exchange.out 1 с, 256 МБ		x53
F	Fool's Game	fool.in / fool.out 1 с, 256 МБ		x4
G	Graveyard	graveyard.in / graveyard.out 1 с, 256 МБ		x232
H	Hard Life	hard.in / hard.out 1 с, 256 МБ		x10
I	Interconnect	interconnect.in / interconnect.out 1 с, 256 МБ		x67
J	Java vs C++	java_c.in / java_c.out 1 с, 256 МБ		x294
K	Kickdown	kickdown.in / kickdown.out 1 с, 256 МБ		x256

Ссылка на контест: http://codeforces.com/gym/100532

Задача G - Graveyard

Programming contests became so popular in the year 2397 that the governor of New Earck — the largest human-inhabited planet of the galaxy — opened a special Alley of Contestant Memories (ACM) at the local graveyard. The ACM encircles a green park, and holds the holographic statues of famous contestants placed equidistantly along the park perimeter. The alley has to be renewed from time to time when a new group of memorials arrives.

When new memorials are added, the exact place for each can be selected arbitrarily along the ACM, but the equidistant disposition must be maintained by moving some of the old statues along the alley.

Surprisingly, humans are still quite superstitious in 24th century: the graveyard keepers believe the holograms are holding dead people souls, and thus always try to renew the ACM with minimal possible movements of existing statues (besides, the holographic equipment is very heavy). Statues are moved along the park perimeter. Your work is to find a renewal plan which minimizes the sum of travel distances of all statues. Installation of a new hologram adds no distance penalty, so choose the places for newcomers wisely!

Input

Input file contains two integer numbers: n — the number of holographic statues initially located at the ACM, and m – the number of statues to be added ($2 \le n \le 1000, 1 \le m \le 1000$). The length of the alley along the park perimeter is exactly 10 000 feet.

Output

Write a single real number to the output file — the minimal sum of travel distances of all statues (in feet). The answer must be precise to at least 4 digits after decimal point.

Sample input and output

graveyard.in	graveyard.out
2 1	1666.6667
2 3	1000.0
3 1	1666.6667
10 10	0.0

Pictures show the first three examples. Marked circles denote original statues, empty circles denote new equidistant places, arrows denote movement plans for existing statues.

Алгоритм

Для того, чтобы решить эту задачу, нужно представить окружность в виде отрезка с длиной 10000. Разобьём отрезок сначала на n частей и запишем точки в вектор a, затем разобьём отрезок на $n+m$ частей и запишем точки в вектор b. Найдём пару ближайших точек из векторов a и b, расстояние между ними и будет ответом. Сложность $O(n^2 + nm)$.

Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <string>
4 #include <algorithm>
5 #include <queue>
6 #include <climits>
7 #include <cctype>
8 #include <cmath>
9 #include <fstream>
10 #include <iomanip>
11 #define ll long long
12 #define ull unsigned long long
13 using namespace std;
14 int main()
15 {
16     ifstream in("graveyard.in");
17     ofstream out("graveyard.out");
18     ll n, m;
19     in >> n >> m;
20     ll nm = n+m;
21     vector<double> a(n);
22     vector<double> b(nm);
23     double s = 0;
24     for(ll i=0; i<n; ++i)
25     {
26         a[i] = s;
27         s += 10000.0/n;
28     }
29     s = 0;
30     for(ll i=0; i<nm; ++i)
31     {
32         b[i] = s;
33         s += 10000.0/nm;
34     }
35     vector<bool> u(nm, 0);
36     vector<double> r(n);
37     double res = 0;
38     for(ll i=0; i<n; ++i)
39     {
40         double min = 200000.0;
41         for(ll j=0; j<nm; ++j)
42         {
43
44             if(!u[j] && fabs(a[i]-b[j])<min)
45             {
46                 u[j] = 1;
47                 min = fabs(a[i]-b[j]);
48             }
49         }
50         res += min;
51     }
52     out << fixed << setprecision(4) << res;
53     in.close();
54     out.close();
55     return 0;
56 }
```

Задача J - Java vs C++

Apologists of Java and C++ can argue for hours proving each other that their programming language is the best one. Java people will tell that their programs are clearer and less prone to errors, while C++ people will laugh at their inability to instantiate an array of generics or tell them that their programs are slow and have long source code.

Another issue that Java and C++ people could never agree on is identifier naming. In Java a multiword identifier is constructed in the following manner: the first word is written starting from the small letter, and the following ones are written starting from the capital letter, no separators are used. All other letters are small. Examples of a Java identifier are `javaIdentifier`, `longAndMnemonicIdentifier`, `name`, `nEERC`.

Unlike them, C++ people use only small letters in their identifiers. To separate words they use underscore character '_'. Examples of C++ identifiers are `c_identifier`, `long_and_mnemonic_identifier`, `name` (you see that when there is just one word Java and C++ people agree), `n_e_e_r_c`.

You are writing a translator that is intended to translate C++ programs to Java and vice versa. Of course, identifiers in the translated program must be formatted due to its language rules — otherwise people will never like your translator.

The first thing you would like to write is an identifier translation routine. Given an identifier, it would detect whether it is Java identifier or C++ identifier and translate it to another dialect. If it is neither, then your routine should report an error. Translation must preserve the order of words and must only change the case of letters and/or add/remove underscores.

Input

The input file consists of one line that contains an identifier. It consists of letters of the English alphabet and underscores. Its length does not exceed 100.

Output

If the input identifier is Java identifier, output its C++ version. If it is C++ identifier, output its Java version. If it is none, output "`Error!`" instead.

Sample input and output

java_c.in	java_c.out
long_and_mnemonic_identifier	longAndMnemonicIdentifier
anotherExample	another_example
i	i
bad_Style	Error!

Алгоритм

Посимвольно считываем и проверяем на признаки $Java$ и $C++$. Если обнаружились оба признака, то выводим ошибку. Если только один, то приводим строку к соответствующему формату. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#define ll long long
#define ull unsigned long long
#define ERROR {out << "Error!"; return 0;}
using namespace std;
int main()
{
    ifstream in("java_c.in");
    ofstream out("java_c.out");
    char c;
    bool java = 0, cpp = 0, us = 0;
    char str[1000];
    ll n=0;
    while(!in.eof() && (c=in.get())!='\n')
    {
        if(in.eof())
            break;
        if(c=='_')
        {
            if(us || !n)
                ERROR
            cpp = 1;
            us = 1;
        }
        else if(isupper(c))
        {
            if(!n)
                ERROR
            java = 1;
            str[n++] = '_';
            str[n++] = tolower(c);
        }
        else if(us)
            str[n++] = toupper(c);
        else
            str[n++] = c;
        if(c!='_')
            us = 0;
        if(java && cpp)
            ERROR
    }
    str[n] = '\0';
    if(n&&!us)
        out << str;
    else
        out << "Error!";
    in.close();
    out.close();
    return 0;
}
```

Задача K - Kickdown

A research laboratory of a world-leading automobile company has received an order to create a special transmission mechanism, which allows for incredibly efficient kickdown — an operation of switching to lower gear. After several months of research engineers found that the most efficient solution requires special gears with teeth and cavities placed non-uniformly. They calculated the optimal flanks of the gears. Now they want to perform some experiments to prove their findings.

The first phase of the experiment is done with planar toothed sections, not round-shaped gears. A section of length n consists of n units. The unit is either a cavity of height h or a tooth of height $2h$. Two sections are required for the experiment: one to emulate master gear (with teeth at the bottom) and one for the driven gear (with teeth at the top).

There is a long stripe of width $3h$ in the laboratory and its length is enough for cutting two engaged sections together. The sections are irregular but they may still be put together if shifted along each other.

The stripe is made of an expensive alloy, so the engineers want to use as little of it as possible. You need to find the minimal length of the stripe which is enough for cutting both sections simultaneously.

Input

There are two lines in the input file, each contains a string to describe a section. The first line describes master section (teeth at the bottom) and the second line describes driven section (teeth at the top). Each character in a string represents one section unit — 1 for a cavity and 2 for a tooth. The sections can not be flipped or rotated.

Each string is non-empty and its length does not exceed 100.

Output

Write a single integer number to the output file — the minimal length of the stripe required to cut off given sections.

Sample input and output

kickdown.in	kickdown.out
2112112112 2212112	10
12121212 21212121	8
2211221122 21212	15

Алгоритм

Для решения этой задачи нужно подвигать шестеренки влево и вправо, проверить совпадение и выбрать ответ с наибольшим совпадением.

Исходный код

```
#include <iostream>
#include <fstream>

bool checkGears1(string master, string driven, int pos) {
    for (int i = 0; pos < master.length() && i < driven.length(); i++) {
        if (master[pos] == driven[i])
            if (master[pos] == '2')
                return false;
        pos++;
    }
    return true;
}

bool checkGears2(string master, string driven, int pos) {
    for (int i = 0; i < master.length() && pos < driven.length(); i++) {
        if (master[i] == driven[pos])
            if (master[i] == '2')
                return false;
        pos++;
    }
    return true;
}

int main() {
  ifstream in("kickdown.in");
    ofstream out("kickdown.out");
    string master, driven;
    in >> master >> driven;
    if (master.length() < driven.length()) swap(master, driven);

    int pos1 = 0;
    while (pos1 < master.length() && !checkGears1(master, driven, pos1)) pos1++;

    int pos2 = 0;
    while (pos2 < master.length() && !checkGears2(master, driven, pos2)) pos2++;

    int d1 = (int)(driven.length() + pos1 - master.length());
    int diff1 = (d1 >= 0) ? d1 : 0;
    int diff2 = pos2;

    if (diff1 < diff2) {
        out << max(master.length(), driven.length() + pos1) << endl;
    }
    else {
        out << master.length() + pos2 << endl;
    }

    in.close();
    out.close();

    return 0;
}
```

2.17 Codeforces Олимпиада школьников Нижегородской обл. Результаты

Задачи

№	Название			
A	Выравнивание вещественных чисел	стандартный ввод/вывод 2 с, 256 МБ		x154
B	Игра в 9	стандартный ввод/вывод 2 с, 256 МБ		x49
C	Преобразование числа	стандартный ввод/вывод 2 с, 256 МБ		x125
D	Марракеш	стандартный ввод/вывод 2 с, 256 МБ		x29
E	Мухи на плоскости	стандартный ввод/вывод 2 с, 256 МБ		x35
F	Фоторамка	стандартный ввод/вывод 2 с, 256 МБ		x141
G	Сны о скобках	стандартный ввод/вывод 2 с, 256 МБ		x70
H	Спуск полос	стандартный ввод/вывод 2 с, 256 МБ		x103
I	Изи	стандартный ввод/вывод 2 с, 256 МБ		x185
J	Затмение	стандартный ввод/вывод 2 с, 256 МБ		x1
K	Парковка	стандартный ввод/вывод 2 с, 256 МБ		x7
L	Пушка Гаусса	стандартный ввод/вывод 2 с, 256 МБ		x14

Ссылка на контест: http://codeforces.com/gym/100528

Задача A - Выравнивание вещественных чисел

Вася придумал новый алгоритм сортировки вещественных чисел, который, по его мнению, будет работать быстрее, даже чем QuickSort! Он очень хочет запатентовать его, но для этого сначала необходимо представить в патентное бюро работающую версию алгоритма. К сожалению, Вася не силен в языках программирования, поэтому попросил друзей реализовать различные куски алгоритма. Вам досталась часть программы, отвечающая за выравнивание чисел.

Задана последовательность A_i положительных вещественных чисел, не превосходящих 10^{1000}. Количество цифр после точки в представлении каждого числа не превосходит 1000. Требуется выписать все числа в исходном порядке в один столбец таким образом, чтобы все точки в их десятичной записи находились друг под другом. Для этого перед некоторыми числами необходимо приписать один или несколько символов #. Если решений несколько, то выведите то, в котором количество добавленных символов минимально.

Входные данные

В первой строке входных данных находится целое число N — количество чисел в последовательности A_i ($1 \le N \le 1000$). Последующие N строк содержат по одному вещественному числу последовательности A_i в десятичной записи. Числа не содержат ведущих нулей. В каждом числе присутствует десятичная точка.

Выходные данные

Выходные данные должны содержать ровно N строк, по одному числу из последовательности A_i в строке. Числа должны быть выровнены по точке десятичной записи с помощью #.

Примеры тестов

входные данные
3 3.1415926 12345.6789 2.71
выходные данные
####3.1415926 12345.6789 ####2.71

Алгоритм

Находим число с максимальным количеством цифр перед точкой. Добавляем к остальным числам спереди решётки чтобы перед точкой количество символов стало равно максимальному. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <cmath>
#define ll long long
#define ull unsigned long long
using namespace std;
int main()
{
    ll n;
    cin >> n;
    char num[1000][2010];
    int logs[1000];
    ll maxLog = 1;
    cin.get();
    for(ll i=0; i<n; ++i)
    {
        ll j=0;
        logs[i] = 0;
        bool pf = 0;
        while((num[i][j]=cin.get())!='\n')
        {
            if(num[i][j] == '.')
                pf = 1;
            if(!pf)
```

```
                ++logs[i];
            ++j;
        }
        num[i][j++] = '\0';
        if(logs[i]>maxLog)
            maxLog = logs[i];
    }
    for(ll i=0; i<n; ++i)
    {
        for(ll j=0; j<maxLog-logs[i]; ++j)
            cout.put('#');
        cout << num[i] << '\n';
    }
    return 0;
}
```

Задача F - Фоторамка

Вася выиграл в школьной викторине приз "— рамку для фотографий, причем не простую, а такую, в которую можно вставить сразу несколько фотографий. У Васиной рамки есть четыре места под фотографии, расположенные в ряд.

В век цифрового фото у Васи не так много бумажных фотографий, которые он мог бы вставить в рамку "— всего N штук. Мальчик планирует менять их состав каждый день, и ему интересно, как долго он сможет это делать. Для этого ему надо подсчитать, сколько есть способов заполнить рамку имеющимися у него фотографиями. Отметим, что Вася считает различными и способы, отличающиеся только порядком фотографий в рамке.

Входные данные
На единственной строке записано одно целое число N ($1 \le N \le 20$) "— количество фотографий, которые есть у Васи.

Выходные данные
Выведите единственное число "— количество различных способов заполнить Васиными фотографиями рамку.

Примеры тестов

входные данные
4
выходные данные
24

Алгоритм

В этой задаче можно заметить закономерность и предпосчитать ответ, так как всего может быть 20 различных входных данных. Таким образом, сложность составляет $O(1)$.

Исходный код

```
#include <iostream>
using namespace std;
int main() {
    long long m[]={0, 0, 0, 24, 120, 360, 840, 1680, 3024, 5040, 7920, 11880, 17160,
    24024, 32760, 43680, 57120, 73440, 93024, 116280};
    int n;
    cin >> n;
    cout << m[n-1];
    return 0;
}
```

Задача I - Изи

Вася вернулся с международной олимпиады школьников по программированию (IOI) и привез с собой N разноцветных камней в качестве сувениров. Вася совсем не жадный мальчик, поэтому решил поделиться камнями со своими друзьями. Каждому другу Вася отдал ровно один камень. Оказалось, что у самого Васи остался тоже только один камень. Определите, сколько же у Васи друзей?

Входные данные

В первой строке входного файла находится число N ($1 \le N \le 100$) — количество камней, привезенных Васей.

Выходные данные

Выведите единственное число — количество друзей Васи.

Примеры тестов

входные данные
2
выходные данные
1

Алгоритм

В задаче просто нужно вывести $n - 1$.

Исходный код

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <queue>
#include <climits>
#include <cctype>
#include <fstream>
#define ll long long
#define ull unsigned long long
using namespace std;
int main()
{
    ll n;
    cin >> n;
    cout << n-1;
    return 0;
}
```

2.18 OpenCup GrandPrix of Central Europe

Результаты

170.	MAI #11: Makarov, Rik, Yakimenko	+ 1:03	-	-1 3:08	-8 2:19	-	-	-	-	-	-	-4 3:57	1	63	0%	9.69	0.08

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10283

Задача A - Адвокат

Байтазар — известный адвокат, совладелец компании «Байтазар и товарищи». Также он участвует в заседаниях различных комитетов и комиссий. Неудивительно, что он всё время занят.

Каждый день Байтазара приглашают на множество различных заседаний; в какой-то момент он понял, что ему сложно понять, сможет ли он участвовать во всех заседаниях.

Поэтому Байтазар нанял секретаря, который должен решать эти вопросы. Байтазар решил, что каждый день он будет участвовать только в двух заседаниях, но с начала и до конца. На оставшиеся заседания Байтазар отправит своих помощников.

Считается, что заседания не перекрываются, если одно из них начинается строго после того, как завершится другое. Помогите секретарю Байтазара составить соответствующее расписание.

Input

Первая строка входного файла содержит два целых числа n и m ($2 \le n \le 500\,000$, $1 \le m \le 20$) — количество совещаний, на которые приглашён Байтазар и количество дней в его расписании. describing the number of meetings in Byteasar's schedule and the number of days included in it.

Каждая из последующих n строк описывает одну встречу. Описание встречи состоит из трёх целых чисел a_i, b_i, d_i ($1 \le a_i < b_i \le 80\,000\,000$, $1 \le d_i \le m$), обозначающих, что в день d_i Байтазар приглашён на встречу, которая начинается на a_i миллисекунде соответствующего дня и завершается на b_i миллисекунде.

Output

Выведите m строк. i-я из этих строк должна содержать информацию, сможет ли Байтазар в i-й день присутствовать на двух заседаниях. В случае, если это невозможно, выведите слово "NIE". Иначе выведите словл "TAK" и номер двух заседаний, на которых может присутствовать Байтазар в соответствующий день. Заседания пронумерованы в том порядке, в котором они описаны во входном файле, нумерация заседаний начинается с единицы. Второе заседание должно начаться как минимум на миллисекунду позже того, как закончилось первое.

Если существует несколько решений, выведите любое из них.

Examples

standard input	standard output
6 3 3 5 1 2 4 2 1 8 1 6 7 3 3 5 2 7 12 1	TAK 1 6 NIE NIE

Алгоритм

Для каждого дня находим встречу с минимальным времение конца и встречу с максимальным временем начала. Если время конца первой встречи меньше времени конца второй, то выведем TAK и номера встреч, если нет, то выведем NIE. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

int main()
{
    ll n, m;
    ll a, b, d;
    cin >> n >> m;
    vector< pair<ll, ll> > maxA(m, make_pair(-1, -1)), minB(m, make_pair(-1, -1));
    for (ll i=0; i<n; ++i) {
        cin >> a >> b >> d;
        --d;
        if (maxA[d].second == -1 || maxA[d].second < a) {
            maxA[d].second = a;
            maxA[d].first = i;
        }
        if (minB[d].second == -1 || minB[d].second > b) {
            minB[d].second = b;
            minB[d].first = i;
        }
    }
    for (ll i=0; i<m; ++i) {
        if (minB[i].second < maxA[i].second) {
            cout << "TAK " << minB[i].first+1 << ' ' << maxA[i].first+1 << '\n';
        }
        else {
            cout << "NIE\n";
        }
    }
    return 0;
}
```

2.19 Codeforces Training S02E10

Результаты

Задачи

№	Название			
A	Abnormal Coins	стандартный ввод/вывод 1 с, 256 МБ		x284
B	Fake Coins	стандартный ввод/вывод 1 с, 256 МБ		x194
C	Coin Graph	стандартный ввод/вывод 1 с, 256 МБ		x117
D	Coin Table	стандартный ввод/вывод 3 с, 256 МБ		x112
E	Volleyball	стандартный ввод/вывод 2 с, 256 МБ		x10
F	Huge Table	стандартный ввод/вывод 2 с, 256 МБ		x7
G	Coin Game	стандартный ввод/вывод 1 с, 256 МБ		x243
H	Dreams Were Important Too!	стандартный ввод/вывод 3 с, 256 МБ		x42
I	Coin Robbery	стандартный ввод/вывод 2 с, 256 МБ		x70
J	Bimetallic coins	стандартный ввод/вывод 2 с, 256 МБ		x46
K	Censorship!	стандартный ввод/вывод 2 с, 256 МБ		x10

Ссылка на контест: http://codeforces.com/gym/100534

Задача A - Abnormal Coins

If you have participated or solved any of the four previous AUT local contests, you may have noticed that first problem of each contest were all about abnormal new students of Amirkabir UT. This time we want to talk about coins in our problem-set but to respect traditions, let's start with **abnormal type of coins**.

Everybody knows what coins are, *"A flat, typically round piece of metal with an official stamp, used as money."*. But as wikipedia *"Not all coins are round. The Australian 50 cent coin, for example, has twelve flat sides. Some coins have wavy edges, e.g. the $2 and 20-cent coins of Hong Kong and the 10 cent coins of Bahamas. Some are square-shaped, such as the 15 cent coin of the Bahamas. During the 1970s, Swazi coins were minted in several shapes, including squares, polygons, and wavy edged circles with 8 and 12 waves."*

Steve is a great coin collector but he has no polygon shaped coin in his collection. Recently he found another coin lover, Johnny, Who has a collection of polygon coins and fortunately agreed to deal with Steve. The deal is simple, e coins for each poly-coin with e edges. Steve can offer n coin(s) and wants to earn as many poly-coin types as possible to extend his collection. Two poly-coins are different if they differ in number of edges.

Input

Input contains a single integer n. ($1 \leq n \leq 10^8$)

Output

Output a single integer indicating the maximum number of distinct poly-coins Steve can earn.

Examples

stdin	stdout
2	0
3	1
100000000	14139

Алгоритм

Будем считать сумму арифметической прогрессии с начальным членом равным 3 и разностью 1 и считать количество итераций в переменную *count*, пока сумма не станет больше числа n, тогда ответом будет значение в счётчике *count*. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#define LL long long
using namespace std;
int main() {
    LL n;
    cin >> n;
    LL count = 0;
    LL sum = 0;
    for (LL i = 3; ; i++) {
        sum += i;
        if (sum > n) break;
        count++;
    }
    cout << count << endl;
    return 0;
}
```

Задача B - Fake Coins

Cristiano, as a leader of his team, works in the Security Lab of Central Bank of Real Mars (SLCBRM). They want to improve security of coins that Central Bank makes. They want to use string codes that will be carved on coins to determine whether the coin is fake or original. Cris's team suggested him an amazing idea for managing security of coins.

They have a base string called B and a sequence of numbers called S. They select some characters from the base string to generate the security codes. If the i-th element of sequence is S_i Then the i-th element of generated code is the S_i-th character of B i.e.$(B[S_i])$. As you can clearly see the length of the generated security string code is equal to the number of elements in the sequence.

Their sequences are very similar to fibonacci numbers but their first and second elements are not always 1! They select two different numbers as first and second element of sequence (first number is always less than the second number). Then the third number of sequence is the sum of first and second numbers, the forth number is the sum of third and second number and so on. This process stops when a new element is greater than the length of the sequence. All elements of the sequence must be less than or equal to length of the base string.

Cris read the algorithm of generating string codes and realized that it can be some problems with same string codes. He wants to know how many different string codes can be generated using his team's algorithm. Because Criastiano is busy with scoring for Real Mars Football team he asked you to solve his problem!

Input

First and only line of input contains the base string $B(3 \leq len(B) \leq 1000)$. Base string is composed of digits, lowercase and uppercase Latin letter.

Output

Print one line containing the number of different security string codes that can be generated using Criastiano's team algorithm.

Examples

stdin	stdout
abba	5
abBA	6
FAKE0123456789original	217

Алгоритм

Нужно посчитать количество возможных кодов безопасности, которые могу быть сгенерированы. Для этого мы создадим ассоциативный массив и будем последовательно инкрементировать значения по строкам. Ответом будет размер ассоциативного массива. Сложность $O(n^2 * log(n))$.

Исходный код

```
#include <iostream>
#include <map>
#include <vector>
#define ll long long
#define ull unsigned long long
#define ERROR {out << "Error!"; return 0;}
using namespace std;
int main() {
    map<string, int> strings;
    string base, current = "";
    vector<int> spos;
    cin >> base;
    int baseSize = (int)base.size();
    for (int i = 1; i < baseSize; i++) {
        for (int j = i + 1; j <= baseSize; j++) {
            current += base[i - 1];
            current += base[j - 1];
            int cPos = i + j;
            int pPos = j, temp = 0;
            while (cPos <= baseSize) {
                current += base[cPos - 1];
                temp = pPos;
                pPos = cPos;
                cPos += temp;
            }
            strings[current]++;
            current.clear();
        }
    }

    cout << strings.size() << endl;

    return 0;
}
```

Задача G - Coin Game

Alice and Bob are very smart guys and they like to play all kinds of games in their spare time. The most amazing thing is that they always find the best strategy, and that's why they feel bored again and again. They just invented a new game, as they usually did.
They are playing with coins. They have a row of $1 and $2 coins. They want to change this row so that all $1's are grouped together and all $2's are grouped together. They are just allowed to swap two neighbor coins.
Using the best strategy what is the minimum number of swaps required to do this task?

Input

Each test case is a line with a string composed of 1 and 2.
$(1 \leq length\ of\ the\ input\ string\ \leq 25000)$

Output

Print the minimum number of swaps required to do this task.

Examples

stdin	stdout
221212	3

Алгоритм

Делаем обмены сначала слева направо, потом справа налево, сравниваем в который раз получилось меньше шагов и выводим это количество шагов. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
#define ll long long
int main()
{
    char a[25001], b[25001], c;
    int n = 0, first1 = -1, last1 = -1;
    while((c=cin.get())!='\n' && !cin.eof())
    {
        if(first1 == -1 && c == '1')
            first1 = n;
        if(c == '1')
            last1 = n;
        if(cin.eof())
            break;
        a[n] = b[n] = c;
        ++n;
    }
    a[n] = b[n] = '\0';
    ll aCount = 0, bCount = 0;
    for(ll i=0; i<n; ++i)
    {
        if(a[i] == '1')
            continue;
        for(ll j=i+1; j<n; ++j)
        {
            if(a[j] == '1')
            {
                swap(a[i], a[j]);
                aCount += j-i;
                break;
            }
        }
    }
    for(ll i=n-1; i>=0; --i)
    {
        if(b[i] == '1')
            continue;
        for(ll j=i-1; j>=0; --j)
        {
            if(b[j] == '1')
            {
                swap(b[i], b[j]);
                bCount += i-j;
                break;
            }
        }
    }
    if(aCount < bCount)
        cout << aCount;
    else
        cout << bCount;
    return 0;
}
```

2.20 OpenCup GrandPrix of Europe

Результаты

19.	MAI #11: Makarov, Rik, Yakimenko	-	-	+2 2:18	+4 4:53	-	-	+3 1:44	-	+ 0:36	+ 0:03	-2 2:52	5	755	64%	0.33

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10284

Задача E - Express As The Sum

Задано целое число N. Требуется представить его в виде суммы двух или более последовательных целых положительных чисел. Например, $10 = 1 + 2 + 3 + 4$

$24 = 7 + 8 + 9$

Если решений несколько, выведите то из них, которое состоит из наименьшего количества слагаемых.

Input

Первая строка входа содержит одно целое число T — количество тестовых примеров.

Каждый тестовый пример состоит из одной строки, содержащей целое число N ($1 \leq N \leq 10^9$).

Output

Для каждого тестового примера выведите одну строку, содержащую требуемое разложение в виде

`N = a + (a+1) + ...+ b`,

форматированное в соответствии с примером из условия. Если решения не существует, выведите текст "`IMPOSSIBLE`".

Example

standard input	standard output
3 8 10 24	IMPOSSIBLE 10 = 1 + 2 + 3 + 4 24 = 7 + 8 + 9

Алгоритм

Перебираем разложения для всех N пока не найдём нужное, начиная от 2. с помощью формулы N/i можно сразу найти одно из чисел, входящее в возможное разложение, а затем просто перебрать i вариантов. Сложность $O(n * log(n))$.

Исходный код

```
#include <iostream>
using namespace std;
#define ll long long
void answer(ll n, ll l, ll r) {
    cout << n << " = " << l;
    for(ll i=l+1; i<=r; ++i) {
        cout << " + " << i;
    }
    cout.put('\n');
}
int main() {
    ll T;
    cin >> T;
    while (T--) {
        ll n;
        cin >> n;
        ll t = n;
        bool f = 0;
        while(t!=0) {
            if (t>1 && t&1) {
                f = 1;
                break;
            }
            t >>= 1;
        }
        if(!f) {
            cout << "IMPOSSIBLE\n";
            continue;
        }
        f = 0;
        for (ll i=2; !f && i<100500; ++i) {
            ll sum = 0;
            ll r = n/i;
            ll l = r-i+1;
            if(l<1) {
                l = 1;
                r = l+i-1;
            }
            for(ll j=l; j<=r; ++j) {
                sum += j;
            }
            if(sum == n) {
                answer(n, l, r);
                break;
            }
            for(ll j=r+1; j<r+i; ++j) {
                sum += j-l;
                if(sum == n) {
                    answer(n, l+1, j);
                    f = 1;
                    break;
                }
                ++l;
            }
        }
    }
    return 0;
}
```

Задача F - Factory

Важный узел секретной машины состоит из n наношестерёнок, занумерованных последовательными целыми числами от 1 до n. Наношестерёнки отличаются от обычных шестерёнок тем, что размер их зубьев настолько мал, что шестерёнки можно считать обычными кругами. Каждая наношестерёнка крутится вокруг своего центра.

Никакие две наношестерёнки не перекрываются (не имеют общих внутренних точек), однако шестерёнки могут касаться друг друга. Если две шестерёнки касаются и одна из них вращается, другая тоже вращается, так как соответствующие зубья зацеплены.

К шестерёнке 1 приложена сила, заставлающая её вращаться со скоростью 1 оборот в минуту по часовой стрелке. Вычислите скорость вращения остальных шестерёнок. Гарантируется, что при вращении механизм не блокируется.

Input

Первая строка входа содержит одно целое число T — количество тестовых примеров.

Каждый тестовый пример начинается со строки, задающей количество наношестерёнок n ($1 \le n \le 1000$). Каждая из последующих n строк содержит три целых числа x, y и r ($-10\,000 \le x, y \le 10\,000$; $1 \le r \le 10\,000$), где (x, y) — координаты центра шестерёнки и r — её радиус.

Output

Для каждого тестового примера выведите n строк, описывающих вращение шестерёнок. Для каждой шестерёнки, выведите или "`p/q clockwise`", или "`p/q counterclockwise`", где несократимая дробь p/q задаёт количество оборотов наношестерёнки в минуьу. Если $q = 1$, выведите p как целое число. Если шестерёнка не вращается, выведите "`not moving`".

Example

standard input	standard output
1 5 0 0 6 6 8 4 -9 0 3 6 16 4 0 -11 4	1 clockwise 3/2 counterclockwise 2 counterclockwise 3/2 clockwise not moving

Алгоритм

Находим пересекающиеся шестерёнки и строим граф их связей. Проходим от первой шестерёнки ко всем остальным с помощью обхода в ширину. Направление вращения чередуется на каждом уровне. Зная радиусы двух смежных шестерёнок и скорость одной из них, легко можно вычислить скорость другой. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}

struct Gear {
    int x, y, r;
};

struct GearSpeed {
    ll n, d;
    bool dir;
};
int main()
{
    ll T;
    cin >> T;
    while (T--) {
        ll n;
        cin >> n;

        vector< list<int> > m(n);
        vector<bool> u(n, 0);
        vector<Gear> gears(n);
        for (ll i=0; i<n; ++i) {
            cin >> gears[i].x >> gears[i].y >> gears[i].r;
        }
        for (int i=0; i<n; ++i) {
            for (int j=i+1; j<n; ++j) {
                double x = gears[i].x-gears[j].x;
                double y = gears[i].y-gears[j].y;
                if(sqrt(x*x+y*y)<=gears[i].r+gears[j].r) {
                    m[i].insert(m[i].begin(), j);
                    m[j].insert(m[j].begin(), i);
                }
            }
        }
        vector<GearSpeed> speed(n);
        speed[0].n = 1;
        speed[0].d = 1;
        speed[0].dir = 0;
        queue<int> q;
        q.push(0);
        u[0] = 1;
        while (!q.empty()) {
            int a = q.front();
            q.pop();
```

```
            list<int>::iterator it = m[a].begin();
            for (; it!=m[a].end(); ++it) {
                int b = *it;
                if(u[b])
                    continue;
                u[b] = 1;
                speed[b].n = speed[a].n * gears[a].r;
                speed[b].d = speed[a].d * gears[b].r;
                speed[b].dir = !speed[a].dir;
                ll u = speed[b].n;
                ll v = speed[b].d;
                ll temp;
                while (v != 0) {
                    temp = u % v;
                    u = v;
                    v = temp;
                }
                speed[b].n /= u;
                speed[b].d /= u;
                q.push(b);
            }
        }
        for (ll i=0; i<n; ++i) {
            if(!u[i])
                cout << "not moving\n";
            else {
                cout << speed[i].n;
                if(speed[i].d != 1) {
                    cout << '/' << speed[i].d;
                }
                cout.put(' ');
                if(speed[i].dir) {
                    cout << "counterclockwise\n";
                }
                else {
                    cout << "clockwise\n";
                }
            }
        }
    }
    return 0;
}
```

Задача K - Keyboard Troubles

Вы только что проснулись и хотите включить микроволновку на k секунд. Вы должны набрать число k на клавиатуре, имеющей следующий вид:

Так как по сути, вы не совсем проснулись, вы можете набрать число только в случае, если ваши руки будут двигаться вниз или вправо. Вы не можете двигаться влево или вверх, но вы можете нажимать одну и ту же кнопку многократно.

Выведите число, которое Вы можете набрать и которое наиболее близко к k. Если решений несколько, выведите любое из двух.

Input

Первая строка содержит одно целое число T — количество тестовых примеров.

Каждый тестовый пример содержит одно целое число k ($1 \leq k \leq 200$) — требуемое количество минут.

Output

Для каждого тестового примера в отдельной строке выведите ближайшее к k число, которое может быть введено в соответствии с условием задачи.

Examples

standard input	standard output
3 180 83 132	180 80 133

Алгоритм

Для этой задачи пришлось заранее предпосчитать массив с правилами перехода между кнопками. Сложность $O(n)$.

Исходный код

```
#include <iomanip>
#include <limits>
#include <iostream>
#include <algorithm>

#define LL  long long
#define ULL unsigned long long
#define EPS 1e-11

#define more_speed ios_base::sync_with_stdio(false);

using namespace std;

bool g(int x, int y) {

    if (x == y) return true;

    if (x == 1) {
        return true;
    }
    else if (x == 2) {
        if (y == 3 || y == 5 || y == 6 || y == 8 || y == 9 || y == 0) return true;
    }
    else if (x == 3) {
        if (y == 6 || y == 9) return true;
    }
    else if (x == 4) {
        if (y >= 4 || y == 0) return true;
    }
    else if (x == 5) {
        if (y == 6 || y == 8 || y == 9 || y == 0) return true;
    }
    else if (x == 6) {
        if (y == 9) return true;
    }
    else if (x == 7) {
        if (y >= 7 || y == 0) return true;
    }
    else if (x == 8) {
        if (y == 9 || y == 0) return true;
    }
    return false;
}

int main() {

    more_speed

    int z[] =
    {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1,1,0,1,1,0,0,0,1,0,0,1,0,0,1,1,0,0,0,

    int t;
```

```
    cin >> t;

    for (int i = 0; i < t; i++) {
        int n;
        cin >> n;
        if (z[n] == 1) {
            cout << n << endl;
        }
        else {
            int y = n;
            int x = n;
            while (z[x] == 0) {
                x++;
            }
            while (z[y] == 0) {
                y--;
            }
            if (n - y > x - n) {
                cout << x << endl;
            }
            else {
                cout << y << endl;
            }
        }
    }

    return 0;
}
```

Задача O - Allo

Задан телефонный номер, составленный из цифр от 0 до 9. Найдите все цифры, которые не используются в этом номере.

Input

Первая строка входа содержит одно целое число T — количество тестовых примеров ($1 \le T \le 150$). Каждая из последующих T строк задаёт один тестовый пример — непустую строку, состоящую из цифр от 0 до 9. Длина заданной строки не превосходит 30.

Output

Для каждого тестового примера выведите в возрастающем порядке все числа, которые не используются в соответствующей строке. Если все цифры используются, то выведите текст "`allo`".

Examples

standard input	standard output
2 2468 0123456789	013579 allo

Алгоритм

Просто отмечаем все цифры, которые встретились в номере в булевском массиве. Потом проходим по этому массиву и проверяем остались ли неотмеченные. Если остались, то выводим их, если нет, то выводим *allo*. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;
#define ll long long
#define ull unsigned long long
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}
int main()
{
    ll T;
    cin >> T;
    cin.get();
    while (T--) {
        vector<bool> nums(10, 0);
        char c;
        while((c=cin.get())!='\n' && !cin.eof())
        {
            if(cin.eof())
                break;
            nums[c-'0'] = 1;
        }
        bool f = 0;
        for(ll i=0; i<10; ++i)
        {
            if(!nums[i])
            {
                f = 1;
                cout << i;
            }
        }
        if(!f)
            cout << "allo";
        cout << '\n';

    }
    return 0;
}
```

Задача N - C--

Язык C-- был разработан специально для студентов, которые не справилсь со стандартным курсом по компиляторам.

Программа на C-- представляет собой список команд. Существуют следующие команды:

-
- { — начало блока;
- } — конец блока;
- int *var* — описание переменной *var*;
- var_1=var_2 — присвоить значение переменной var_2 переменной var_1;
- *var*=*constant* — присвоить переменной *var* значение *constant*;
- print *var* — вывести значение *var*.

Все имена переменных состоят из одной строчной латинской буквы. Константы представляют собой целые числа в диапазоне от 0 до 10^9. Пробелы допустимы только после операторов "int" и "print".

Каждая переменная в блоке описывается не более одного раза, однако переменные с одним и тем же именем могут быть описаны в различных блоках. В этом случае имя задаёт переменную, описанную в ближайшем открытом блоке. Переменные могут быть описаны в середине блока.

Гарантируется, что программа корректна, то есть баланс и вложенность скобок соблюдены, обращение к переменным происходит только после их описания, перед чтением переменные всегда инициализированы и так далее. Программа всегда содержит как минимум один оператор "print".

Напишите программу, исполняющую код на языке C--.

Input

Первая строка входа содержит целое число N ($3 \le N \le 1000$). Последующие N строк содержат команды языка C--, по одному на строку.

Output

Для каждой команды "print" в отдельной строке выведите печатаемое этой командой значение.

Example

standard input	standard output

10
int a
int x
{
a=50
int a
a=60
x=a
print x
}
print a
	60
	50

Алгоритм

Задача на реализацию. При объявлении переменной, она добавляется в стек в виде структуры *pair*, в которой первый элемент - значение переменной, а второй - номер области видимости. При выходе из текущей области видимости, все переменный, созданные в ней, удаляются из стека. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;
#define ll long long
int main()
{
    ll n;
    vector < stack< pair<ll , int> > > vars(26);
    int scope = 0;
    cin >> n;
    char str[20];
    for (int i=0; i<n; ++i) {
        cin >> str;
        if(str[0] == '{') {
            ++scope;
        }
        else if(str[0] == '}') {
            for (int i=0; i<26; ++i) {
                if(!vars[i].empty() && vars[i].top().second == scope) {
                    vars[i].pop();
                }
            }
            --scope;
        }
        else if(str[1] == '=') {
            int c1 = str[0]-'a';
            ll a;
            if(isalpha(str[2])) {
                int c2 = str[2]-'a';
                a = vars[c2].top().first;
                int c1Scope = vars[c1].top().second;
                vars[c1].pop();
                vars[c1].push(make_pair(a, c1Scope));
            }
            else {
                a = 0;
                for (int i=2; str[i]!='\0'; ++i)
                    a = a*10+str[i]-'0';
            }
            int c1Scope = vars[c1].top().second;
            vars[c1].pop();
            vars[c1].push(make_pair(a, c1Scope));
        }
        else if(!strcmp(str, "int")) {
            cin >> str;
            int c;
            c = str[0]-'a';
            vars[c].push(make_pair(-1, scope));
        }
        else if(!strcmp(str, "print")) {
            cin >> str;
            int c = str[0]-'a';
            cout << vars[c].top().first << '\n';
        }
    }
    return 0;
}
```

2.21 OpenCup GrandPrix of Peterhof

Результаты

31.	MAI #11: Makarov, Rik, Yakimenko	-	-	-	-	-	-	-	+ 3:24	-	-	-	1	204	0%	0.27

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10285

Задача H - Некратчайший путь

В этой задаче нужно пройти из одного угла клетчатого поля 4×4 *в другой по любому простому пути, не являющемуся кратчайшим.*

На плоскости нарисовано квадратное клетчатое поле из 4×4 клеток. Каждая клетка поля либо свободна, либо целиком занята стеной.

Робот Аврелий проходит тест на измерение уровня интеллекта, решая различные задачи на этом поле. Он уже успешно справился с поиском произвольного и кратчайшего пути из одной клетки в другую.

В очередном задании Аврелию следует пройти из левого верхнего угла поля в правый нижний по свободным клеткам. За один шаг из любой клетки можно переместиться в любую другую клетку, имеющую с ней общую сторону. При этом путь должен быть простым, то есть в одной и той же клетке нельзя оказываться дважды. Кроме того, длина пути должна быть строго больше, чем длина кратчайшего пути между этими углами на заданном поле. Длиной пути считается количество шагов в нём.

Напишите программу, которая позволит роботу Аврелию справиться с заданием.

Формат входных данных

На входе задан один или несколько тестовых случаев. Каждый тестовый случай задаётся на четырёх строках. Каждая из этих строк описывает один ряд поля и содержит ровно четыре символа, задающих клетки этого ряда. Пустая клетка обозначается символом «`.`» (точка, ASCII-код 46), а стена — символом «`#`» (решётка, ASCII-код 35). Соседние тестовые случаи отделяются друг от друга строкой, состоящей из четырёх символов «`-`» (минус, ASCII-код 45).

Гарантируется, что левая верхняя и правая нижняя клетки поля пусты. Также гарантируется, что тестовые случаи на входе не повторяются в пределах одного теста.

Формат выходных данных

В ответ на каждый тестовый случай выведите строку, определяющую путь. Строка должна состоять из символов, соответствующих командам перемещения в соседнюю клетку поля, в порядке их применения: «`D`» для движения вниз, «`U`» для движения вверх, «`L`» для движения влево и «`R`» для движения вправо. Если возможных путей несколько, выведите любой из них. Если же пути с нужными свойствами не существует, выведите вместо него число «`-1`».

Пример

nsp.in	nsp.out
.#.. .... ..#. .#.. ---- ..## ...# #... .#.. ---- ...# ..#. .#.. #...	DDRURURDDD -1 -1

Алгоритм

С помощью рекурсии находим все возможные пути и составляем из них вектор. Находим среди них кратчайший и после этого находим путь, который не является кратчайшим, если же не находим, то выводим -1. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}

vector < vector<char> > paths;
vector<char> path;
size_t __min;

void foo(char map[4][4], int x, int y) {
    if (x == 3 && y == 3) {
        if (path.size() < __min) {
            __min = path.size();
        }
        paths.push_back(path);
    }
    if (x<3 && map[y][x+1] == '.') {
        path.push_back('R');
        map[y][x] = 'X';
        foo(map, x+1, y);
        map[y][x] = '.';
        path.pop_back();
    }
    if (x>0 && map[y][x-1] == '.') {
        path.push_back('L');
        map[y][x] = 'X';
        foo(map, x-1, y);
        map[y][x] = '.';
        path.pop_back();
    }
    map[y][x] = '.';
    if (y<3 && map[y+1][x] == '.') {
        path.push_back('D');
        map[y][x] = 'X';
        foo(map, x, y+1);
        map[y][x] = '.';
        path.pop_back();
    }
    map[y][x] = '.';
    if (y>0 && map[y-1][x] == '.') {
        path.push_back('U');
        map[y][x] = 'X';
        foo(map, x, y-1);
        map[y][x] = '.';
        path.pop_back();
    }
}
```

```
53
54 int main()
55 {
56     char map[4][4];
57     bool e = 1;
58     while (e) {
59         for (int i=0; i<4; ++i) {
60             for (int j=0; j<4; ++j) {
61                 map[i][j] = cin.get();
62             }
63             cin.get();
64         }
65         __min = 100500;
66         foo(map, 0, 0);
67         bool f = 0;
68         for (int i=0; i<paths.size(); ++i) {
69             if (paths[i].size() != __min) {
70                 for (int j=0; j<paths[i].size(); ++j) {
71                     cout.put(paths[i][j]);
72                 }
73                 f = 1;
74                 break;
75             }
76         }
77         if (!f) {
78             cout << -1;
79         }
80         cout.put('\n');
81         for (int i=0; i<5; ++i) {
82             cin.get();
83             if (cin.eof()) {
84                 e = 0;
85                 break;
86             }
87         }
88         paths.clear();
89         path.clear();
90     }
91     return 0;
92 }
```

2.22 OpenCup GrandPrix of Japan

Результаты

16.	MAI #11: Makarov, Rik, Yakimenko	-2 4:56	-	-	-	-	-	+4 3:50	+ 0:21	+ 0:38	+1 1:07	4	457	55%	0.20

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10286

Задача K - Beads

Задано ожерелье из N жемчужин, пронумерованых последовательными целыми числами от 1 до N по часовой стрелке. Каждая жемчужина помечена заглавной латинской буквой. Таким образом, если прочитать эти буквы последовательно по часовой стрелке, получится строка длины N. Так как ожерельне замкнуто, то в зависимости от того, с какой жемчужины начинать чтение, можно получить различные строки.

Требуется найти среди этих строк лексикографически наименьшую.

Input

Первая строка входа содержит целое число N ($1 \le N \le 2 \cdot 10^6$) — длину ожерелья. Во второй строке задано само ожерелье. Ожерелье задаётся строкой, содержащей N заглавных латинских букв; k-я буква — это буква, написанная на k-й жемчужине.

Output

Выведите номер жемчужины, начиная с которой, можно прочитать по часовой стрелке лексикографически наименьшую строку длиной N. Если ответов несколько, выведите наименьший.

Example

standard input	standard output
6 CABCAB	2

Алгоритм

Для того чтобы решить задачу, нужно найти минимальный циклический сдвиг, для этого мы применим алгоритм Дюваля. Сложность $O(n)$.

Исходный код

```
#include <iostream>

using namespace std;

long min_cyclic_shift (string s) {
    s += s;
    long n = (long) s.length();
    long i = 0, ans = 0;
    while (i < n/2) {
        ans = i;
        long j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j])
                k = i;
            else
                ++k;
            ++j;
        }
        while (i <= k)  i += j - k;
    }
    return ans;
}

int main(int argc, const char * argv[]) {
    long n;
    string s;
    cin >> n >> s;
    long answer = min_cyclic_shift(s);
    cout << answer + 1 << endl;
    return 0;
}
```

Задача L - The Maximum Sum

Заданы N целых положительных чисел. Среди них выбираются два, сумма которых является наибольшей, не превосходящей заданного числа M. Требуется найти значение соответствующей суммы.

Input

Первая строка входа содержит два целых положительных числа N ($3 \leq N \leq 100$) и M ($1 \leq M \leq 100$). Вторая строка содержит N целых положительных чисел.

Output

Выведите одно число — требуемую сумму.

Examples

standard input	standard output
5 8 5 3 4 6 5	8
4 116 31 52 73 84	115

Алгоритм

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

Исходный код

```
#include <iomanip>
#include <iostream>
#include <algorithm>
int main() {
    int n, M;
  cin >> n >> M;
  vector <int> num;
  int temp;
  for (int i = 0; i < n; i++) {
    cin >> temp;
    num.push_back(temp);
  }
  int max_sum = 0;
  for(int i = 0; i < n; i++)
    for(int j = i + 1; j < n; j++) {
      temp = num[i] + num[j];
      if(temp <= M && temp > max_sum)
        max_sum = temp;
    }
  cout << max_sum << endl;
    cin >> n;
    return 0;
}
```

Задача M - Spellcheck

Вася написал спеллчекер, который выдаёт ошибки в следующих случаях:

- "`u`" или "`ur`" вместо "`you`" или "`your`".
- "`would of`", "`should of`" вместо "`would have`", "`should have`".
- "`lol`". Более того, программа выдаёт ошибку на любое слово, в котором можно прочитать "`lol`" как подслово (если слово содержит "`lol`" несколько раз, как словл "`ololo`", всё равно выдаётся одна ошибка).

Напишите программу, которая считывает предложения одно за другим и для каждого предложения определяет, сколько ошибок найдёт новый спеллчекер.

Input

Первая строка входа содержит целое число T ($1 \le T \le 50$) — количество предложений в тесте. Каждая из последующих T строк содержит одно предложение — одно или несколько слов, разделённых пробелами. Слова состоят из строчных латинских букв. Слова разделены ровно одним пробелом, пробелы в начале и в конце предложения отсутствуют. Общая длина каждого предложения (вместе с пробелмаи) не превосходит 100 символов.

Output

Для каждого предложения в отдельной строке выведите, сколько ошибок найдёт в нём новый спеллчекер.

Examples

standard input	standard output
5 r u haz trololo my name is vasya i got the lollipop u should of lollollol i should off line	2 0 1 3 0

Алгоритм

Просто сравним все входные слова через $strcmp$, и если будет совпадение, то инкрементируем счётчик ошибок для строки. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    long T;
    cin >> T;
    char aStr[100], bStr[100];
    cin.get();
    while (T--) {
        char c;
        long a = 0, aN = 0, bN = 0, res = 0;
        while (c = cin.get()) {
            if (c == ' ' || c == '\n') {
                if (a&1) {
                    if (!strcmp("u", bStr) || !strcmp("ur", bStr) || strstr(bStr, "lol") != NULL) {
                        ++res;
                    }
                    else if((!strcmp("would", aStr) || !strcmp("should", aStr)) && !strcmp("of", bStr)) {
                        ++res;
                    }
                    aN = 0;
                }
                else {
                    if (!strcmp("u", aStr) || !strcmp("ur", aStr) || strstr(aStr, "lol") != NULL) {
                        ++res;
                    }
                    else if(a && (!strcmp("would", bStr) || !strcmp("should", bStr)) && !strcmp("of", aStr)) {
                        ++res;
                    }
                    bN = 0;
                }
                if (c == '\n') {
                    break;
                }
                ++a;
                continue;
            }
            if (a&1) {
                bStr[bN++] = c;
                bStr[bN] = '\0';
            }
            else {
                aStr[aN++] = c;
                aStr[aN] = '\0';
            }
        }
        cout << res << '\n';
    }
    return 0;
}
```

Задача N - Bluetooth

Вы пытаетесь отправить сообщение другу через Bluetooth. Смартфон может соединяться по Bluetooth с любым смартфоном, если расстояние между ними не превышает D.

Определите, можно ли отправить сообщение «по цепочке» (напрямую или через некоторое количество промежуточных смартфонов).

Input

Первая строка входа содержит два целых числа N ($1 \le N \le 10$) и D ($1 \le D \le 10$), где N — общее количество смартфонов с Bluetooth в помещении, а D — максимальное расстояние, на котором два смартфона соединяются по Bluetooth.

Каждая из следующих N строк содержит по два целых числа — координаты X и Y очередного смартфона. При этом первая строка задаёт координаты Вашего смартфона, последняя — координаты смартфона Вашего друга ($0 \le X, Y \le 100$).

Output

Выведите '`y`', если сообщение передать можно и '`n`' в противном случае.

Example

standard input	standard output
4 7 1 4 6 2 9 7 14 4	y
5 6 7 1 5 5 1 6 8 7 20 15	n

Алгоритм

Составляем граф связей между смартфонами. Затем с помощью поиска в ширину пытаемся добраться до смартфона друга. Если получилось, то выводим y, если нет, то n. Сложность $O(n^2 + n)$.

Исходный код

```
#include <iomanip>
#include <limits>
#include <iostream>
#include <algorithm>
double dist(const pair<int, int> &a, const pair<int, int> &b) {
    double x, y;
    x = (a.first - b.first);
    y = (a.second - b.second);
    x *= x;
    y *= y;
    return sqrt(x + y);
}
int main() {
    int n, d;
    cin >> n >> d;
    vector<pair<int, int> > points(n);
    for (int i = 0; i < n; i++) {
        cin >> points[i].first >> points[i].second;
    }
    vector<vector<int> > g(n, vector<int>());
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i != j) {
                double dis = dist(points[i], points[j]);
                if (dis <= d) {
                    g[i].push_back(j);
                }
            }
        }
    }
    int s = 0;
    queue<int> q;
    q.push(s);
    vector<bool> used (n);
    used[s] = true;
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (size_t i = 0; i < g[v].size(); ++i) {
            int to = g[v][i];
            if (!used[to]) {
                if (to == n - 1) {
                    cout << "y" << endl;
                    return 0;
                }
                used[to] = true;
                q.push (to);
            }
        }
    }
    cout << "n" << endl;
    return 0;
}
```

2.23 OpenCup Northern GrandPrix

Результаты

30.	MAI #11: Makarov, Rik, Yakimenko	-	-	-	-	-	-	+7 3:03	-	+ 0:39	-	2	362	77%	0.20

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10287

Задача K - Kill The PSU

Вам поручена разработка системы контроля стабильности работы устройств и их драйверов. Система должна выдавать подсказки пользователю в зависимости от типа устройства, с которым произошёл сбой.

Устройства с точки зрения этой системы делятся на следующие три категории (каждое устройство принадлежит только к одной из них).

- Категория 1: Внешняя периферия. Проблемы этих устройств заключаются в том, что время от времени они переходят в режим экономии питания (засыпают). В случае, если это устройство встретилось в логе, Вы должны рекомендовать пользователю разбудить (wake) его.
- Категория 2: Внутренние устройства с нестабильными драйверами. При появлении проблем с такими устройствами требуется загрузить (load) драйвер, если он не был загружен, или выгрузить в противном случае. Так как драйверы являются экспериментальными, то в момент начальной загрузки система их не загружает.
- Категория 3: Основные устройства. Проблемы этих устройств обычно вызваны нехваткой питания. Соответственно, реакцией на появление этого устройства в логе должна быть выдача соответствующего сообщения, использование резервной мощности блока питания (если она ещё не использована), а в крайнем случае, когда питания устройству не хватает совсем — выдача рекомендации о замене блока питания и остановка системы.

  Считается, что каждому основному устройству идёт 100 ватт от блока питания в момент старта системы. Появление основного устройства в логе обозначает, что на него стало поступать на 10 ватт меньше. Первоначально у блока питания есть резерв в 20 ватт; так что первые два сбоя должны быть компенсированы; далее же подаваемая на устройства мощность начинает уменьшаться. В случае, когда на устройство подаётся не более 10 ватт, Ваша программа должна выдать рекомендацию о покупке нового блока питания и остановить систему; все последующие сообщения в логе (если таковые есть) обрабатываться уже не должны.

По заданному списку имён устройств в системе, категориям, к которым они принадлежат и логу сбоев выведите требуемые рекомендации для пользователя.

Input

Первая строка входа содержит одно целое число T — количество тестовых примеров ($0 \le T \le 100$). Далее заданы сами тестовые примеры.

Первая строка каждого тестового примера состоит из четырёх целых чисел A, B, C, D, разделённых пробелами. Первые три числа задают количество устройств в первой, второй и третьей категориях соответственно, последнее задаёт количество событий в логе.

Далее следуют A строк, содержащие имена устройств из первой категории, B строк, содержащих имена устройств из второй категории, C строк, содержащих имена устройств из третьей категории, и D строк, задающих лог сбоев; каждая запись в логе представляет собой имя устройства, с которым возникли проблемы ($0 \le A, B, C, D \le 100$, имена устройств состоят из строчных латинских букв и пробелов, начинаются и заканчиваются строчной латинской буквой и имеют длину не менее 2 и не более 22).

Output

Для каждого тестового примера выведите рекомендацию системы по каждому из сбоев в порядке их появления в логе.

Для устройств первой категории выведите "`wake` (имя устройства)". Для устройств второй категории выведите "`load` (имя устройства)", если драйвер не был загружен (в случае первого после загрузки системы появления этого устройства в логе или после выгрузки) и "`unload` (имя устройства)" в противном случае.

Для устройств третьей категории выведите "`power fail on` (имя устройства)", если на устройство подаётся более 10 ватт. В противном случае выведите "`buy the new PSU`" и завершите обработку данного тестово-

го примера вне зависимости от того, остались ли события в логе. В случае неясности с форматов вывода следуйте формату из тестовых примеров.

Examples

killthepsu.in	killthepsu.out
2 1 1 2 5 video card wireless network south bridge cpu video card wireless network wireless network south bridge cpu 0 0 1 11 motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard motherboard	wake video card load wireless network unload wireless network power fail on south bridge power fail on cpu power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard power fail on motherboard buy the new PSU

Алгоритм

Задача на реализацию. Нужно построить что-то на подобии экспертной системы по заданным правилам. Сложность $O(n)$.

Исходный код

```
#include <iomanip>
#include <iostream>
#include <algorithm>
#include <vector>
#include <fstream>
#define LL  long long
using namespace std;
ifstream in("killthepsu.in");
ofstream out("killthepsu.out");

class item {
public:
    string name;
    virtual void message() = 0;
};
class cat1 : public item {
public:
    void message() {
        out << "wake " << name << endl;
    }
    cat1(string n) {
        name = n;
    }
};
class cat2 : public item {
public:
    bool load;
    void message() {
        if (load) {
            out << "unload ";
            load = false;
        }
        else {
            out << "load ";
            load = true;
        }
        out << name << endl;
    }
    cat2(string n) {
        name = n;
        load = false;
    }
};
int reserve = 20;
bool nextIter = false;
class cat3 : public item {
public:
    int power;
    void message() {
        if (reserve) {
            reserve -= 10;
            out << "power fail on " << name << endl;
            return;
        }
        power -= 10;
        if (power > 10) {
            out << "power fail on " << name << endl;
        }
```

```
        else if (power <= 10) {
            out << "buy the new PSU" << endl;
            nextIter = true;
        }
    }
    cat3(string n) {
        power = 100;
        name = n;
    }
};

int main() {
    int t;
    in >> t;
    for (int k = 0; k < t; k++) {
        reserve = 20;
        map<string, item *> system;
        int a, b, c, d;
        in >> a >> b >> c >> d;
        string name;
        in.get();

        for (int i = 0; i < a; i++) {
            getline(in, name);
            item *newItem = new cat1(name);
            system[name] = newItem;
        }
        for (int i = 0; i < b; i++) {
            getline(in, name);
            item *newItem = new cat2(name);
            system[name] = newItem;
        }
        for (int i = 0; i < c; i++) {
            getline(in, name);
            item *newItem = new cat3(name);
            system[name] = newItem;
        }
        for (int i = 0; i < d; i++) {
            getline(in, name);
            if (!nextIter) {
                map<string, item *>::iterator current = system.find(name);
                if (current != system.end())
                    current->second->message();
            }
        }
        nextIter = false;
    }
    in.close();
    out.close();
    return 0;
}
```

Задача M - Мозаика

Мальчик Костя очень любит не только собирать мозаики, но и с удовольствием придумывает различные игры во время складывания их обратно в коробку.

Мозаика имеет форму прямоугольника $M \times N$, каждая клетка которого содержит одну плитку. На каждой плитке написано целое число от 1 до $M \cdot N$ (на всех плитках числа различны). В собранном виде мозаика имеет следующий вид:

1	2	3	$\dots$	$N-1$	N
$N+1$	$N+2$	$N+3$	$\dots$	$2 \cdot N-1$	$2 \cdot N$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$(M-1) \cdot N+1$	$(M-1) \cdot N+2$	$(M-1) \cdot N+3$	$\dots$	$M \cdot N-1$	$M \cdot N$

Костя хочет сыграть в игру, в которой требуется убрать как можно больше плиток мозаики по определенным правилам. За один ход можно убрать любую плитку, у которой есть 4 соседних плитки (плитки считаются соседними, если они находятся в соседних клетках в одной строке или в одном столбце).

Напишите программу, которая определит максимально возможное количество плиток, которые можно убрать по описанным выше правилам. А также найдите какой-либо корректный максимальный набор плиток.

Input

В единственной строке входных данных записаны два целых числа M и N ($1 \le M, N \le 100$) — высота и ширина мозаики.

Output

В первой строке выведите максимальное количество плиток, которые Костя может убрать по описанным выше правилам. Во второй строке выведите номера плиток в порядке, в котором их следует убирать. Если существует несколько решений, выведите любое из них.

Examples

mosaic.in	mosaic.out
3 3	1 5
2 1	0
4 4	2 6 11
4 5	3 7 9 13

Алгоритм

Если N или M меньше трёх, то ответ будет 0, если нет, то вычисляем максимальное количество плиток по выведенной формуле и заполняем мозаику. Сложность $O(N * M)$.

Исходный код

```
#include <iomanip>
#include <limits>
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
    ifstream in("mosaic.in");
    ofstream out("mosaic.out");

  int N, M;
  in >> M >> N;
  int answ;

  if(N < 3 || M < 3) {
    out << "0" << endl;
    return 0;
  }
  if((N - 2)*(M - 2) % 2 == 0)
    answ = (N - 2)*(M - 2) / 2;
  else
    answ = (N - 2)*(M - 2) / 2 + 1;

  out << answ << endl;

  int k = 2;
  for(int i = 2; i < M; i++) {
    for(int j = k; j < N; j+=2) {
      out << (i - 1)*N + j << " ";
    }
    if(k == 2)
      k = 3;
    else k = 2;
  }

    in.close();
    out.close();

    return 0;
}
```

2.24 OpenCup GrandPrix of Karelia

Результаты

36.	MAI #11: Makarov, Rik, Yakimenko	-	-	-	-	+2 1:26	+ 0:46	+6 3:47	+ 0:35	4	555	66%	0.18

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10288

Задача I - Jam

Два программиста сидят в автомобиле, который едет за автобусом в пробке. На остановке два человека зашли в автобус, а три вышли. После чего один из программистов заметил, что если ещё один человек зайдёт в автобус, то автобус будет пустым.

Так как пробка была длинной, то программисты сделали несколько наблюдений, на каждой следующей остановке записывая количество вошедших и вышедших пассажиров.

Ваша задача — написать программу, которая вычислит наименьшее количество пассажиров в автобусе перед тем, как автомобиль программистов попал в пробку.

Input

Первая строка входного файла содержит одно целое число T $(1 \leq T \leq 50)$ — количество тестовых примеров.

Каждый тестовый пример начинается строкой, содержащей целое число M $(1 \leq M \leq 100)$ — количество остановок. Далее следует M строк, содержащих результаты наблюдений. Каждое наблюдение задано двумя целыми числами P_1 и P_2, разделёнными пробелом — количество человек, вошедших в автобус на соответствующей остановке, и количество человек, покинувших автобус, соответственно $(0 \leq P_1, P_2 \leq 1000)$. При этом сначала P_1 человек входят в автобус, затем двери «на выход» открываются и P_2 человек выходят из него.

Output

Для каждого тестового примера выведите в отдельной строке минимальное количество пассажиров, находившихся в салоне автобуса до того, как программисты попали в пробку.

Example

jam.in	standard output
1 3 4 6 5 6 2 0	3

Алгоритм

На каждой остановке смотрим, если вышло человек больше чем вошло, то прибавляем к ответу разницу между вошедшими и вышедшими. Выводим ответ. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <algorithm>

int main() {
    ifstream in("jam.in");

    int T;
    in >> T;

    while (T--) {
        int m, zashli, vishli;
        long ans = 0;
        long buf = 0;
        in >> m;
        for (int i = 0; i < m; i++) {
            in >> zashli >> vishli;
            zashli += buf;
            int diff = zashli - vishli;
            if (diff < 0) {
                ans += diff * (-1);
                buf = 0;
            }
            else {
                buf = diff;
            }
        }
        cout << ans << endl;
    }
    in.close();
    return 0;
}
```

Задача J - King of Guess

Игра «Угадай число» играется следующим образом: загадано целое число N и заданы два целых числа X и Y, строго между которыми оно находится. Ход состоит в том, что участник называет некоторое целое число между X и Y, соответственно, ответы могут быть «больше», «меньше» и «угадал» (после ответа «угадал» игра заканчивается). В случае, если число не угадано, одно из чисел X или Y заменяется на названное число (в зависимости от ответа), и игра продолжается.

Пусть участники всякий раз называют среднее целое число в текущем интервале (если таких чисел два, они называют наименьшее). За какое количество ходов закончится игра?

Input

Первая строка входа содержит три целых числа N, X и Y ($0 \le N, X, Y \le 10^4$, $X < Y$).

Output

Выведите одно число — количество ходов, за которое закончится игра.

Examples

kingofguess.in	standard output
42 20 80	3

Алгоритм

Задача решается прямой симуляцией процесса игры по довольно простым правилам, заданным в условии задачи. Сложность $O(log(n))$.

Исходный код

```
#include <iostream>
#include <algorithm>

int main() {
    ifstream in("kingofguess.in");
  int X;
  int N;
  int Y;
  int mid;
  int step = 0;

  in >> N >> X >> Y;

  while(true) {
    step++;
    mid = (X + Y)/2;
    if(mid == N) {
      cout << step << endl;
      //cin >> N;
      return 0;
    }
    if(mid > N)
      Y = mid;
    if(mid < N)
      X = mid;
  }
    return 0;
}
```

Задача K - Lesson

Алиса и Боб играют в «Морской бой» на уроке в школе. Игра идёт по следующим правилам:

У каждого игрока есть поле $N \times N$, на котором он располагает четыре неперекрывающихся корабля; каждый корабль является цепочкой клеток, параллельных одной из сторон поля. Требуется разместить по одному из следующих кораблей:

Name	Length
Sail	1
Frigate	2
Cruiser	3
Dreadnought	4

После этого игроки делают ходы по очереди, называя клетки на игровом поле. Если клетка, названная игроком X, занята на игровом поле игрока Y кораблём, игрок Y сообщает, что корабль ранен; если же игрок X своим ходом назвал последнюю из не названных им ранее клеток корабля, то игрок Y сообщает, что корабль потоплен, после чего игрок X получает дополнительный ход; если же игрок X своим ходом не потопил корабль игрока Y, то очередь хода переходит к игроку Y. Игра заканчивается, когда один из игроков потопил все корабли противника; в этом случае тот игрок, у которого остались корабли, объявляется победителем.

Чтобы учитель не заметил происходящего, Алиса и Боб модифицировали игру: сейчас после расстановки кораблей они записывают на бумаге последовательность ходов; разумеется, все ходы корректны и один и тот же игрок не записывает одно и то же поле дважды (ибо незачем).

По заданным ходам каждого игрока выведите, какие корабли и в каком порядке будут потоплены и кто в итоге выиграет, если первый ход делает Алиса.

Input

Первая строка входного файла содержит целое число T — количество тестовых примеров ($1 \le T \le 20$).

Далее следуют тестовые примеры. Каждый тестовый пример начинается строкой, содержащей целое число N ($4 \le N \le 10$) — размер игрового поля. Далее следуют N строк, задающих расстановку кораблей на поле Алисы. Каждая из этих строк состоит из 4 символов. Это один из пяти символов '.', '1', '2', '3' или '4'. Точка ('.') обозначает, что данная клетка пуста. '1', '2', '3' или '4' обозначают, что эта клетка занята одним из кораблей Алисы. Клетки с одинаковыми числами принадлежат одному кораблю; корабли параллельны сторонам поля. Обратите внимание, что цифра, обозначающая корабль, не обязана совпадать с его длиной.

Следующие N строк задают расстановку кораблей на поле Боба в аналогичном формате. Следующие $N \cdot N$ строк задают ходы Алисы. Каждая из строк содержит два целых числа R_i и C_i — строка и столбец, которые она планирует называть для этого хода (если до этого хода дойдёт очередь). Следующие $N \cdot N$ строк задают ходы Боба в аналогичном формате ($1 \le R_i, C_i \le N$).

Output

Для каждого тестового примера выведите для каждого потопленного корабля сообщение в формате "`PlayerX sank PlayerY's ShipName`", в порядке, в котором корабли были потоплены. Последняя строка вывода для каждого тестового примера должна содержать имя победителя.

Алгоритм

Нужно воспроизвести игру "Морской бой"по ходам обоих игроков. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}

int main()
{
    ifstream in("lesson.in");

    long T;
    in >> T;
    in.get();
    const string names[] = {"Sail", "Frigate", "Cruiser", "Dreadnought"};
    while (T--) {
        ll n;
        in >> n;
        in.get();
        char mapA[10][10], mapB[10][10];
        vector< pair<int, int> > A(4, make_pair(0, 0)), B(4, make_pair(0, 0));
        for(ll i=0; i<n; ++i) {
            for (ll j=0; j<n; ++j) {
                mapA[i][j] = in.get();
                if (mapA[i][j] != '.')
                    ++A[mapA[i][j]-'0'-1].first;
            }
            in.get();
        }
        for(ll i=0; i<n; ++i) {
            for (ll j=0; j<n; ++j) {
                mapB[i][j] = in.get();
                if (mapB[i][j] != '.')
                    ++B[mapB[i][j]-'0'-1].first;
            }
            in.get();
        }
        vector< pair<ll, ll> > movesA(n*n), movesB(n*n);
        for (ll i=0; i<n*n; ++i) {
            ll x, y;
            in >> y >> x;
            movesA[i] = make_pair(y - 1, x - 1);
        }
        for (ll i=0; i<n*n; ++i) {
            ll x, y;
            in >> y >> x;
            movesB[i] = make_pair(y - 1, x - 1);
        }
        ll shipsA = 4, shipsB = 4;
        bool win = 0;
        for (ll moveA = 0, moveB = 0; moveA < n*n && moveB < n*n && !win;) {
            ll y, x;
            bool f = 1;
```

```
            while (f) {
                f = 0;
                x = movesA[moveA].second;
                y = movesA[moveA].first;
                ++moveA;
                if (mapB[y][x] != '.') {
                    ++B[mapB[y][x]-'0'-1].second;
                    if (B[mapB[y][x]-'0'-1].second == B[mapB[y][x]-'0'-1].first) {
                        cout << "Alice sank Bob's " << names[B[mapB[y][x]-'0'-1].
    first-1] << '\n';
                        f = 1;
                        --shipsB;
                        if (!shipsB) {
                            cout << "Alice\n";
                            win = 1;
                            break;
                        }
                    }
                }
            }
            if (win) {
                break;
            }
            f = 1;
            while (f) {
                f = 0;
                x = movesB[moveB].second;
                y = movesB[moveB].first;
                ++moveB;
                if (mapA[y][x] != '.') {
                    ++A[mapA[y][x]-'0'-1].second;
                    if (A[mapA[y][x]-'0'-1].second == A[mapA[y][x]-'0'-1].first) {
                        cout << "Bob sank Alice's " << names[A[mapA[y][x]-'0'-1].
    first-1] << '\n';
                        f = 1;
                        --shipsA;
                        if (!shipsA) {
                            cout << "Bob\n";
                            win = 1;
                            break;
                        }
                    }
                }
            }
            if (win) {
                break;
            }
        }
    }

    in.close();
    return 0;
}
```

Задача L - Maze

На некотором астероиде был найден лабиринт ровно с одним входом, не содержащий циклов и пустот, окружённых стенами. Для исследования лабиринта был отправлен робот.

Робот всегда смотрит в том направлении, в котором он движется. На каждом шаге робот будет пытаться повернуть направо. Если на этом месте стена, он будет пытаться сделать шаг вперёд. Если и это невозможно — пытаться повернуть налево, если же и это невозможно — развернуться назад.

Робод записывает лог, который содержит его путь от момента входа в лабиринт до момента выхода. Передвижения записаны единичными буквами: '`F`' для движения вперёд, '`L`' для движения налево, '`R`' для движения направо и '`B`' для движения назад.

Каждая из букв '`L`', '`R`' и '`B`' обозначает не только поворот робота, но и продвижение на одно поле в данном направлении. Изначально робот ориентирован на восток. Путь робота всегда заканчивается в точке выхода.

Вам задан лог. Требуется по нему восстановить план лабиринта.

Input

Первая строка входа содержит одно целое число T — количество тестовых примеров ($1 \le T \le 100$). Каждый тестовый пример представляет собой одну строку — корректный лог пути робота. Гарантируется, что исходный лабиринт имел размеры не более, чем 100×100.

Output

Для каждого тестового примера выведите в первой строке два целых числа h и w ($3 \le h, w \le 100$) — высоту и ширину лабиринта, соответственно. Далее выведите h строк, каждая из которых содержит w символов, задающих лабиринт, где '`X`' обозначает стену, а '`.`' — пустое поле.

Контур лабиринта должен состоять из стен, за исключением одного квадрата с левой стороны — входа в лабиринт. Лабиринт не должен содержать циклов (то есть путей, по которым робот сможет придти в какую-то клетку не с той стороны, с которой в неё входил) и пустых полей, которые не могут быть достигнуты от входа. Каждая строка и каждый столбец (кроме верхней и нижней строки и правого столбца) должны содержать как минимум одно пустое поле.

maze.in	standard output
3 FFRBLF FFRFRBRFBFRBRFLF FRLFFFLBRFFFRFFFRFRFBRFLBRFRLFLFFR	4 4 XXXX ...X XX.X XXXX 7 5 XXXXX ...XX XX.XX X...X XX.XX XX.XX XXXXX 7 7 XXXXXXX X...X.X X.X...X X.X.XXX ..XXX.X X.....X XXXXXXX

Алгоритм

Создаём матрицу, изначально заполненную решётками. Воспроизводим путь робота согласно входным символам и отмечаем путь точками, в процессе запоминая максимальную и минимальную координаты. Используя найденные граничные координаты, вычисляем размер поля и выводим само поле. Сложность $O(n)$.

Исходный код

```
#include <iostream>
using namespace std;
int main() {
    ifstream in("maze.in");
    long T;
    in >> T;
    in.get();
    while (T--) {
        char map[210][210];
        for (ll i=0; i<210; ++i) {
            for (ll j=0; j<210; ++j) {
                map[i][j] = 'X';
            }
        }
        ll maxX = 0, maxY = 0, minX = 0, minY = 0, x = 0, y = 0;
        int dir = 0;
        char c;
        while((c = in.get()) != '\n') {
            if (c == 'B') {
                if (dir == 0) dir = 2;
                else if (dir == 1) dir = 3;
                else if (dir == 2) dir = 0;
                else if (dir == 3) dir = 1;
            }
            else if (c == 'R') {
                if (dir == 0) dir = 3;
                else if (dir == 1) dir = 0;
                else if (dir == 2) dir = 1;
                else if (dir == 3) dir = 2;
            }
            else if (c == 'L') {
                if (dir == 0) dir = 1;
                else if (dir == 1) dir = 2;
                else if (dir == 2) dir = 3;
                else if (dir == 3) dir = 0;
            }
            map[x+105][y+105] = '.';
            if (dir == 0) ++x;
            else if (dir == 1) --y;
            else if (dir == 2) --x;
            else ++y;
            if (x > maxX) maxX = x;
            else if(x < minX) minX = x;
            if (y > maxY) maxY = y;
            else if(y < minY) minY = y;
        }
        cout << maxY - minY + 3 << ' ' << maxX - minX + 2 << '\n';
        for (ll i=minY-1; i<maxY+2; ++i) {
            for (ll j=minX; j<maxX+2; ++j) {
                cout.put(map[j+105][i+105]);
            }
            cout.put('\n');
        }
    }
    return 0;
}
```

2.25 OpenCup GrandPrix of Udmurtia

Результаты

42.	MAI #11: Makarov, Rik, Yakimenko	+ 3:00	-	-	-	-	-	+ 1:49	+ 1:21	+2 4:00	-1 0:30	4	651	33%	0.18

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10289

Задача A - Коллекционеры

Вчера Тая посещала музей. Экскурсия была долгой и интересной, но особенно ей понравилась комната, где были собраны коллекции кубиков 10 известных людей. Один из кубиков ей сильно приглянулся, но она забыла, кому он принадлежал. Зато она помнит как выглядели 3 видимые стороны кубика и кто какие кубики коллекционирует. По этой информации вам следует определить имена коллекционеров, в чьих коллекциях мог быть данный кубик.

У каждого кубика 6 граней. На каждой грани записано число от 1 до 6, на разных гранях разные числа. Числа могут изображаться точками, арабским числом или римским числом. Вдобавок каждая из граней покрашена в один из цветов — черный (Black), белый (White), зеленый (Green), желтый (Yellow), голубой (Skyblue), красный (Red), оранжевый (Orange) и фиолетовый (Purple).

Ниже представлен список имен коллекционеров и параметры кубиков, которым удовлетворяет вся его коллекция:

John	Все числа обозначены точками
David	Все числа обозначены не римскими числами
Peter	Кубик полностью белый
Robert	Грани кубика черные или белые
Mark	Нечетные числа на белом фоне, четные — на черном
Paul	Все простые числа арабские и все арабские числа простые
Patrick	Одноцветный цветной (не черный и не белый) кубик
Jack	Все римские числа на желтом фоне
Max	Все грани разноцветные
Alex	Числа одной формы записи на одном фоне, разного — на разном

Формат входных данных

Входной файл состоит их трех строк, описывающих видимые стороны кубика.

Первый символ i-й строки c_i ($c_i \in \{B, W, G, Y, S, R, O, P\}$) — цвет i-ой стороны (черный, белый, зеленый, желтый, голубой, красный, оранжевый и фиолетовый соответственно). Далее через пробел записана строка, обозначающая число на стороне в одном из трех форматов:

1. От 1 до 6 символов «`.`» (ASCII 46), означающих что число записано точками и равно количеству этих точек;
2. Арабское число от 1 до 6;
3. Римское число, записанное символами «`I`» (ASCII 73) и «`V`» (ASCII 86).

Гарантируется, что представленный кубик принадлежит хотя бы одному коллекционеру.

Формат выходных данных

Выходной файл должен содержать одну строку, содержащую имена коллекционеров, которым может принадлежать данный кубик. Имена должны быть записаны через пробел в любом порядке.

Все имена коллекционеров должны быть из следующего списка: `John`, `David`, `Peter`, `Robert`, `Mark`, `Paul`, `Patrick`, `Jack`, `Max`, `Alex`.

Примеры

input.txt	output.txt
W .. W ... W	John David Peter Robert Jack Alex
B 2 W 3 B 6	David Robert Mark Jack
G 1 G 2 G V	Patrick
G 2 G 3 Y	David Paul Jack Alex
W . B 2 W III	Robert Mark

Алгоритм

Логическая задача на реализацию. Расписываем все условия из задачи для каждого человека и затем последовательно применяем их ко входным данным. Сложность $O(1)$.

Исходный код

```
#include <iostream>
#include <algorithm>

typedef enum {
    BLACK,
    WHITE,
    GREEN,
    YELLOW,
    BLUE,
    RED,
    ORANGE,
    PURPLE
} color;

typedef enum {
    DOT,
    ARABIC,
    ROMAN,
} numberType;

typedef struct {
    color col;
    numberType num;
    int value;
} cubeFace;

color parseColor(char col) {
    if (col == 'B') return BLACK;
    else if (col == 'W') return WHITE;
    else if (col == 'G') return GREEN;
```

```
    else if (col == 'Y') return YELLOW;
    else if (col == 'S') return BLUE;
    else if (col == 'R') return RED;
    else if (col == 'O') return ORANGE;
    else return PURPLE;
}

int parseDots() {
    int value;
    string dots;
    in >> dots;
    value = (int)dots.size();
    return value;
}

int parseArabic() {
    int value;
    in >> value;
    return value;
}

int parseRoman() {
    string roman;
    in >> roman;
    if (roman == "I") return 1;
    else if (roman == "II") return 2;
    else if (roman == "III") return 3;
    else if (roman == "IV") return 4;
    else if (roman == "V") return 5;
    else return 6;
}

int parseValue(numberType num) {
    if (num == DOT) return parseDots();
    else if (num == ARABIC) return parseArabic();
    else return parseRoman();
}

numberType recognizeNumberType(char num) {
    if (num == '.') return DOT;
    else if (isdigit(num)) return ARABIC;
    else return ROMAN;
}

bool john(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.num == DOT && f1.num == f2.num && f2.num == f3.num)
        return true;
    return false;
}

bool david(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.num != ROMAN && f2.num != ROMAN && f3.num != ROMAN)
        return true;
    return false;
}

bool peter(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.col == WHITE && f1.col == f2.col && f2.col == f3.col)
        return true;
    return false;
```

```
}

bool robert(cubeFace f1, cubeFace f2, cubeFace f3) {
    if ((f1.col == BLACK || f1.col == WHITE) &&
        (f2.col == BLACK || f2.col == WHITE) &&
        (f3.col == BLACK || f3.col == WHITE))
        return true;
    return false;
}

bool mark(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.value % 2 == 0 && f1.col != BLACK)
        return false;
    if (f1.value % 2 == 1 && f1.col != WHITE)
        return false;
    if (f2.value % 2 == 0 && f2.col != BLACK)
        return false;
    if (f2.value % 2 == 1 && f2.col != WHITE)
        return false;
    if (f3.value % 2 == 0 && f3.col != BLACK)
        return false;
    if (f3.value % 2 == 1 && f3.col != WHITE)
        return false;
    return true;
}

bool paul(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.value == 2 || f1.value == 3 || f1.value == 5) {
        if (f1.num != ARABIC)
            return false;
    }
    if (f2.value == 2 || f2.value == 3 || f2.value == 5) {
        if (f2.num != ARABIC)
            return false;
    }
    if (f3.value == 2 || f3.value == 3 || f3.value == 5) {
        if (f3.num != ARABIC)
            return false;
    }
    if (f1.num == ARABIC) {
        if (f1.value == 1 || f1.value == 4 || f1.value == 6) {
            return false;
        }
    }
    if (f2.num == ARABIC) {
        if (f2.value == 1 || f2.value == 4 || f2.value == 6) {
            return false;
        }
    }
    if (f3.num == ARABIC) {
        if (f3.value == 1 || f3.value == 4 || f3.value == 6) {
            return false;
        }
    }
    return true;
}

bool patrick(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.col == f2.col && f2.col == f3.col) {
        if (f1.col != BLACK && f1.col != WHITE)
```

```
            return true;
    }
    return false;
}

bool jack(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.num == ROMAN && f1.col != YELLOW)
        return false;
    if (f2.num == ROMAN && f2.col != YELLOW)
        return false;
    if (f3.num == ROMAN && f3.col != YELLOW)
        return false;
    return true;
}

bool maxx(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.col != f2.col && f1.col != f3.col && f2.col != f3.col)
        return true;
    return false;
}

bool alex(cubeFace f1, cubeFace f2, cubeFace f3) {
    if (f1.num == f2.num && f1.col != f2.col)
        return false;
    if (f1.num != f2.num && f1.col == f2.col)
        return false;

    if (f1.num == f3.num && f1.col != f3.col)
        return false;
    if (f1.num != f3.num && f1.col == f3.col)
        return false;

    if (f2.num == f3.num && f2.col != f3.col)
        return false;
    if (f2.num != f3.num && f2.col == f3.col)
        return false;

    return true;
}

int main() {

    char col, num;

    // FACE 1
    cubeFace face1;
    col = in.get();
    in.get();

    num = in.get();
    in.unget();

    face1.col = parseColor(col);
    face1.num = recognizeNumberType(num);
    face1.value = parseValue(face1.num);
    in.get();

    // FACE 2
    cubeFace face2;
    col = in.get();
```

```
    in.get();

    num = in.get();
    in.unget();

    face2.col = parseColor(col);
    face2.num = recognizeNumberType(num);
    face2.value = parseValue(face2.num);
    in.get();

    // FACE 3
    cubeFace face3;
    col = in.get();
    in.get();

    num = in.get();
    in.unget();

    face3.col = parseColor(col);
    face3.num = recognizeNumberType(num);
    face3.value = parseValue(face3.num);
    in.get();

    if (john(face1, face2, face3))
        out << "John ";
    if (david(face1, face2, face3))
        out << "David ";
    if (peter(face1, face2, face3))
        out << "Peter ";
    if (robert(face1, face2, face3))
        out << "Robert ";
    if (mark(face1, face2, face3))
        out << "Mark ";
    if (paul(face1, face2, face3))
        out << "Paul ";
    if (patrick(face1, face2, face3))
        out << "Patrick ";
    if (jack(face1, face2, face3))
        out << "Jack ";
    if (maxx(face1, face2, face3))
        out << "Max ";
    if (alex(face1, face2, face3))
        out << "Alex ";

    in.close();
    out.close();

    return 0;
}
```

Задача K - Data Mining

Одна крупная IT-компания использует для анализа данных программу, которая была написана без распараллеливания. Когда требуется запустить эту программу, сотрудник запускает её и ждёт результата выполнения, тратя своё рабочее время.

Руководство компании рассматривает план, заключающийся в переписывании программы с использованием параллельных технологий. Чтобы проверить, окупится ли решение за год, используется следующая схема.

Сначала выясняется количество запусков программы в год, затем выясняется количество человеко-часов, которые уйдут на переписывание, затем оценивается время работы для параллельной версии программы, после чего проверяется, будет ли достигнута экономия в человеко-часах.

Выясните, будет ли эффективным переписывание программы в течение ближайшего года (то есть будет ли суммарное количество человеко-часов, затраченное на переписывание программы и на суммарное ожидание результата выполнения параллельной версии меньше количества человеко-часов, затраченных на суммарное ожидание результата выполнения существующей версии).

Формат входных данных

В первой строке входного файла задано целое число T — количество тестовых примеров ($1 \leqslant T \leqslant 1000$).

Каждый тестовый пример задан в одной строке и содержит четыре целых числа d, n, p и s — количество человеко-часов на разработку, количество запусков программы в год, время ожидания завершения работы параллельной версии, время ожидания завершения работы существующей версии соответственно. Время задаётся в часах, $0 \leqslant d \leqslant 10^6$, $0 \leqslant n \leqslant 10^5$, $0 \leqslant p, s \leqslant 1000$.

Формат выходных данных

Для каждого тестового примера выведите “`Rewrite`”, если переписывание программы даст выигрыш в человеко-часах, “`Keep`”, если оно даст проигрыш и “`Flip a Coin`”, если количество затраченных человеко-часов не зависит от решения о переписывании.

Пример

input.txt	output.txt
3 11 3 3 2 19 6 9 3 0 5 100 100	Keep Rewrite Flip a Coin

Алгоритм

Подсчитываем время выполнения программы без её переписывания и с переписыванием и сравниваем. Сложность $O(1)$.

Исходный код

```
#include <iostream>
#include <algorithm>

int main() {
    ifstream in("input.txt");
    ofstream out("output.txt");

    long T;
    in >> T;
    while (T--) {
        long dev, runs, execParallel, execCurrent;
        in >> dev >> runs >> execCurrent >> execParallel;
        long keepTime, rewriteTime;
        keepTime = runs * execCurrent;
        rewriteTime = dev + runs * execParallel;
        if (keepTime < rewriteTime) {
            out << "Keep" << endl;
        }
        else if (keepTime > rewriteTime) {
            out << "Rewrite" << endl;
        }
        else {
            out << "Flip a Coin" << endl;
        }
    }

    in.close();
    out.close();

    return 0;
}
```

Задача L - Performance

Производительность автомобильного двигателя с несколькими передачами для заданной передачи не является постоянной: она зависит от количества оборотов двигателя в минуту. Обычно эта зависимость задаётся так называемой кривой производительности.

Для серии двигателей "Параболика" кривая производительности представляет собой параболу $P = -aR^2 + bR + c$, где R — количество оборотов двигателя в секунду, а P — соответствующая производительность.

По заданным параметрам парабол, описывающих все передачи в двигателе, определите, на какой передаче достигается максимум производительности.

Гарантируется, что такая передача ровно одна и что значение максимальной производительности для этой параболы отличается от значений максимальной производительности для других парабол не менее, чем на 1.

Формат входных данных

В первой строке входного файла задано целое число T ($1 \leqslant T \leqslant 100$) — количество тестовых примеров.

Каждый тестовый пример начинается строкой, содержащей целое число n ($1 \leqslant n \leqslant 10$) — количество передач в двигателе. Далее следуют n строк, каждая из которых содержит по три целых числа a, b и c ($1 \leqslant a, b, c \leqslant 10^4$) — параметры соответствующей параболы. Передачи занумерованы с 1 в порядке, в котором они перечислены во входном файле.

Формат выходных данных

Для каждого тестового примера выведите одно целое число — номер передачи, позволяющей достичь максимума производительности.

Примеры

input.txt	output.txt
1 2 3 130 1423 2 153 210	2

Алгоритм

Вычисляем квадратное уравнение, выведенное из заданной формулы и находим передачу с максимальной производительностью. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <algorithm>

typedef struct {
    long gear;
    long a, b, c;
} eq;

int main() {
    ifstream in("input.txt");
    ofstream out("output.txt");

    int T;
    in >> T;
    while (T--) {
        long maxGear = 0;
        double maxValue = -1000000;
        int n;
        in >> n;
        for (int i = 1; i <= n; i++) {
            long a, b, c;
            in >> a >> b >> c;
            double max = (-1) * (((b * b) - (4 * (-a) * c)) / (4 * (-a)));
            if (max > maxValue) {
                maxValue = max;
                maxGear = i;
            }
        }
        out << maxGear << endl;
    }

    in.close();
    out.close();

    return 0;
}
```

Задача M - Tic-tac-toe

Дана недоигранная партия в крестики-нолики на доске 3×3. Всего сделано 7 ходов. Вычислите, у кого из игроков выше вероятность выигрыша, если игрок, делающий следующий ход, сделает его случайным образом.

Формат входных данных

Входной файл состоит из 3 строк, задающих игровое поле. Каждая строка состоит из трёх символов; допустимыми символами являются ('o', 'x' или '.', обозначающий свободную клетку). На доске осталось ровно две свободные клетки; гарантируется, что партия корректна (то есть разность между количеством крестиков и ноликов по модулю равна единице и не существует тройки одинаковых символов по горизонтали, вертикали и диагонали).

Формат выходных данных

Выведите 'o', если играющий ноликами имеет лучшие шансы на победу, 'x', если лучшие шансы на победу у крестиков, и "tie", если шансы равны.

Примеры

input.txt	output.txt
xo. oxo xx.	x

Алгоритм

Перебираем все возможные варианты ходов и подсчитываем кто из игроков выигравает в большинстве ходов. Сложность $O(n)$.

Исходный код

```
#include <iostream>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}

struct Point{
    int x, y;
};

int check(char map[3][3]) {
    int winner = 2;
    for (int i=0; i<3; ++i) {
        bool oFH = 1, xFH = 1, oFV = 1, xFV = 1;
        for (int j=0; j<3; ++j) {
            if (map[i][j] != 'o') {
                oFH = 0;
            }
            if (map[i][j] != 'x') {
                xFH = 0;
            }
            if (map[j][i] != 'o') {
                oFV = 0;
            }
            if (map[j][i] != 'x') {
                xFV = 0;
```

```
            }
        }
        if (oFH || oFV) {
            winner = 1;
        }
        else if (xFH || xFV) {
            winner = 0;
        }
    }
    bool oFD1 = 1, xFD1 = 1, oFD2 = 1, xFD2 = 1;
    for (int i=0; i<3; ++i) {
        if (map[i][i] != 'o') {
            oFD1 = 0;
        }
        if (map[i][i] != 'x') {
            xFD1 = 0;
        }
        if (map[i][2-i] != 'o') {
            oFD2 = 0;
        }
        if (map[i][2-i] != 'x') {
            xFD2 = 0;
        }
    }

    if (oFD1 || oFD2) {
        winner = 1;
    }
    else if (xFD1 || xFD2) {
        winner = 0;
    }
    return winner;
}
int main()
{
    ifstream in("input.txt");
    ofstream out("output.txt");

    Point freeCell[2];
    int n = 0;
    int xN = 0, oN = 0;
    char map[3][3];
    for (int i=0; i<3; ++i) {
        for (int j=0; j<3; ++j) {
            map[i][j] = in.get();
            if (map[i][j] == '.') {
                freeCell[n].x = j;
                freeCell[n].y = i;
                ++n;
            }
            if (map[i][j] == 'x') {
                ++xN;
            }
            else if (map[i][j] == 'o') {
                ++oN;
            }
        }
        in.get();
    }
    char firstPlayer, secondPlayer;
```

```
    if (oN < xN) {
        firstPlayer = 'o';
        secondPlayer = 'x';
    }
    else {
        firstPlayer = 'x';
        secondPlayer = 'o';
    }
    map[freeCell[0].y][freeCell[0].x] = firstPlayer;
    int winner = check(map);
    float pX = 0, pO = 0;
    if (firstPlayer == 'o' && winner == 1 || firstPlayer == 'x' && winner == 0) {
        pO += 1;
    }
    else {
        map[freeCell[1].y][freeCell[1].x] = secondPlayer;
        winner = check(map);
        if (firstPlayer == 'o' && winner == 0 || firstPlayer == 'x' && winner == 1) {
            pX += 1;
        }
    }

    map[freeCell[0].y][freeCell[0].x] = '.';
    map[freeCell[1].y][freeCell[1].x] = firstPlayer;
    winner = check(map);
    if (firstPlayer == 'o' && winner == 1 || firstPlayer == 'x' && winner == 0) {
        pO += 1;
    }
    else {
        map[freeCell[0].y][freeCell[0].x] = secondPlayer;
        winner = check(map);
        if (firstPlayer == 'o' && winner == 0 || firstPlayer == 'x' && winner == 1) {
            pX += 1;
        }
    }

    if (pO > pX) {
        if (firstPlayer == 'o')
            out << "o";
        else
            out << "x";
    }
    else if (pO < pX) {
        if (secondPlayer == 'x')
            out << "x";
        else
            out << "o";
    }
    else {
        out << "tie";
    }
    in.close();
    out.close();
    return 0;
}
```

2.26 OpenCup GrandPrix of China

Результаты

46.	MAI #11: Makarov, Rik, Yakimenko	-	-	-	-	-	-	-11 4:51	-1 4:09	-2 3:32	+ 0:19	1	19	0%	0.08

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10300

Задача N - Ordered Sequences

Рассмотрим строки, составленные из символов 'A' и 'B', такие, что первым символом является 'B'. Упорядочим их сначала по длине (меньшие идут раньше), затем лексикографически и занумеруем, начиная с единицы.

По заданной последовательности найдите её номер.

Input

Первая строка входа содержит целое число $1 \le n \le 1000$ — количество тестовых примеров.

Каждая из последующих n строк содержит непустую строку, состоящую из 'A' и 'B' и начинающуюся с 'B'. Длина строки не превосходит 24.

Output

Для каждого тестового примера выведите номер соответствующей строки в соответствии с введённой в задаче нумерацией.

Example

standard input	standard output
3 B BA BB	1 2 3

Алгоритм

Представим последовательность символов $'A'$ и $'B'$ как двоичное число в котором единица - это $'B'$, а ноль - это $'A'$. Тогда ответом будет просто значение этого числа в десятичном представлении. Сложность $O(n)$.

Исходный код

```
#include <iostream>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}

int main()
{
    int T;
    cin >> T;
    cin.get();
    while (T--) {
        ull a = 0;
        char c;
        while ((c=cin.get())!='\n') {
            a <<= 1;
            if (c == 'B') {
                a |= 1;
            }
        }
        cout << a << '\n';
    }
    return 0;
}
```

2.27 OpenCup GrandPrix of Tatarstan

Результаты

62.	MAI #11: Makarov, Rik, Yakimenko	-	-	-	-	-	-5 4:22	+ 0:23	-	-3 4:36	-	-1 1:11	1	23	0%	0.01

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10301

Задача M - The Dress

После высадки на планете i1c5l люди заметили, что среди аборигенов очень популярна сине-черная одежда. У каждого аборигена в гардеробе есть хотя бы один сине-черный предмет. Этот факт мог бы и не заинтересовать людей, если бы не одно «но» — аборигены все как один утверждали, что это не сине-черная одежда, а бело-золотая.

Таким образом появился простой тест, однозначно отличающий инопланетянина от человека. На одной из межвидовой свадеб аборигена и симпатичной землянки люди сделали фотографию сине-черного платья матери жениха. Эту фотографию люди показывали тестируемым и просили назвать цвет платья. Если в ответе содержалось «blue» и «black», то это однозначно говорило о том, что тестируемый — человек с Земли. Если в ответе содержалось «white» и «gold», то это указывало на существо с планеты i1c5l. Если же ответ не удовлетворял ни одному из условий, то тестируемый был залётным пришельцем с другой планеты.

Перед вами полная история опросов живых существ, проведённого на планете i1c5l. Ваша задача — определить состав населения планеты на основе проведённого опроса.

Input

В первой строке записано единственное целое число N — количество опрошенных существ ($1 \le N \le 100$). Следующие N строк содержат ответы опрошенных. Каждая строка не пуста, имеет длину не более 100 символов и задаёт ответ одного из опрошенных. Ответ состоит из строчных букв латинского алфавита и пробелов. Гарантируется, что ни один ответ не содержит одновременно «blue», «black»,«white» и «gold».

Output

Выведите три числа, описывающих население планеты, каждое на отдельной строке.

Первое число — количество землян, выраженное в процентах по отношению к общему количеству опрошенных.

Второе число — количество аборигенов с планеты i1c5l, выраженное в процентах по отношению к общему количеству опрошенных.

Третье число — количество залётных пришельцев, выраженное в процентах по отношению к общему количеству опрошенных.

Все числа выведите с точностью не менее 10^{-5}.

Examples

стандартный ввод	стандартный вывод
3 goldandwhite white and pinkman bluebluebluе and a little bit black	33.3333333333 33.3333333333 33.3333333333
4 this dress is blue and black this dress is goldblackblue eto plate kazhetsia sirenevenkoe no comments	50.0000000000 0.0000000000 50.0000000000

Алгоритм

Делаем поиск в строке по ключевым словам и подсчитываем количество существ для каждой группы. Затем выводим в процентах. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <algorithm>

int main(){
    int n;
    cin >> n;
    int humans = 0;
    int aliens = 0;
    int fuckers = 0;
    int total = n;
    string answer;
    cin.get();

    while (n--) {
        getline(cin, answer);
        if (answer.find("blue") != -1 && answer.find("black") != -1) {
            humans++;
        }
        else if (answer.find("gold") != -1 && answer.find("white") != -1) {
            aliens++;
        }
        else {
            fuckers++;
        }
    }

    cout << setprecision(10) << (double)humans / total * 100 << endl;
    cout << setprecision(10) << (double)aliens / total * 100 << endl;
    cout << setprecision(10) << (double)fuckers / total * 100 << endl;
    return 0;
}
```

2.28 OpenCup GrandPrix of America

Результаты

34.	MAI #11: Makarov, Rik, Yakimenko	-	-	-	-	-	+ 4:51	+ 3:25	-4 4:47	+4 2:39	+ 0:14	4	750	50%	0.23

Ссылка на контест: http://opentrains.snarknews.info/~ejudge/team.cgi?contest_id=10302

Задача J - Zig Zag Nametag

На слётах ниндзя уровень секретности настолько высок, что на бейджах указываются не реальные имена, а псевдонимы. Один ниндзя захотел прозивести впечатление на своего сенсэя. Он знает, какое число сенсэй считает "счастливым", так что ниндзя хочет писать на бейдже имя, которое кодирует это "счастливое число" следующим образом.

Пусть надпись на бейдже состоит только из строчных латинских букв. Присвоим каждой букве значение, равное её номеру в алфавите при нумерации с единицы (то есть $a = 1$, $b = 2$, $\ldots$, $z = 26$). Числовое значение имени равно сумме модулей разностей значений пар соседних букв; например, для строки "**azxb**" её числовое значение равно

$$|a - z| + |z - x| + |x - b| = |1 - 26| + |26 - 24| + |24 - 2| = 49$$

Требуется найти кратчайшую строку, значение которой равно заданному "счастливому числу" k. Если строк минимальной длины более одной, то ниндзя хочет выбрать ту, которая идёт раньше по алфавиту.

Input

Вход состоит из одной строки, задающей одно целое число k ($1 \le k \le 10^6$) — "счастливое число" сенсэя. Гарантируется, что всегда существует как минимум одна строка, значение которой равно заданному числу.

Output

Выведите кратчайшую строку, значение которой равно заданному числу; в случае, если решений несколько, выведите лексикографически наименьшее.

Examples

standard input	standard output
1	ab
19	at
77	aoazb

Алгоритм

Составляем строку из чередующихся букв $'a'$ и $'z'$. Меняем некоторые буквы на $'b'$ или $'y'$ если получившаяся строка не удовлетворяет условиям. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <algorithm>

int main(){
    long k;
    cin >> k;
    long length = (k - 1) / 25 + 2;

    string s = "";
    for (long i = 0; i < length; i++) {
        if (i  % 2)
            s += "z";
        else
            s += "a";
    }

    if (length > 2) {
        s[1] = (char)('n' + (k - 25 * (length - 2)) / 2);
        long last = s.size() - 1;
        if (s[last] == 'z' && k % 2 == 0) {
            s[last] = 'y';
        }
        else if (s[last] == 'a' && k % 2 == 1) {
            s[last] = 'b';
        }
    }
    else {
        s[0] = 'a';
        s[1] = 'a' + k;
    }
    cout << s << endl;
    return 0;
}
```

Задача K - Knight Jumps

Шахматный конь ходит на два поля в одном направлении, затем на одно поле в перпендикулярном. При этом для того, чтобы ход был возможен, достаточно того, чтобы то поле, на которое делается ход, было свободно.

Вам дана доска $m \times n$, поле, на котором находится конь и поле, на которое он должен дойти. Некоторые поля этой доски заняты, и коня туда ставить нельзя. Выясните, за какое минимальное количество ходов конь сможет добраться до цели (если он вообще туда сможет добраться).

Input

В первой строке входа содержатся два целых числа n и m ($2 \leq n, m \leq 100$), задающие высоту и ширину доски. Каждая из последующих n строк содержит ровно m символов, задающих доску; допустимыми символами являются '`.`' — обозначение для свободного поля, '`#`' — для занятого поля, '`K`' для поля, в котором расположен конь и '`X`' для поля, куда конь должен придти. Гарантируется, что в каждом входном файле будет ровно один символ '`K`' и ровно один символ '`X`'.

Output

Выведите одно число — наименьшее количество ходов, за которое конь сможет добраться до цели, или -1, если не существует способа до неё дойти.

Examples

standard input	standard output
5 4 K... .... .#.. .#.. ...X	5
3 3 K.. .X. ...	-1

Алгоритм

Построим граф всех возможных ходов коня. С помощью поиска в ширину найдём минимальное количество шагов до цели. Сложность $O(n+m)$.

Исходный код

```
#include <iostream>
#include <algorithm>
#include <vector>

int main(){
    int n, m;
    cin >> n >> m;

    int x_num = 0;
    int k_num = 0;

    char symb;

    vector<bool> used(n * m, false);
    vector<vector<int> > g(n * m, vector<int>());

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> symb;
            //used[i][j] = false;
            if (symb == '#') {
                used[i * m + j] = true;
            }
            else if (symb == 'K') {
                k_num = i * m + j;
                used[i * m + j] = true;
            }
            else if (symb == 'X') {
                x_num = i * m + j;
            }
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            int to = i * m + j;

            if (i - 2 >= 0 && j - 1 >= 0)
                g[to].push_back((i - 2) * m + (j - 1));
            if (i - 2 >= 0 && j + 1 < m)
                g[to].push_back((i - 2) * m + (j + 1));

            if (i - 1 >= 0 && j - 2 >= 0)
                g[to].push_back((i - 1) * m + (j - 2));
            if (i + 1 < n && j - 2 >= 0)
                g[to].push_back((i + 1) * m + (j - 2));

            if (i + 2 < n && j - 1 >= 0)
                g[to].push_back((i + 2) * m + (j - 1));
            if (i + 2 < n && j + 1 < m)
                g[to].push_back((i + 2) * m + (j + 1));

            if (i - 1 >= 0 && j + 2 < m)
```

```
                g[to].push_back((i - 1) * m + (j + 2));
            if (i + 1 < n && j + 2 < m)
                g[to].push_back((i + 1) * m + (j + 2));
        }
    }

    queue<int> ways;
    used[k_num] = true;
    ways.push(k_num);

    vector<int> d(n * m), p(n * m);
    p[k_num] = -1;

    while (!ways.empty()) {
        int v = ways.front();
        ways.pop();
        for (size_t i = 0; i < g[v].size(); ++i) {
            int to = g[v][i];
            if (!used[to]) {
                used[to] = true;
                ways.push(to);
                d[to] = d[v] + 1;
                p[to] = v;
            }
        }
    }

    if (!used[x_num])
        cout << "-1" << endl;
    else {
        vector<int> path;
        for (int v = x_num; v!= -1; v = p[v])
            path.push_back (v);
        cout << path.size() - 1 << endl;
    }
    return 0;
}
```

Задача M - Multiple Tom N

Среди обозревателей, следящих за многочисленными студенческими лигами в США, популярно составление различных рейтингов типа Top 25 (или, в общем случае, Top n) команд в стране. Учитывая, что команды из разных лиг между собой могут и не встречаться, данные списки субъективны и часто различаются, однако в целом они более или менее похожи. Ваша задача — сравнить два таких списка одинаковой длины, составленные из одних и тех же команд, и выяснить, в чём они похожи.

Более формально, Вы должны разбить списки на непустые множества, состоящие из одинакового числа позиций в обеих списках и обладающие следующим свойством: набор команд, занимающих эти позиции в обоих списках, одинаков. При этом каждое множество должно содержать как можно меньше команд. Например, для совпадающих списков длины n Вы должны получить n множеств, каждое из которых содержит одну позицию (так как в совпадающих списках все позиции совпадают).

Input

Первая строка входа содержит целое число n ($1 \le n \le 10^6$) — количество команд в списках. Следующие n строк содержат первый список, перечисленный сверху вниз по одной команде в строке. Последующие n строк содержат в аналогичном формате второй список. Названия команд непусты, состоят из не более, чем из 8 заглавных латинских букв и не содержат других символов. Первый и второй списки состоят из одних тех же команд; внутри списков имена команд не повторяются.

Output

Выведите размер каждого множества в порядке следования сверху вниз, по одному в строке.

Examples

standard input	standard output
5 A B C D E A C D B E	1 3 1

Алгоритм

Считываем буквы обоих команд в два вектора в виде номера буквы. Далее циклом проходим одновременно по двум векторам и сравниваем буквы. Если буквы равны, то выводим 1, а если буквы различны, то для каждого вектора начинаем считать сумму букв тем самым добавляя их в два множества (для каждого вектора своё) и подсчитываем количество букв в этих множествах. Делаем это до тех пор пока сумма обоих множеств не станет равна и соответственно оба множества будут равны. Когда они станут равны, выводим количество элементов во множествах и обнуляем их. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <algorithm>
#include <vector>

#define LL  long long
#define ULL unsigned long long

int main() {
  ULL n;
  cin >> n;

  vector <ULL> v1(n);
  vector <ULL> v2(n);
    cin.get();
  for (ULL i = 0; i < n; i++) {
        ULL a = 0;
        char c;
        while((c=cin.get())!='\n') {
            a = a*26+(c-'A');
        }
    v1[i] = a;
  }

  for (ULL i = 0; i < n; i++) {
        ULL a = 0;
        char c;
        while((c=cin.get())!='\n') {
            a = a*26+(c-'A');
        }
        v2[i] = a;
  }

  ULL count = 0;
  ULL sum1 = 0;
  ULL sum2 = 0;

  for(ULL i = 0; i < n; i++) {
    if(v1[i] == v2[i] && count == 0) {
      cout << "1" << endl;
      continue;
    }
    else {
      sum1 += v1[i];
      sum2 += v2[i];
      count++;
    }

    if(sum1 == sum2) {
      cout << count << endl;
      count = 0;
      sum1 = 0;
      sum2 = 0;
    }
  }

    return 0;
}
```

Задача N - New Contest Director

Только что закончилось голосование на выборах нового директора регионального контеста. Поданные голоса уже переписаны в отдельный файл и избирательная комиссия поручила Вам посчитать голоса и сообщить имя того, кто набрал наибольшее количество голосов. Если два и более кандидата набрали одинаковое число голосов, Вы должны вывести имена всех этих кандидатов в алфавитном порядке.

Input

В первой строке входа задано одно целое число n ($1 \leq n \leq 1000$) — общее количество голосов. Каждая из последующих n строк содержит имя кандидата, за которого подан голос; имя состоит только из заглавных латинских букв, непусто и имеет длину не более 20 символов.

Output

Выведите имя кандидата, набравшего наибольшее количество голосов. В случае ничьей выведите имена всех кандидатов, разделивших первое место, в алфавитном порядке по одному имени на строку.

Examples

standard input	standard output
5 BILL MIKE BILL BILL MIKE	BILL
5 BILL MARSHA BILL MARSHA ANONYMOUS	BILL MARSHA

Алгоритм

Создадим ассоциативный массив в котором ключ - имя кандидата, а значение - количество голосов. При считывании очередного имени, инкрементируем ему количество голосов и проверяем не стало ли его количество голосов максимальным, если стало, то записываем количество голосов в переменную max. Затем находим всех кандидатов у которых количество голосов равно максимальному и выводим их. Сложность $O(n)$.

Исходный код

```
#!/usr/bin/perl
<>;
my %h;
my $max = 0;
while($a=<>) {
  ++$h{$a};
  if ($max < $h{$a}) {
    $max = $h{$a};
  }
}
my @m = %h;
my @r;
while(($a = shift @m, $b = shift @m)) {
  if ($b == $max) {
    $a =~ s/\n//;
    push @r, $a;
  }
}
print join "\n", sort @r;
```

2.29 Vekua Cup 2015 Командный этап

Так как соревнование проводилось в центре 1С, исходные коды программ не доступны.

Результаты

88.	[9246-1] MAI #11 (Макаров, Рик, Якименко)	+2 3:17	-	+3 3:59	-	+ 2:58	-	-	-	-	-	-7 5:05	3	714	62%		0.22

Ссылка на контест: http://vekua.snarknews.info

2.30 OpenCup GrandPrix of Ural

Результаты

22.	MAI #11: Makarov, [Rik,] Yakimenko	-	-10 4:54	+1 1:48	-1 1:04	-	+1 0:38	-	-	-	-	-	2	187	50%	0.25

Ссылка на контест: http://opencup.ru

Задача C - Древние ПСП

Во время раскопок в одной из пещер были обнаружены старинные надписи. Каждая из найденных записей представляла собой некоторое число N и правильную скобочную последовательность (ПСП).

Напомним, что ПСП может быть определена так:

- пустая строка является ПСП;
- если S является ПСП, то (S) также является ПСП;
- если S_1 и S_2 являются ПСП, то S_1S_2 является ПСП.

По некоторым из записей археологи поняли, что первое число обозначает количество подстрок в ПСП, которые также являются ПСП. Это навело археологов на мысль, что если они восстановят оставшиеся надписи, то им будет доступно какое-то тайное знание. Поскольку на стенах пещеры в основном обнаружены числа, то они считают, что восстановление отсутствующих ПСП возможно. Также ученые узнали, что длина каждой из ПСП не должна превышать 10^5 символов, поскольку может не поместиться на стену в пещере.

Соответственно, теперь им необходимо построить такую ПСП, которая содержит в себе ровно N подстрок, которые также являются ПСП.

Input

В единственной строке расположено целое положительное число N — количество подстрок в ПСП, которые также являются ПСП ($1 \le N \le 10^9$).

Output

ПСП, которая содержит в себе ровно N подстрок, являющихся ПСП. Длина выводимой строки не должна превышать 10^5 символов.

Examples

стандартный ввод	стандартный вывод
2	(())

Алгоритм

Если дописывать сбоку по одной правильной скобочной последовательности (ПСП), то можно заметить что количество подстрок увеличивается по сумме арифметической последовательности $1, 1+2, 1+2+3, .., 1+2+..+n$, а если записывать внутрь другой скобочной последовательности, то при добавлении каждой ПСП, количество подстрок увеличивается на единицу. Программа основана на рекурсивной функции, которая на каждом шаге считает квадратное уравнение для того чтобы по количеству подстрок узнать необходимое количество ПСП записанных в ряд. Остаток от входного числа передаётся в рекурсивную функцию, результат которой будет записан вложенно в первую ПСП. Сложность $O(log(n))$.

Исходный код

```
#!/usr/bin/perl
$\ = "\n";
sub func {
  my $s = shift;
  my $a = int ((sqrt(1+8*$s)-1)/2);
  my $n = $s - ($a+1)*$a/2;
  my $r = "";
  $r = func($n) if $n != 0;
  "(". $r .")". "()" x ($a-1) ;
}
my $a = <>;
print func $a;
```

Задача F - Фокус

В новом сезоне цирка «Печальный клоун» публику решили развлечь математическим фокусом. Самым опытным работником цирка является фокусник Георгий. Ему и доверили честь показывать этот фокус.

Суть фокуса состоит в том, что Георгий выбирает из зала случайного зрителя и просит его назвать случайное число N. Далее фокусник думает в течение ровно 5,674 секунды и говорит некоторое число M отличное от N. Особенностью этих двух чисел является то, что они имеют одинаковую сумму цифр.

Как только Георгию это удается?

Input

Единственная строка содержит целое неотрицательное число N — число, названное зрителем ($0 \le N \le 10^9$). Гарантируется, что в записи числа N нет лидирующих нулей.

Output

В единственной строке необходимо вывести целое неотрицательное число M — число, названное фокусником. Если существует несколько подходящих чисел, то выведите любое из них.

Число M не должно содержать лидирующих нулей и при этом не должно превышать 10^9.

Если фокусник не может назвать число, то необходимо вывести -1.

Examples

стандартный ввод	стандартный вывод
24	42

Алгоритм

Если число равно нулю, то сразу выводим -1. Если нет, то считаем сумму цифр числа и создаём новое число с такой же суммой. Для этого последовательно вычитаем из числа девятки. Новое число будет состоять из девяток и одной цифры отличной от нуля. Если же это число получилось равно входному, то вычитаем из первой цифры 1 и дописываем 1 спереди. Если число больше 10^9, выводим -1, а если нет, то выводим получившееся число. Сложность $O(log(n))$

Исходный код

```
#!/usr/bin/perl
$a = <>;
if ($a == 0) {
  print "-1\n";
  exit;
}
@m = split "", $a;
map{$s+=$_}@m;
while ($s >= 9) {
  $r .= 9;
  $s -= 9;
}
$r .= $s if $s != 0;
if ($r == $a) {
  $r =~ s/^(\d)(\d+)(\d)$/$3$2$1/;
}
if ($r == $a) {
  $r =~ s/^(\d)(\d*)$/"1".($1-1).$2/e;
}
if ($r > 10**9) {
  print "-1\n";
  exit;
}
print $r;
```

3 Журнал по личным контестам Макарова Н.А.

3.1 Codeforces Round 267 Div 2

Результаты

Задачи

№	Название			
A	Юра и заселение	стандартный ввод/вывод 1 с, 256 МБ		x7565
B	Федя и новая игра	стандартный ввод/вывод 1 с, 256 МБ		x4885
C	Юра и работа	стандартный ввод/вывод 1 с, 256 МБ		x2649
D	Федя и реферат	стандартный ввод/вывод 2 с, 256 МБ		x634
E	Леша и сложная задача	стандартный ввод/вывод 2 с, 256 МБ		x264

Ссылка на контест: http://codeforces.com/contest/467

Задача A - Юра и заселение

Недавно Юра поступил в БГУКП (Берляндский Государственный Университет Крутых Программистов). У Юры есть друг Леша, который поступил вместе с ним, и теперь они заселяются в общежитие.

Юра и Леша хотят жить в одной комнате. Всего в общежитии есть n комнат. В данный момент в комнате с номером i живут p_i человек, когда всего в этой комнате может жить q_i человек ($p_i \le q_i$). Посчитайте, сколько комнат общежития смогут вместить Юру и Лешу вместе?

Входные данные

В первой строке содержится единственное целое число n ($1 \le n \le 100$) — количество комнат.

В i-й из n последующих строк содержатся два целых числа p_i и q_i ($0 \le p_i \le q_i \le 100$) — количество людей, которые уже живут в комнате, и максимальное допустимое количество людей, живущих в i-й комнате.

Выходные данные

Выведите одно целое число — количество комнат, в которые Юра с Лешей могут заселиться.

Примеры тестов

входные данные
3 1 1 2 2 3 3
выходные данные
0

Алгоритм

Задача на простую реализацию. Достаточно пройти по всем p_i и q_i, проверить выполнение условия $p_i + 2 \le q_i$ и если оно выполняется, то инкрементировать счетчик. Решается за $O(n)$.

Исходный код

```
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    int n = 0;
    cin >> n;
    int p = 0, q = 0;
    int res = 0;
    for (int i = 0; i < n; i++) {
        cin >> p >> q;
        if (p + 2 <= q)
            res++;
    }
    cout << res << endl;
    return 0;
}
```

Задача B - Федя и новая игра

Как только вы помогли Юре с Лешей заселиться, они пошли помогать своему другу Феде играть в новую компьютерную игру «Call of Soldiers 3».

Всего в игре есть $(m+1)$ игроков и n видов солдат. Игроки «Call of Soldiers 3» пронумерованы от 1 до $(m+1)$, а виды солдат пронумерованы от 0 до n - 1. У каждого игрока есть армия, армия i-го игрока характеризуется целым неотрицательным числом x_i. Рассмотрим битовое представление числа x_i: если j-й бит числа x_i равен единице, то в армии i-го игрока есть солдаты j-го вида.

Федя — игрок с номером $m+1$. Федя считает, что два игрока могут дружить, если их армии отличаются не более чем на k видов солдат (другими словами, битовые представления соответствующих чисел различаются не более чем в k битах). Помогите Феде посчитать, сколько игроков могут с ним дружить.

Входные данные

В первой строке записаны три целых числа n, m, k $(1 \le k \le n \le 20;\ 1 \le m \le 1000)$.

В i-й из $(m+1)$ последующих строк содержится одно целое число x_i $(1 \le x_i \le 2^n - 1)$, которое характеризует армию i-го игрока. Напомним, что Федя — это игрок с номером $(m+1)$.

Выходные данные

Выведите единственное целое число — количество возможных друзей Феди.

Примеры тестов

входные данные
7 3 1 8 5 111 17
выходные данные
0

Алгоритм

В этой задаче нужно побитово сравнивать число Феди и числа других игроков и инкрементировать счетчик, если количество различных бит меньше k. Задача решается за $O(nb)$, где b – максимальная разрядность числа.

Исходный код

```
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

int main() {
    long n, m, k;
    cin >> n >> m >> k;
    vector<long> gamers(m + 1);

    for (int i = 0; i < m + 1; i++)
        cin >> gamers[i];

    long fedya = gamers[m];
    long friends = 0;
    long diffBits = 0;

    for (int i = 0; i < m; i++) {
        diffBits = 0;
        for (int j = 0; j < 20; j++) {
            if (((fedya >> j) & 1) != ((gamers[i] >> j) & 1))
                diffBits++;
            if (diffBits > k)
                break;
        }
        if (diffBits <= k)
            friends++;
    }

    cout << friends << endl;

    return 0;
}
```

3.2 Codeforces Round 268 Div 2

Результаты

Задачи

№	Название			
A	I Wanna Be the Guy	стандартный ввод/вывод 1 с, 256 МБ		x6005
B	Онлайн чат	стандартный ввод/вывод 1 с, 256 МБ		x3078
C	24 Game	стандартный ввод/вывод 1 с, 256 МБ		x1632
D	Два множества	стандартный ввод/вывод 1 с, 256 МБ		x327
E	Взламываем!	стандартный ввод/вывод 1 с, 256 МБ		x55

Ссылка на контест: http://codeforces.com/contest/469

Задача A - I Wanna Be the Guy

Есть такая игра под названием «I Wanna Be the Guy», в ней n уровней. Little X и его друг Little Y подсели на эту игру. Каждый из них хочет пройти игру полностью.

Little X может пройти только p уровней этой игры. А Little Y может пройти только q уровней этой игры. Вам даны номера уровней, которые может пройти Little X, и номера уровней, которые может пройти Little Y. Могут ли Little X и Little Y пройти игру полностью, если объединят свои усилия?

Входные данные

В первой строке записано единственное целое число n ($1 \le n \le 100$).

В следующей строке сначала записано целое число p ($0 \le p \le n$), затем следуют p различных целых чисел $a_1, a_2, ..., a_p$ ($1 \le a_i \le n$). Эти числа обозначают номера уровней, которые может пройти Little X. В следующей строке содержатся номера уровней, которые может пройти Little Y, в аналогичном формате. Предполагается, что уровни пронумерованы от 1 до n.

Выходные данные

Если друзья могут пройти все уровни вместе, выведите «`I become the guy.`». Если это невозможно, выведите «`Oh, my keyboard!`» (без кавычек).

Примеры тестов

входные данные
4 3 1 2 3 2 2 4
выходные данные
I become the guy.

Алгоритм

В этой задаче нужно проверить, дает ли объединение чисел $a_1, a_2, ..., a_p$ и $a_1, a_2, ..., a_q$ множество чисел $1, 2, ..., n$. Решается за $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, p, q, num;
    cin >> n;
    long sum = ((1 + n) * n) / 2;
    vector<bool> v(n + 1, false);
    cin >> p;
    for (int i = 0; i < p; i++) {
        cin >> num;
        sum -= num;
        v[num] = true;
    }
    cin >> q;
    for (int i = 0; i < q; i++) {
        cin >> num;
        if (!v[num]) sum -= num;
        v[num] = true;
    }
    if (sum)
        cout << "Oh, my keyboard!" << endl;
    else
        cout << "I become the guy." << endl;
    return 0;
}
```

Задача B - Онлайн чат

Little X и Little Z — хорошие друзья. Они постоянно общаются в онлайн-чате. К сожалению, у каждого из них свое расписание.

У Little Z фиксированное расписание. Он онлайн в любой момент времени от a_1 до b_1, от a_2 до b_2, ..., от a_p до b_p (границы включаются в интервалы). У Little X довольно странное расписание, оно зависит того, во сколько он проснется. Если он проснется в момент времени 0, то он будет онлайн в любой момент времени от c_1 до d_1, от c_2 до d_2, ..., от c_q до d_q (границы включаются). Но если он встает в момент t, эти отрезки сдвигаются на t. Другими словами, они будут иметь следующий вид: $[c_i + t, d_i + t]$ (для всех i).

Если в какой-то момент времени и Little X, и Little Z онлайн одновременно, они могут поболтать в чате. Известно, что Little X может встать в любой целочисленный момент времени от l до r (обе границы включительно). Также известно, что Little X хочет встать в такое время, чтобы у него была возможность побеседовать с Little Z (должен быть хотя бы один момент времени, в который они оба онлайн). Сколько целочисленных моментов времени из отрезка $[l, r]$ для этого подходят?

Входные данные

В первой строке записано четыре целых числа через пробел p, q, l, r ($1 \le p, q \le 50$; $0 \le l \le r \le 1000$).

В каждой из следующих p строк записано два целых числа через пробел a_i, b_i ($0 \le a_i < b_i \le 1000$). В каждой из следующих q строк записано по два целых числа через пробел c_j, d_j ($0 \le c_j < d_j \le 1000$).

Гарантируется, что $b_i < a_{i+1}$ и $d_j < c_{j+1}$ для всех i и j, для которых неравенства имеют смысл.

Выходные данные

Выведите единственное целое число — количество подходящих целочисленных моментов времени отрезка $[l, r]$.

Примеры тестов

входные данные
1 1 0 4 2 3 0 1
выходные данные
3

Алгоритм

Создадим вектор всех моментов времени и пометим все моменты, когда не спит Little Z. Далее будем смотреть время Little X и инкрементировать ответ, если у Little Z помечено время, в которое не спит Little X. Решается за $O(rq)$.

Исходный код

```
#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>

using namespace std;

int main() {

    int p, q, l, r;
    cin >> p >> q >> l >> r;

    vector< pair<int, int> > timesX(q);
    vector<bool> timesZ(1002, false);
    pair<int, int> z;

    for (int i = 0; i < p; i++) {
        cin >> z.first >> z.second;
        for (int j = z.first; j <= z.second; j++)
            timesZ[j] = true;
    }
```

```

    for (int i = 0; i < q; i++)
        cin >> timesX[i].first >> timesX[i].second;

    bool marker = false;
    int answer = 0;

    for (int i = l; i <= r; i++) {
        marker = false;
        for (int j = 0; j < q; j++) {
            for (int k = timesX[j].first + i; k <= timesX[j].second + i; k++) {
                if (k > 1000) {
                    marker = true;
                    break;
                }
                if (timesZ[k] == true) {
                    marker = true;
                    answer++;
                }
                if (marker) break;
            }
            if (marker) break;
        }
    }

    cout << answer << endl;

    return 0;
}
```

3.3 Codeforces Отборочный контест СГАУ на четвертьфинал ACM-ICPC

Результаты

Задачи

№	Название			
A	Yet another пусти козла в огород	стандартный ввод/вывод 2 с, 256 МБ		x223
B	Невозможно угадать	стандартный ввод/вывод 2 с, 256 МБ		x167
C	Древний храм	стандартный ввод/вывод 2 с, 256 МБ		x190
D	Игрушечные солдатики	стандартный ввод/вывод 2 с, 256 МБ		x419
E	Просто поменяй слово	стандартный ввод/вывод 2 с, 256 МБ		x344
F	Два конверта	стандартный ввод/вывод 2 с, 256 МБ		x469
G	Задача о размене монет	стандартный ввод/вывод 2 с, 256 МБ		x416
H	Tony Hawk's Pro Skater	стандартный ввод/вывод 2 с, 256 МБ		x93
I	Раскраска карты	стандартный ввод/вывод 2 с, 256 МБ		x149
J	Гипердромы наносят ответный удар	стандартный ввод/вывод 2 с, 256 МБ		x34
K	Два пирата	стандартный ввод/вывод 2 с, 256 МБ		x126
L	Две головы - лучше!	стандартный ввод/вывод 2 с, 256 МБ		x105
M	Построение перестановки	стандартный ввод/вывод 2 с, 256 МБ		x230

Ссылка на контест: http://codeforces.com/gym/100488

Задача D - Игрушечные солдатики

Petya loves toy soldiers very much. He has n soldiers, and i-th soldier is painted the a_i-th color.

Petya constantly doesn't like how his soldiers are colored so he takes one of them and repaints it. He does that m times, and Vova, watching on it, becomes interested when all the soldiers have the same color for the first time. Help Vova to answer his question.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of soldiers Petya has.

The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$) separated by spaces — the initial colors the soldiers were painted.

The third line contains a single integer m ($1 \le m \le 3 \cdot 10^5$) — the number of times Petya has repainted his soldiers.

The next m lines contains two integers each, separated by a space: k_j and x_j ($1 \le k_j \le n$, $1 \le x_j \le 10^9$) — the number of soldier and the number of color it has been repainted (possibly the same as before the repaintment).

Output

Output a single integer — the number of repaintments Petya has made before all his soldiers have the same color for the first time. In particular, if all soldiers had the same color before all Petya's actions, output «0». If such an event has never taken place at all, even after m-th repaintment, output «-1».

Examples

stdin	stdout
3 4 3 7 4 1 5 2 7 1 7 3 3	3
3 6 2 7 3 1 1 2 7 1 6	-1

Алгоритм

Запомним цвета всех солдатиков. Затем будем идти по всем солдатикам, перекрашивать его в новый цвет и проверять, совпадают ли цвета всех солдатиков после перекраски. Если найдено состояние, когда все цвета совпадают – нужно завершить проход и вывести ответ. Работает за $O(nm)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
#define ll unsigned long long

using namespace std;

int main() {
ll n, m;
    ll remColor = 0;
    ll currentSoldier, currentColor, answer = 0;
    bool sameColors = true;
    cin >> n;
    vector<ll> soldiers(n + 1);
    for (ll i = 1; i <= n; i++) {
        cin >> soldiers[i];
        if (i == 1) remColor = soldiers[i];
        else if (soldiers[i] != remColor) sameColors = false;
    }
    if (sameColors) {
        cout << "0" << endl;
        return 0;
    }
    cin >> m;
    sameColors = true;
    bool flag = true;

    for (ll i = 0; i < m; i++) {
        sameColors = false;
        flag = true;
        cin >> currentSoldier >> currentColor;
        soldiers[currentSoldier] = currentColor;
        if (i == 0) remColor = currentColor;
        if (currentColor == remColor) {
            for (ll j = 1; j <= n; j++) {
                if (soldiers[j] != remColor) {
                    flag = false;
                    break;
                }
            }
            if (flag) sameColors = true;
        }
        if (sameColors) {
            answer = i + 1;
            break;
        }
        remColor = currentColor;
    }

    if (sameColors) {
        cout << answer << endl;
    }
    else {
        cout << "-1" << endl;
    }

  return 0;
}
```

Задача F - Два конверта

Mike has two envelopes. He thought up a random integer in the segment $[a, b]$ (he could think up every number from the segment with equal probability) and put exactly this amount of roubles into one of the envelopes and the double amount into the second one. Then he suggested Constantine to play the game: Constantine can open one envelope and then either take its contents or take the contents of another envelope, by his choice.

Constantine has opened one of these envelopes and has discovered c roubles there. Should he take another envelope, if he wants to maximize the expected prize?

Input

The only line contains three integers a, b and c ($1 \le a, b, c \le 10^9$), separated by spaces — the bounds of the segment from which Mike thought up his random number and the amount of roubles in the envelope opened by Constantine. It is guaranteed that the situation described by the input is possible.

Output

Output «`Take another envelope`», if it's advantageous for Constantine to take another envelope, and «`Stay with this envelope`», if it's advantageous for him to stay with the one he has already opened.

Examples

stdin	stdout
3 5 3	Take another envelope
4 6 10	Stay with this envelope

Алгоритм

Задача на теорию вероятности. Если в открытом конверте денег больше чем b, то не нужно менять выбор, иначе нужно поменять, потому что вероятность того, что в другом конверте в 2 раза меньше денег такая же, как и вероятность того, что в другом конверте в 2 раза больше денег. Сложность $O(1)$.

Исходный код

```
#include <iostream>

using namespace std;

int main() {
    long long b, c;
    cin >> b >> b >> c;
    if (c > b) cout << "Stay with this envelope" << endl;
    else cout << "Take another envelope" << endl;
    return 0;
}
```

Задача G - Задача о размене монет

There are $(n+1)$ types of coins in the country R, the cheapest of which has denomination 1 and each of the next types has denomination a_i times greater than the previous one. You need to pay the sum s using as few coins as possible. Of course you can use multiple coins of the same denomination.

Input

The first line contains two integers separated by a space: n and s ($1 \le n \le 10^5$, $0 \le s \le 10^9$) — the number of coins' types, excluding the cheapest one, and the sum to pay.

The second line contains n integers separated by spaces: a_i ($2 \le a_i \le 10^9$) — the number of times each of the next coins is more expensive than the previous one.

Output

Output a single integer — the minimum number of coins required to pay the sum s.

Examples

stdin	stdout
3 42 3 2 2	4
5 228 5 2 5 2 5	8

Алгоритм

Для начала нужно найти, где находится самая дорогая монета, которая не дороже необходимой суммы s. После этого будем набирать сумму последовательно с наибольшей монеты. Работает за $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    unsigned long long n, s, startIndex, coinsNeeded = 0;
    cin >> n >> s;
    vector<unsigned long long> k(n + 1);
    k[0] = 1;
    startIndex = n;
    for (unsigned long long i = 1; i <= n; i++) {
        cin >> k[i];
        if (k[i] * k[i - 1] > s) {
            startIndex = i - 1;
            break;
        }
        k[i] *= k[i - 1];
    }
    for (unsigned long long i = startIndex; ; i--) {
        coinsNeeded += s / k[i];
        s %= k[i];
        if (s == 0 || i == 0)
            break;
    }
    cout << coinsNeeded << endl;
    return 0;
}
```

3.4 Codeforces Round 270 Div 2

Результаты

Задачи

№	Название			
A	Уроки дизайна задач: учимся у математики	стандартный ввод/вывод 1 с, 256 МБ		x7223
B	Уроки дизайна задач: учимся у жизни	стандартный ввод/вывод 1 с, 256 МБ		x4714
C	Уроки дизайна задач: недетеминированность	стандартный ввод/вывод 2 с, 256 МБ		x3767
D	Уроки дизайна задач: обратные задачи	стандартный ввод/вывод 2 с, 256 МБ		x1717
E	Уроки дизайна задач: учимся у игр	стандартный ввод/вывод 1 с, 256 МБ		x66
F	Уроки дизайна задач: меняем цель задачи	стандартный ввод/вывод 2 с, 256 МБ		x120
G	Уроки дизайна задач: увеличиваем ограничения	стандартный ввод/вывод 7 с, 256 МБ		x94

Ссылка на контест: http://codeforces.com/contest/472

Задача A - Уроки дизайна задач: учимся у математики

Один из способов придумать новую задачу: использовать математику. Например, можно придумать какое-нибудь рандомное математическое утверждение или модифицировать некоторые теоремы, чтобы получить что-то новое. Используя такие методы, можно придумать новую задачу.

Например, есть утверждение под названием «Гипотеза Гольдбаха». Оно гласит: «каждое четное число не менее четырех можно представить в виде суммы двух простых чисел». Давайте модифицируем его следующим образом: «каждое целое число не менее 12 можно представить в виде суммы двух составных чисел». В отличие от гипотезы Гольдбаха, я могу доказать эту гипотезу.

Вам дано целое число n не менее 12, представьте его в виде суммы двух составных чисел.

Входные данные
В единственной строке записано целое число n ($12 \le n \le 1000000$).

Выходные данные
Выведите два таких составных целых числа x и y ($1 < x, y < n$), что $x + y = n$. Если есть несколько правильных ответов, можно вывести любой из них.

Примеры тестов

входные данные
12
выходные данные
4 8

Алгоритм

Нужно найти такие числа a и b, чтобы они не были простыми, а $a + b = n$. Для этого положим $a = 4$, а $b = n - 4$. Далее будем проверять, являются ли числа простыми. Если хоть одно из них простое, то инкрементирум a и декрементируем b. В конце концов таким образом найдутся искомые a и b. Сложность $O(n)$.

Исходный код

```
#include <iostream>
bool prime(long long n) {
    long long sq = sqrt(n);
    for (long long i = 2; i <= sq; i++)
        if (n % i == 0)
            return false;
    return true;
}
using namespace std;
int main() {
    long long n;
    cin >> n;
    long long a = 4;
    long long b = n - a;
    bool ok = false;
    while (!ok) {
        if (!prime(a) && !prime(b))
            ok = true;
        else { a++; b--; }
    }
    cout << a << " " << b << endl;
    return 0;
}
```

Задача B - Уроки дизайна задач: учимся у жизни

Один из способов придумывания задач — наблюдать за процессами в реальной жизни. Можно выбрать какую-то реально существующую ситуацию из жизни, формализовать ее и так получить новую задачу.

Представим жизненную ситуацию: стоит много людей, все ждут лифта. Каждый человек хочет попасть на какой-то конкретный этаж. Мы можем формализовать это следующим образом. Пусть n людей стоит на первом этаже, и i-й человек хочет попасть на f_i-й этаж. К сожалению, в здании только один лифт, способный вместить k человек (иными словами, одновременно лифт могут использовать не более k человек). Изначально лифт расположен на первом этаже. Лифту требуется $|a - b|$ секунд для того, чтобы доехать от a-го до b-го этажа (время, необходимое пассажирам лифта на вход и выход, не учитывается).

Какое минимальное количество секунд необходимо для того, чтобы развезти всех людей по необходимым этажам и затем вернуть лифт на первый этаж?

Входные данные

В первой строке записано два целых числа n и k $(1 \le n, k \le 2000)$ — количество людей и максимальная вместимость лифта.

В следующей строке записано n целых чисел: $f_1, f_2, \ldots, f_n$ $(2 \le f_i \le 2000)$, где f_i обозначает этаж, до которого хочет доехать i-й человек.

Выходные данные

Выведите единственное целое число — минимальное время, необходимое для достижения поставленной цели.

Примеры тестов

входные данные
3 2 2 3 4
выходные данные
8

Алгоритм

Чтобы доставить всех людей на нужные этажи за наименьшее количество времени, отсортируем всех по номеру этажа и будем доставлять их порциями по вместимости лифта. Сложность $O(nlog(n))$.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    long n, k;
    cin >> n >> k;
    vector<long> floor(n);
    for (long i = 0; i < n; i++) {
        cin >> floor[i];
    }
    sort(floor.begin(), floor.end(), greater<long>());

    long answer = 0;

    for (long long i = 0; i < n; i += k) {
        answer += 2 * (floor[i] - 1);
    }

    cout << answer << endl;

    return 0;
}
```

3.5 Codeforces Round 273 Div 2

Результаты

Задачи

№	Название			
A	Начальная ставка	стандартный ввод/вывод 1 с, 256 МБ		x6292
B	Случайные команды	стандартный ввод/вывод 1 с, 256 МБ		x4464
C	Украшение столов	стандартный ввод/вывод 1 с, 256 МБ		x2672
D	Красно-зеленые башни	стандартный ввод/вывод 2 с, 256 МБ		x1171
E	Волнистые числа	стандартный ввод/вывод 1,5 с, 256 МБ		x66

Ссылка на контест: http://codeforces.com/contest/478

Задача A - Начальная ставка

В игру «Щедрость» играют пять человек. Каждый из них вносит некоторое ненулевое количество монет b в качестве начальной ставки. После того, как все игроки сделали ставку в b монет, некоторое число раз повторяется следующая операция: у одного из игроков берется одна монета и отдается какому-то другому игроку.

Ваша задача — написать программу, которая по количеству монет у каждого из игроков в конце игры определит размер начальной ставки b или определит, что такой итог игры не мог быть получен ни при каком положительном количестве монет b в начальной ставке.

Входные данные

Ввод состоит из единственной строки, содержащей пять целых чисел c_1, c_2, c_3, c_4 и c_5 — количество монет в конце игры у первого, второго, третьего, четвертого и пятого игрока соответственно ($0 \le c_1, c_2, c_3, c_4, c_5 \le 100$).

Выходные данные

В единственной строке требуется вывести единственное положительное целое число b — количество монет в начальной ставке каждого из игроков. Если не существует такого значения b, то в единственной строке выходных данных требуется вывести «`-1`» (без кавычек).

Примеры тестов

входные данные
`2 5 4 0 4`
выходные данные
`3`

Алгоритм

В этой задаче достаточно заметить, что суммарное количество монет должно без остатка делиться на 5. Иначе исход игры невозможен. Сложность $O(1)$.

Исходный код

```
#include <iostream>

#define LL  long long
#define ULL unsigned long long
using namespace std;

int main() {
    LL c1, c2, c3, c4, c5;
    cin >> c1 >> c2 >> c3 >> c4 >> c5;

    c1 = c1 + c2 + c3 + c4 + c5;

    if (c1 % 5 == 0 && c1) {
        cout << c1 / 5 << endl;
    }
    else cout << "-1" << endl;

    return 0;
}
```

Задача B - Случайные команды

Для участия в соревнованиях n участников были разбиты некоторым образом на m команд так, чтобы в каждой команде был хотя бы один участник. После соревнований каждая пара участников из одной команды стала друзьями.

Ваша задача — написать программу, которая определит, какое минимальное и какое максимальное количество пар друзей могло образоваться после соревнования.

Входные данные

В единственная строке содержатся два целых числа n и m, разделенных одним пробелом ($1 \le m \le n \le 10^9$) — количество участников и количество команд соответственно.

Выходные данные

Требуется вывести два целых числа k_{min} и k_{max} — минимальное возможное количество пар друзей и максимальное возможное количество пар друзей соответственно.

Примеры тестов

входные данные
5 1
выходные данные
10 10

Алгоритм

Задача на комбинаторику. Минимальное количество получается если всех разделить на команды поровну, а максимальное если в одну из команд поместить максимально возможное количество участников. Сложность $O(nlog(1))$.

Исходный код

```
#include <iostream>
#define LL  long long
using namespace std;
LL fun(LL n) {
    return (n * n - n) / 2;
}
int main() {
    LL n, m;
    cin >> n >> m;
    LL min, max;
    LL k = n - (m - 1);
    max = fun(k);
    LL g = n / m;
    if (n % m == 0) {
        min = m * fun(g);
    }
    else {
        min = (m - (n % m)) * fun(g);
        min += (n % m) * fun(g + 1);
    }
    cout << min << " " << max << endl;
    return 0;
}
```

3.6 Codeforces Round 274 Div 2

Результаты

Задачи

№	Название			
A	Выражение	стандартный ввод/вывод 1 с, 256 МБ		x6231
B	Башни	стандартный ввод/вывод 1 с, 256 МБ		x2942
C	Экзамены	стандартный ввод/вывод 1 с, 256 МБ		x2939
D	Прыжки в длину	стандартный ввод/вывод 1 с, 256 МБ		x1196
E	Катаемся на лифте	стандартный ввод/вывод 2 с, 256 МБ		x685

Ссылка на контест: http://codeforces.com/contest/480

Задача A - Выражение

Петя учится в школе и очень любит математику. Уже несколько занятий они с классом проходят арифметические выражения. На последнем уроке учительница написала на доске три положительных целых числа a, b, c. Задание заключалось в том, чтобы расставить между этими числами знаки операций '+' и '*', а также, возможно, скобки. Значение получившегося выражения должно быть как можно больше. Рассмотрим пример: пусть учительница выписала на доску числа 1, 2 и 3. Вот некоторые варианты расстановки знаков и скобок:

- 1+2*3=7
- 1*(2+3)=5
- 1*2*3=6
- (1+2)*3=9

Обратите внимание на то, что знаки операций можно вставлять только между a и b, а также между b и c, то есть нельзя менять числа местами. Так, в приведенном примере нельзя получить выражение (1+3)*2.

Легко убедиться, что максимальное значение, которое можно получить, — это 9.

Ваша задача — по заданным a, b и c вывести, какое максимальное значение выражения можно получить.

Входные данные
Во входных данных записаны три целых числа a, b и c, каждое в отдельной строке ($1 \le a, b, c \le 10$).

Выходные данные
Выведите максимальное значение выражения, которое можно получить.

Примеры тестов

входные данные
1 2 3
выходные данные
9

Алгоритм

Всего может быть 6 вариантов выражения, поэтому просто переберем все варианты и выберем наибольший ответ. Сложность алгоритма $O(1)$.

Исходный код

```
#include <iostream>
using namespace std;
int main() {
    int a, b, c;
    cin >> a >> b >> c;
    int s;
    int answer = 0;
    s = a + b + c;
    if (s > answer) answer = s;
    s = a * b + c;
    if (s > answer) answer = s;
    s = a + b  * c;
    if (s > answer) answer = s;
    s = a * b * c;
    if (s > answer) answer = s;
    s = (a + b) * c;
    if (s > answer) answer = s;
    s = a * (b + c);
    if (s > answer) answer = s;
    cout << answer << endl;
    return 0;
}
```

Задача B - Башни

Как известно, все дети в Берляндии любят играть с кубиками. У маленького Пети имеется n башен, состоящих из кубиков одинакового размера. Башня под номером i представляет собой a_i кубиков, поставленных друг на друга. *Неустойчивостью* набора башен Петя называет величину, равную разности высот самой высокой и самой низкой башни. К примеру, если Петя построил из кубиков пять башен с высотами (8, 3, 2, 6, 3), то неустойчивость этого набора равна 6 (самая высокая башня имеет высоту 8, самая низкая — высоту 2).

Мальчик хочет, чтобы неустойчивость его набора башен была как можно меньше. Все, что он может сделать, это несколько раз проделать следующую операцию: взять верхний кубик с какой-то из башен и положить его сверху на какую-то другую башню из своего набора. Обратите внимание, что Петя никогда не будет класть кубик на ту же башню, с которой тот был снят, потому что считает это пустой тратой времени.

Прежде чем отправиться в школу, мальчик успеет произвести не более k таких операций. Петя не хочет опоздать на урок, поэтому вам придется помочь ему выполнить эту задачу.

Входные данные

В первой строке через пробел записаны два целых положительных числа n и k ($1 \le n \le 100$, $1 \le k \le 1000$) — количество башен в имеющемся наборе и максимальное число операций, которые Петя может произвести. Во второй строке через пробел записаны n целых положительных чисел a_i ($1 \le a_i \le 10^4$) — исходные высоты башен.

Выходные данные

В первой строке выведите через пробел два целых неотрицательных числа s и m ($m \le k$). Первое из чисел — это величина минимально возможной неустойчивости, которую можно достичь, применив не более k операций, а второе — количество операций, необходимых для этого.

Затем в m строках выведите описание каждой из операций в виде двух целых положительных чисел i и j, каждое из которых лежит в пределах от 1 до n. Они обозначают, что Петя переложил верхний кубик с i-й башни на j-ю ($i \ne j$). Обратите внимание, что в процессе выполнения операций высоты некоторых башен могут стать равны нулю.

Если существует несколько корректных последовательностей операций, при которых достигается минимально возможная неустойчивость, разрешается вывести любую из них.

Примеры тестов

входные данные
3 2 5 8 5
выходные данные
0 2 2 1 2 3

Алгоритм

Отсортируем башни по высоте и будем перекладывать кубик с самой высокой на самую низкую. После этого нужно снова отсортировать новые высоты. Это нужно проделать k раз. Сложность алгоритма $O(knlog(n))$.

Исходный код

```
#include <iostream>
using namespace std;

typedef struct {
    int n, h;
} tow;

bool cmp(const tow &a, const tow &b) {
    return a.h > b.h;
}

int main() {
    int n, k;
    cin >> n >> k;
```

```
    vector<tow> t(n);
    vector<pair<int, int>> actions;
    for (int i = 0; i < n; i++) {
        cin >> t[i].h;
        t[i].n = i + 1;
    }
    pair<int, int> move;
    int min = inf;
    vector<int> states(k + 1);
    sort(t.begin(), t.end(), cmp);
    states[0] = t[0].h - t[n - 1].h;
    for (int i = 1; i <= k; i++) {
        t[0].h--;
        t[n - 1].h++;
        move.first = t[0].n;
        move.second = t[n - 1].n;
        actions.push_back(move);
        sort(t.begin(), t.end(), cmp);
        states[i] = t[0].h - t[n - 1].h;
    }
    int minPos = -1;
    for (int i = 0; i < k + 1; i++) {
        if (states[i] < min) {
            min = states[i];
            minPos = i;
        }
    }
    cout << min << " " << minPos << endl;
    for (int i = 0; i < minPos; i++) cout << actions[i].first << " " << actions[i].
   second << endl;
    return 0;
}
```

Задача C - Экзамены

Студент Валера учится на первом курсе университета. Скоро у него сессия, и ему предстоит сдать ровно n экзаменов. Валера — умный парень, поэтому он сможет сдать любой экзамен с первого раза. Кроме того, он может сдавать несколько экзаменов в один день и в любом порядке.

Согласно расписанию, экзамен по i-му предмету нужно сдать в день с номером a_i. Однако Валера договорился с каждым преподавателем, и преподаватель i-го предмета разрешил организовать досрочную сдачу своего экзамена в день b_i ($b_i < a_i$). Таким образом, Валера может сдать экзамен по i-му предмету либо в день a_i, либо в день b_i. Все преподаватели ставят запись о сдаче экзамена в зачетную книжку в день фактической сдачи экзамена и датируют эту запись числом a_i.

Валера считает, что будет достаточно странно, если записи в зачетной книжке будут идти не в порядке неубывания даты. Поэтому Валера просит вас помочь ему. Найдите минимально возможный номер дня, когда Валера сможет сдать последний экзамен, если он будет сдавать экзамены так, чтобы все записи в его зачетной книжке шли в порядке неубывания даты.

Входные данные

В первой строке записано единственное целое положительное число n ($1 \le n \le 5000$) — количество экзаменов, которые будет сдавать Валера.

В каждой из следующих n строк записано по два целых положительных числа через пробел a_i и b_i ($1 \le b_i < a_i \le 10^9$) — дата сдачи по расписанию и досрочная дата сдачи i-го экзамена соответственно.

Выходные данные

Выведите единственное целое число — минимально возможный номер дня, когда Валера сможет сдать последний экзамен, если он будет сдавать экзамены так, чтобы все записи в его зачетной книжке шли в порядке неубывания даты.

Примеры тестов

входные данные
3 5 2 3 1 4 2
выходные данные
2

Алгоритм

Для решения этой задачи нужно отсортировать экзамены сначала по дате сдачи по расписанию, а затем устойчиво отсортировать по датам досрочной сдачи. Потом нужно пройти по отсортированным датам и если у текущего экзамена досрочная дата раньше чем досрочная предыдущего, то текущий нужно сдавать по расписанию. Сложность алгоритма $O(nlog(n))$.

Исходный код

```
#include <iostream>
using namespace std;

bool cmp_pair_first(const pair<LL, LL> &a, const pair<LL, LL> &b) {
    return a.first < b.first;
}

bool cmp_pair_second(const pair<LL, LL> &a, const pair<LL, LL> &b) {
    return (a.second < b.second) && (a.first == b.first);
}

int main() {
    int n;
    cin >> n;
    vector<pair<LL, LL>> e(n);
    LL maxFirst = -1, maxSecond = -1;
    for (int i = 0; i < n; i++) {
        cin >> e[i].first >> e[i].second;
        if (e[i].first  > maxFirst)  maxFirst  = e[i].first;
        if (e[i].second > maxSecond) maxSecond = e[i].second;
    }
    sort(e.begin(), e.end(), cmp_pair_first);
    stable_sort(e.begin(), e.end(), cmp_pair_second);
    for (int i = 1; i < n; i++) {
        if (e[i].second < e[i - 1].second) {
            e[i].second = e[i].first;
        }
    }
    cout << e[n - 1].second << endl;
    return 0;
}
```

3.7 Codeforces Round 275 Div 2

Результаты

Задачи

№	Название			
A	Опровержение гипотез	стандартный ввод/вывод 1 с, 256 МБ		x4232
B	Друзья и подарки	стандартный ввод/вывод 1 с, 256 МБ		x1626
C	Разнообразная перестановка	стандартный ввод/вывод 1 с, 256 МБ		x2211
D	Интересный массив	стандартный ввод/вывод 1 с, 256 МБ		x402
E	Игра со строками	стандартный ввод/вывод 1 с, 256 МБ		x28

Ссылка на контест: http://codeforces.com/contest/483

Задача C - Разнообразная перестановка

Перестановкой p называется упорядоченный набор чисел $p_1, p_2, \ldots, p_n$, состоящий из n различных целых положительных чисел, каждое из которых не больше чем n. Число n будем называть длиной перестановки $p_1, p_2, \ldots, p_n$.

Ваша задача — найти такую перестановку p длины n, что среди чисел $|p_1 - p_2|, |p_2 - p_3|, \ldots, |p_{n-1} - p_n|$ ровно k различных.

Входные данные

В единственной строке входных данных находятся два разделённых пробелом целых положительных числа n, k ($1 \le k < n \le 10^5$).

Выходные данные

Выведите n целых чисел — искомую перестановку. Если существует несколько ответов, разрешается вывести любой.

Примеры тестов

входные данные
3 2
выходные данные
1 3 2

Алгоритм

Получить искомую перестановку можно записывая поочередно $b = 1$ и $e = n$. После записи b нужно инкрементировать, а e декрементировать. В результате будет получена искомая перестановка. Сложность алгоритма $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    long n, k;
    cin >> n >> k;
    vector<long> v(n, 0);
    long b = 1, e = n;
    for (long i = 0; i < k; i++) {
        if (i % 2 == 0) v[i] = b++;
        else v[i] = e--;
    }
    for (long i = k; i < n; i++) {
        if (k % 2 == 1) v[i] = b++;
        else v[i] = e--;
    }
    for (auto n : v) cout << n << " ";
    cout << endl;
    return 0;
}
```

3.8 VK Cup 2015 Квалификация

Результаты

Задачи

№	Название			
A	Репосты	стандартный ввод/вывод 1 с, 256 МБ		x1741
B	Фото на память	стандартный ввод/вывод 2 с, 256 МБ		x1801
C	Chicken or Fish?	стандартный ввод/вывод 1 с, 256 МБ		x387
D	Ближайшие равные	стандартный ввод/вывод 3 с, 256 МБ		x623

Ссылка на контест: http://codeforces.com/contest/522

Задача A - Репосты

Однажды Поликарп опубликовал в социальной сети смешную картинку с опросом про цвет своего хэндла. Многие из его друзей стали репостить шутку Поликарпа себе в ленту. Некоторые из них репостили репосты и так далее.

Эти события заданы в виде последовательности строк «*name1* `reposted` *name2*» где *name1* — это имя того, кто репостнул, а *name2* — имя того, с чей ленты репостнули шутку. Гарантируется, что для каждой строки «*name1* `reposted` *name2*» пользователь «*name1*» еще не имел эту шутку в свой ленте, а «*name2*» уже имел ее в своей ленте к моменту репоста. Поликарп зарегистрирован под именем «`Polycarp`», и изначально шутка есть только в его ленте.

Поликарп измеряет успешность шутки как длину наибольшей цепочки репостов. Выведите успешность шутки Поликарпа.

Входные данные

В первой строке входных данных записано целое число n ($1 \le n \le 200$) — количество репостов. Далее записаны сами репосты в порядке их совершения. Каждый из них записан в отдельной строке и имеет вид «*name1* `reposted` *name2*». Все имена во входных данных состоят из прописных или строчных латинских букв и/или цифр и имеют длины от 2 до 24 символов включительно.

Известно, что имена пользователей регистронезависимы, то есть два имени, отличающиеся исключительно регистром букв, соответствуют одному и тому же пользователю соцсети.

Выходные данные

Выведите единственное целое число — максимальную длину цепочки репостов.

Примеры тестов

входные данные
5 tourist reposted Polycarp Petr reposted Tourist WJMZBMR reposted Petr sdya reposted wjmzbmr vepifanov reposted sdya
выходные данные
6

Алгоритм

Так как требуется найти наибольшую цепочку, то задачу можно решить поиском в глубину и запомнить наибольшую длину пути. Сложность $O(N + M)$.

Исходный код

```
#include <iostream>

#define LL  long long
#define ULL unsigned long long
using namespace std;

vector<vector<int>> g;
vector<bool> used;
int depth = 0;
int maxDepth = 0;

void dfs(int v) {
    used[v] = true;
    depth++;
    for (vector<int>::iterator i = g[v].begin(); i != g[v].end(); ++i) {
        if (!used[*i]) {
            dfs (*i);
        }
    }
    if (depth > maxDepth) {
        maxDepth = depth;
    }
```

```
    depth--;
}

int main(int argc, const char * argv[]) {
    int n;
    cin >> n;

    map<string, int> names;
    vector<pair<string, string>> log(n);
    string left, right, dummy;
    int nameNumber = 0;

    for (int i = 0; i < n; i++) {
        cin >> left >> dummy >> right;
        transform(left.begin(), left.end(), left.begin(), ::tolower);
        transform(right.begin(), right.end(), right.begin(), ::tolower);
        if (names.find(left) == names.end()) {
            names[left] = nameNumber++;
        }
        if (names.find(right) == names.end()) {
            names[right] = nameNumber++;
        }
        log[i].first = left;
        log[i].second = right;
    }

    g.resize(names.size(), vector<int>());
    used.resize(g.size());
    map<string, int>::iterator from, to;

    for (size_t i = 0; i < log.size(); i++) {
        from = names.find(log[i].second);
        to = names.find(log[i].first);
        g[from->second].push_back(to->second);
    }

    int start = names.find("polycarp")->second;
    dfs(start);
    cout << maxDepth << endl;

    return 0;
}
```

Задача B - Фото на память

На вечеринке встретились n друзей, они давно не собирались все вместе и поэтому решили сделать общее групповое фото.

Упрощённо процесс фотографирования можно описать следующим образом. На фотографии каждый из друзей занимает прямоугольник из пикселей: i-й из них занимает прямоугольник ширины w_i пикселей и высоты h_i пикселей. На групповом фото все фотографируемые стоят в ряд, таким образом минимальный размер в пикселях фотографии, включающей всех друзей, составляет $W \times H$, где W — суммарная ширина всех фотографируемых, а H — максимальная из высот всех фотографируемых.

Как это обычно и бывает, друзья сфотографировались n раз — на j-й ($1 \le j \le n$) фотографии присутствовали все, кроме j-го из них, ведь он был фотографом.

Выведите минимальный размер в пикселях каждого из сделанных фото.

Входные данные
В первой строке записано целое число n ($2 \le n \le 200\,000$) — количество друзей.

Далее следует n строк: i-я из них содержит информацию об i-м из друзей. В строке содержится пара целых чисел w_i, h_i ($1 \le w_i \le 10$, $1 \le h_i \le 1000$) — ширина и высота в пикселях соответствующего ему прямоугольника.

Выходные данные
Выведите n разделённых пробелами чисел $b_1, b_2, \ldots, b_n$, где b_i — общее количество пикселей на минимальной фотографии, вмещающей всех друзей, кроме i-го из них.

Примеры тестов

входные данные
3 1 10 5 5 10 1
выходные данные
75 110 60

Алгоритм

В этой задаче нужно знать высоту самого высокого и второго по высоте. Это можно сделать с помощью k-й порядковой статистики. А потом нужно посчитать количество пикселей. Можно заметить, что высота фотографии всегда будет равна высоте самого высокого из друзей, и один раз высоте второго по высоте.

Исходный код

```
#include <iostream>

#define LL  long long
#define ULL unsigned long long
using namespace std;

int main() {
  int temp = 0;

  LL n;
  cin >> n;

  vector <int> width(n);
  vector <int> height(n);
  vector <int> height_2(n);

  int max_h = 0, premax_h = 0;
  LL sum_w = 0;

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

    cin >> width[i] >> height[i];
    height_2[i] = height[i];

    sum_w += width[i];
  }

  nth_element(height_2.begin(), height_2.end() - 1, height_2.end());
  max_h = height_2[n - 1];
  nth_element(height_2.begin(), height_2.end() - 2, height_2.end());
  premax_h = height_2[n - 2];

  LL temp_sum, temp_max;
  for (int i = 0; i < n; i++) {
    temp_sum = sum_w - width[i];
    if (height[i] == max_h)
      temp_max = premax_h;
    else temp_max = max_h;
    cout << temp_sum * temp_max << " ";
  }
  return 0;
}
```

3.9 VK Cup 2015 - Уайлд-кард раунд 1

Результаты

Задачи

№	Название			
A	Квадратное уравнение	стандартный ввод/вывод 2 с, 256 МБ		x298
B	Строка наизнанку	стандартный ввод/вывод 2 с, 256 МБ		x240
C	Диофантово уравнение	стандартный ввод/вывод 2 с, 256 МБ		x190
D	Разность множеств	стандартный ввод/вывод 2 с, 256 МБ		x195
E	Сумма и произведение	стандартный ввод/вывод 2 с, 256 МБ		x118
F	Прыгающие лягушки	стандартный ввод/вывод 2 с, 256 МБ		x29
G	Расстояние Левенштейна	стандартный ввод/вывод 2 с, 256 МБ		x29
H	Точки в треугольнике	стандартный ввод/вывод 2 с, 256 МБ		x48
I	Разные переменные	стандартный ввод/вывод 2 с, 256 МБ		x13

Ссылка на контест: http://codeforces.com/contest/530

Задача A - Квадратное уравнение

Вам дано квадратное уравнение с целыми коэффициентами $A * X^2 + B * X + C = 0$. Гарантируется, что $A \neq 0$ и уравнение имеет хотя бы один действительный корень. Найдите корни уравнения.

Входные данные

Единственная строка входных данных содержит целые числа A, B и C (- $1000 \leq A, B, C \leq 1000, A \neq 0$).

Выходные данные

Выведите корни уравнения в порядке их возрастания. Если уравнение имеет один корень кратности 2, выведите его только один раз. Корень считается правильным, если его абсолютная или относительная погрешность не превосходит 10^{-4}.

Примеры тестов

входные данные
1 -2 1
выходные данные
1

Алгоритм

Сложность этой задачи состоит в том, что написать ее нужно на эзотерическом языке *Picat*.

Исходный код

```
main =>
 A = read_int(),
 B = read_int(),
 C = read_int(),

 D = (B * B) - (4 * A * C),

 X1 = ((-1)* B + sqrt(D)) / (2 *A ),

 X2 = ((-1)* B - sqrt(D)) / (2 * A),


 if(X1 < X2) then
     println(X1),
     println(X2)
     end,

 if(X1 > X2) then
     println(X2),
     println(X1)
     end,
 if(X1 == X2) then
     println(X2)
     end.
```

3.10 Vekua Cup 2015 Личный этап

Так как соревнование проводилось в центре 1С, исходные коды программ не доступны.

Результаты

131.	[9246-25] Никита Макаров (МАИ)	-	-	-	-	-	-	-	+ 1:25	1	85	0%		0.09

Ссылка на контест: http://vekua.snarknews.info

3.11 Mail.ru Russian Code Cup 2015 Квалификация

Результаты

Задачи	Статус	Попытки	Штрафное время
Покупка велосипеда	✓	1	0:14
Цифровые корни	✗	5	–
Две улитки	–	–	–
Игровые автоматы	–	–	–
Интернетопровод	–	–	–

Ссылка на контест: http://www.russiancodecup.ru/championship/round/38/

Задача A - Покупка велосипеда

Сегодня Петя убирался в комнате и нашел под кроватью очень много мелочи — рублевых и двухрублевых монет. Внимательно все пересчитав, Петя понял, что у него есть a монет номиналом в один рубль и b монет номиналом в два рубля.

Петя очень давно хотел купить себе велосипед, поэтому сразу же пошел в магазин и узнал, что его заветное средство передвижения стоит c рублей. К сожалению, в магазине велосипедов проблемы с мелкими деньгами, а без сдачи Петя уходить не хочет. Поэтому теперь он задался вопросом: можно ли купить велосипед, заплатив при этом ровно c рублей?

Помогите Пете, скажите, можно ли, имея a рублевых монет и b двухрублевых, купить без сдачи велосипед стоимостью c рублей?

Формат входных данных

Первая строка входных данных содержит одно число t ($1 \leqslant t \leqslant 100000$) — количество тестов. Следующие t строк содержат по тесту каждая. Каждый тест задается тремя целыми числами: a, b, c ($0 \leqslant a, b, c \leqslant 10^8$) — количество рублевых, двухрублевых монет и стоимость велосипеда соответственно.

Формат выходных данных

Для каждого набора данных выведите единственную строку: «YES», если можно купить велосипед без сдачи и «NO» в противном случае.

Примеры

Входные данные	Выходные данные
4 1 2 4 2 1 4 1 2 3 3 1 6	YES YES YES NO

Алгоритм

В этой задаче по числам a, b и c нужно определить, существуют ли такие a_1 и b_1, что $a_1 \leq a$, $b_1 \leq b$ и $a_1 + 2b_1 = c$. Для этого достаточно проверить, что $a + 2b \geq c$, а если c – нечетное, то проверить условие $a \geq 1$. Сложность $O(1)$.

Исходный код

```
#include <iostream>
using namespace std;
int main() {
  long t, a, b, c;
  cin >> t;
  while (t--) {
    cin >> a >> b >> c;
    if (c % 2) {
      if (a + 2*b >= c && a >= 1) {
        cout << "YES";
      } else {
        cout << "NO";
      }
    } else {
      if (a + 2*b >= c && a >= 1) {
        cout << "YES";
      } else {
        cout << "NO";
      }
    }
    cout << endl;
  }
  return 0;
}
```

4 Журнал по личным контестам Якименко А.В.

4.1 Codeforces Round 267 Div 2

Результаты

Задачи

№	Название			
A	Юра и заселение	стандартный ввод/вывод 1 с, 256 МБ		x7593
B	Федя и новая игра	стандартный ввод/вывод 1 с, 256 МБ		x4898
C	Юра и работа	стандартный ввод/вывод 1 с, 256 МБ		x2651
D	Федя и реферат	стандартный ввод/вывод 2 с, 256 МБ		x634
E	Леша и сложная задача	стандартный ввод/вывод 2 с, 256 МБ		x264

Ссылка на контест: http://codeforces.com/contest/467

Задача A - Юра и заселение

Недавно Юра поступил в БГУКП (Берляндский Государственный Университет Крутых Программистов). У Юры есть друг Леша, который поступил вместе с ним, и теперь они заселяются в общежитие.

Юра и Леша хотят жить в одной комнате. Всего в общежитии есть n комнат. В данный момент в комнате с номером i живут p_i человек, когда всего в этой комнате может жить q_i человек ($p_i \le q_i$). Посчитайте, сколько комнат общежития смогут вместить Юру и Лешу вместе?

Входные данные

В первой строке содержится единственное целое число n ($1 \le n \le 100$) — количество комнат.

В i-й из n последующих строк содержатся два целых числа p_i и q_i ($0 \le p_i \le q_i \le 100$) — количество людей, которые уже живут в комнате, и максимальное допустимое количество людей, живущих в i-й комнате.

Выходные данные

Выведите одно целое число — количество комнат, в которые Юра с Лешей могут заселиться.

Примеры тестов

входные данные
3 1 1 2 2 3 3
выходные данные
0

Алгоритм

Задача на простую реализацию. Достаточно пройти по всем p_i и q_i, проверить выполнение условия $p_i + 2 \le q_i$ и если оно выполняется, то инкрементировать счетчик. Решается за $O(n)$.

Исходный код

```
#include <iostream>
using namespace std;

int main()
{
    int count = 0;
    int n;
    cin >> n;
    for(int i=0; i<n; ++i)
    {
        int a, b;
        cin >> a >> b;
        if(a+2<=b)
            ++count;
    }
    cout << count;
    return 0;
}
```

Задача B - Федя и новая игра

Как только вы помогли Юре с Лешей заселиться, они пошли помогать своему другу Феде играть в новую компьютерную игру «Call of Soldiers 3».

Всего в игре есть $(m+1)$ игроков и n видов солдат. Игроки «Call of Soldiers 3» пронумерованы от 1 до $(m+1)$, а виды солдат пронумерованы от 0 до n - 1. У каждого игрока есть армия, армия i-го игрока характеризуется целым неотрицательным числом x_i. Рассмотрим битовое представление числа x_i: если j-й бит числа x_i равен единице, то в армии i-го игрока есть солдаты j-го вида.

Федя — игрок с номером $m+1$. Федя считает, что два игрока могут дружить, если их армии отличаются не более чем на k видов солдат (другими словами, битовые представления соответствующих чисел различаются не более чем в k битах). Помогите Феде посчитать, сколько игроков могут с ним дружить.

Входные данные

В первой строке записаны три целых числа n, m, k $(1 \le k \le n \le 20;\ 1 \le m \le 1000)$.

В i-й из $(m+1)$ последующих строк содержится одно целое число x_i $(1 \le x_i \le 2^n - 1)$, которое характеризует армию i-го игрока. Напомним, что Федя — это игрок с номером $(m+1)$.

Выходные данные

Выведите единственное целое число — количество возможных друзей Феди.

Примеры тестов

входные данные
7 3 1 8 5 111 17
выходные данные
0

Алгоритм

В этой задаче нужно побитово сравнивать число Феди и числа других игроков и инкрементировать счетчик, если количество различных бит меньше k. Задача решается за $O(nb)$, где b – максимальная разрядность числа.

Исходный код

```
#include <iostream>
#include <vector>
#include <map>
#include <cmath>
#include <algorithm>
#include <iomanip>
#include <list>
#include <iterator>
#include <climits>
using namespace std;
#define ll long long
#define ull unsigned long long
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}
int main()
{
    ull n, m, k;
    cin >> n >> m >> k;
    vector<ull> v(m);
    for(ll i=0; i<m; ++i)
        cin >> v[i];
    ull x;
    cin >> x;
    ll count = 0;
    for(ll i=0; i<m; ++i)
    {
        ull a = v[i]^x;
        ull bitCount = 0;
        for(ll i=0; i<n; ++i)
            if((a>>i)&1)
                ++ bitCount;
        if(bitCount <= k)
            ++count;
    }
    cout << count;
    return 0;
}
```

4.2 Codeforces Round 268 Div 2

Результаты

Задачи

№	Название			
A	I Wanna Be the Guy	стандартный ввод/вывод 1 с, 256 МБ		x6025
B	Онлайн чат	стандартный ввод/вывод 1 с, 256 МБ		x3083
C	24 Game	стандартный ввод/вывод 1 с, 256 МБ		x1636
D	Два множества	стандартный ввод/вывод 1 с, 256 МБ		x327
E	Взламываем!	стандартный ввод/вывод 1 с, 256 МБ		x55

Ссылка на контест: http://codeforces.com/contest/469

Задача A - I Wanna Be the Guy

Есть такая игра под названием «I Wanna Be the Guy», в ней n уровней. Little X и его друг Little Y подсели на эту игру. Каждый из них хочет пройти игру полностью.

Little X может пройти только p уровней этой игры. А Little Y может пройти только q уровней этой игры. Вам даны номера уровней, которые может пройти Little X, и номера уровней, которые может пройти Little Y. Могут ли Little X и Little Y пройти игру полностью, если объединят свои усилия?

Входные данные

В первой строке записано единственное целое число n ($1 \le n \le 100$).

В следующей строке сначала записано целое число p ($0 \le p \le n$), затем следуют p различных целых чисел $a_1, a_2, ..., a_p$ ($1 \le a_i \le n$). Эти числа обозначают номера уровней, которые может пройти Little X. В следующей строке содержатся номера уровней, которые может пройти Little Y, в аналогичном формате. Предполагается, что уровни пронумерованы от 1 до n.

Выходные данные

Если друзья могут пройти все уровни вместе, выведите «`I become the guy.`». Если это невозможно, выведите «`Oh, my keyboard!`» (без кавычек).

Примеры тестов

входные данные
4 3 1 2 3 2 2 4
выходные данные
I become the guy.

Алгоритм

В этой задаче нужно проверить, дает ли объединение чисел $a_1, a_2, ..., a_p$ и $a_1, a_2, ..., a_q$ множество чисел $1, 2, ..., n$. Решается за $O(n)$.

Исходный код

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, const char * argv[]) {
    int n;
    scanf("%d", &n);
    char mas[100];
    int i;
    for(i=0; i<n; ++i)
        mas[i] = 0;
    int a, b;
    scanf("%d", &a);
    for(i=0; i<a; ++i)
    {
        scanf("%d", &b);
        mas[b-1] = 1;
    }
    scanf("%d", &a);
    for(i=0; i<a; ++i)
    {
        scanf("%d", &b);
        mas[b-1] = 1;
    }
    for(i=0; i<n; ++i)
    {
        if(mas[i] == 0)
        {
            printf("Oh, my keyboard!");
            return 0;
        }
    }
    printf("I become the guy.");
    return 0;
}
```

Задача B - Онлайн чат

Little X и Little Z — хорошие друзья. Они постоянно общаются в онлайн-чате. К сожалению, у каждого из них свое расписание.

У Little Z фиксированное расписание. Он онлайн в любой момент времени от a_1 до b_1, от a_2 до b_2, ..., от a_p до b_p (границы включаются в интервалы). У Little X довольно странное расписание, оно зависит того, во сколько он проснется. Если он проснется в момент времени 0, то он будет онлайн в любой момент времени от c_1 до d_1, от c_2 до d_2, ..., от c_q до d_q (границы включаются). Но если он встает в момент t, эти отрезки сдвигаются на t. Другими словами, они будут иметь следующий вид: $[c_i + t, d_i + t]$ (для всех i).

Если в какой-то момент времени и Little X, и Little Z онлайн одновременно, они могут поболтать в чате. Известно, что Little X может встать в любой целочисленный момент времени от l до r (обе границы включительно). Также известно, что Little X хочет встать в такое время, чтобы у него была возможность побеседовать с Little Z (должен быть хотя бы один момент времени, в который они оба онлайн). Сколько целочисленных моментов времени из отрезка $[l, r]$ для этого подходят?

Входные данные

В первой строке записано четыре целых числа через пробел p, q, l, r ($1 \le p, q \le 50$; $0 \le l \le r \le 1000$).

В каждой из следующих p строк записано два целых числа через пробел a_i, b_i ($0 \le a_i < b_i \le 1000$). В каждой из следующих q строк записано по два целых числа через пробел c_j, d_j ($0 \le c_j < d_j \le 1000$).

Гарантируется, что $b_i < a_{i+1}$ и $d_j < c_{j+1}$ для всех i и j, для которых неравенства имеют смысл.

Выходные данные

Выведите единственное целое число — количество подходящих целочисленных моментов времени отрезка $[l, r]$.

Примеры тестов

входные данные
1 1 0 4 2 3 0 1
выходные данные
3

Алгоритм

Прямой перебор в три вложенных цикла. Сложность $O(n^3)$.

Исходный код

```
#include <iostream>
using namespace std;
int main(int argc, const char * argv[]) {
    int p, q, l, r;
    cin >> p >> q >> l >> r;
    int a[51], b[51], c[51], d[51];
    for(int i=0; i<p; ++i) cin >> a[i] >> b[i];
    for(int i=0; i<q; ++i) cin >> c[i] >> d[i];
    int count = 0;
    for(int i=l; i<=r; ++i) {
        for(int j=0; j<p; ++j) {
            for(int k=0; k<q; ++k) {
                if((b[j]>=c[k]+i && b[j]<=d[k]+i)||(a[j]<=d[k]+i&&a[j]>=c[k]+i)||(c[k]+i>=a[j]&&c[k]+i<=b[j])||(d[k]+i>=a[j]&&d[k]+i<=b[j])) {
                    ++count; goto cont;
                }
            }
        }
        cont:;
    }
    cout << count;
    return 0;
}
```

4.3 Codeforces Отборочный контест СГАУ на четвертьфинал ACM-ICPC

Результаты

Задачи

№	Название				
A	Yet another пусти козла в огород	стандартный ввод/вывод 2 с, 256 МБ			x223
B	Невозможно угадать	стандартный ввод/вывод 2 с, 256 МБ			x167
C	Древний храм	стандартный ввод/вывод 2 с, 256 МБ			x190
D	Игрушечные солдатики	стандартный ввод/вывод 2 с, 256 МБ			x419
E	Просто поменяй слово	стандартный ввод/вывод 2 с, 256 МБ			x345
F	Два конверта	стандартный ввод/вывод 2 с, 256 МБ			x469
G	Задача о размене монет	стандартный ввод/вывод 2 с, 256 МБ			x416
H	Tony Hawk's Pro Skater	стандартный ввод/вывод 2 с, 256 МБ			x93
I	Раскраска карты	стандартный ввод/вывод 2 с, 256 МБ			x149
J	Гипердромы наносят ответный удар	стандартный ввод/вывод 2 с, 256 МБ			x34
K	Два пирата	стандартный ввод/вывод 2 с, 256 МБ			x126
L	Две головы - лучше!	стандартный ввод/вывод 2 с, 256 МБ			x105
M	Построение перестановки	стандартный ввод/вывод 2 с, 256 МБ			x230

Ссылка на контест: http://codeforces.com/gym/100488

Задача F - Два конверта

Mike has two envelopes. He thought up a random integer in the segment $[a, b]$ (he could think up every number from the segment with equal probability) and put exactly this amount of roubles into one of the envelopes and the double amount into the second one. Then he suggested Constantine to play the game: Constantine can open one envelope and then either take its contents or take the contents of another envelope, by his choice.

Constantine has opened one of these envelopes and has discovered c roubles there. Should he take another envelope, if he wants to maximize the expected prize?

Input

The only line contains three integers a, b and c ($1 \le a, b, c \le 10^9$), separated by spaces — the bounds of the segment from which Mike thought up his random number and the amount of roubles in the envelope opened by Constantine. It is guaranteed that the situation described by the input is possible.

Output

Output «`Take another envelope`», if it's advantageous for Constantine to take another envelope, and «`Stay with this envelope`», if it's advantageous for him to stay with the one he has already opened.

Examples

stdin	stdout
3 5 3	Take another envelope
4 6 10	Stay with this envelope

Алгоритм

Задача на теорию вероятности. Если в открытом конверте денег больше чем b, то не нужно менять выбор, иначе нужно поменять, потому что вероятность того, что в другом конверте в 2 раза меньше денег такая же, как и вероятность того, что в другом конверте в 2 раза больше денег. Сложность $O(1)$.

Исходный код

```
#include <iostream>
using namespace std;

int main()
{
    unsigned long a, b, c;
    cin >> a >> b >> c;
    if(c>b)
   cout << "Stay with this envelope";
    else
   cout << "Take another envelope";
    return 0;
}
```

4.4 Codeforces Round 269 Div 2

Результаты

Задачи

№	Название			
A	МУХ и палочки	стандартный ввод/вывод 1 с, 256 МБ		x5496
B	МУХ и важные дела	стандартный ввод/вывод 1 с, 256 МБ		x3473
C	МУХ и карточные домики	стандартный ввод/вывод 1 с, 256 МБ		x2054
D	МУХ и стенки из кубиков	стандартный ввод/вывод 2 с, 256 МБ		x1351
E	МУХ и много-много отрезков	стандартный ввод/вывод 2 с, 512 МБ		x21

Ссылка на контест: http://codeforces.com/contest/471

Задача A - МУХ и палочки

Белым медведям Меньшикову и Усладе из Санкт-Петербургского зоопарка и слонику Хорасу из Киевского зоопарка в рамках проверки их творческих способностей дали для игры шесть палочек. Меньшиков, Услада и Хорас решили собрать из этих палочек либо слона, либо медведя. Животные из палочек собираются так:

- Четыре палочки изображают лапы животного, эти палочки должны быть равными по длине.
- Две оставшиеся палочки изображают голову и тело животного. У медведя палочка-голова должна быть короче палочки-тела. У слона же есть длинный хобот, поэтому палочка-голова слона должна быть такой же длины, как палочка-тело. Обратите внимание, что нет никаких ограничений на отношения между палочками лап и палочками головы и тела животных.

От вас требуется узнать, какое животное можно собрать из заданного набора палочек. После игры палочки необходимо вернуть руководителю зоопарка, поэтому ломать их категорически запрещено, это даже медведям понятно.

Входные данные

В единственной строке через пробел даны шесть чисел l_i ($1 \le l_i \le 9$) — длины шести палочек. Гарантируется, что входные данные таковы, что собрать обоих животных из них никак не получится.

Выходные данные

Если из заданного набора можно собрать медведя, то выведите строку «`Bear`» (без кавычек). Если можно собрать слона, то выведите строку «`Elephant`» (без кавычек). Если ни медведя, ни слона собрать невозможно, то выведите строку «`Alien`» (без кавычек).

Примеры тестов

входные данные

```
4 2 5 4 4 4
```

выходные данные

```
Bear
```

входные данные

```
4 4 5 4 4 5
```

выходные данные

```
Elephant
```

входные данные

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

выходные данные

```
Alien
```

Алгоритм

По заданным длинам палочек идентифицируем существо, которое перед нами. Сложность $O(n)$.

Исходный код

```
#include <cctype>
#include <iostream>
#include <cstring>
#include <cmath>
#include <vector>
#include <algorithm>
#include <set>
#include <stack>
using namespace std;
long min(long a, long b){return (a<b?a:b);}
int main()
{
    vector<int> m(10, 0);
    for(int i=0; i<6; ++i)
    {
        int a;
        cin >> a;
        ++m[a];
    }
    bool beast=0, elephant=0;
    for(int i=1; i<=9; ++i)
    {
        if(m[i] >= 4)
            beast = 1;
        if(m[i] == 2 || m[i]==6)
            elephant = 1;
    }
    if(beast && elephant)
        cout << "Elephant";
    else if(beast)
        cout << "Bear";
    else
        cout << "Alien";
    return 0;
}
```

4.5 Codeforces Round 270 Div 2

Результаты

Задачи

№	Название				
A	Уроки дизайна задач: учимся у математики	стандартный ввод/вывод 1 с, 256 МБ			x7252
B	Уроки дизайна задач: учимся у жизни	стандартный ввод/вывод 1 с, 256 МБ			x4724
C	Уроки дизайна задач: недетеминированность	стандартный ввод/вывод 2 с, 256 МБ			x3769
D	Уроки дизайна задач: обратные задачи	стандартный ввод/вывод 2 с, 256 МБ			x1718
E	Уроки дизайна задач: учимся у игр	стандартный ввод/вывод 1 с, 256 МБ			x66
F	Уроки дизайна задач: меняем цель задачи	стандартный ввод/вывод 2 с, 256 МБ			x120
G	Уроки дизайна задач: увеличиваем ограничения	стандартный ввод/вывод 7 с, 256 МБ			x94

Ссылка на контест: http://codeforces.com/contest/472

Задача A - Уроки дизайна задач: учимся у математики

Один из способов придумать новую задачу: использовать математику. Например, можно придумать какое-нибудь рандомное математическое утверждение или модифицировать некоторые теоремы, чтобы получить что-то новое. Используя такие методы, можно придумать новую задачу.

Например, есть утверждение под названием «Гипотеза Гольдбаха». Оно гласит: «каждое четное число не менее четырех можно представить в виде суммы двух простых чисел». Давайте модифицируем его следующим образом: «каждое целое число не менее 12 можно представить в виде суммы двух составных чисел». В отличие от гипотезы Гольдбаха, я могу доказать эту гипотезу.

Вам дано целое число n не менее 12, представьте его в виде суммы двух составных чисел.

Входные данные
В единственной строке записано целое число n ($12 \le n \le 1000000$).

Выходные данные
Выведите два таких составных целых числа x и y ($1 < x, y < n$), что $x + y = n$. Если есть несколько правильных ответов, можно вывести любой из них.

Примеры тестов

входные данные
12
выходные данные
4 8

Алгоритм

Нужно найти такие числа a и b, чтобы они не были простыми, а $a + b = n$. Для этого положим $a = 4$, а $b = n - 4$. Далее будем проверять, являются ли числа простыми. Если хоть одно из них простое, то инкрементирум a и декрементируем b. В конце концов таким образом найдутся искомые a и b. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <cmath>
using namespace std;

bool prime(long long n){
    for(long long i=2;i<=sqrt(n);i++)
        if(n%i==0)
            return false;
    return true;
}

int main() {
    int n;
    cin >> n;
    for(int i=4; i<n-4; ++i) {
        if(!prime(i) && !prime(n-i)) {
            cout << i << ' ' << n-i;
            break;
        }
    }
    return 0;
}
}
```

Задача B - Уроки дизайна задач: учимся у жизни

Один из способов придумывания задач — наблюдать за процессами в реальной жизни. Можно выбрать какую-то реально существующую ситуацию из жизни, формализовать ее и так получить новую задачу.

Представим жизненную ситуацию: стоит много людей, все ждут лифта. Каждый человек хочет попасть на какой-то конкретный этаж. Мы можем формализовать это следующим образом. Пусть n людей стоит на первом этаже, и i-й человек хочет попасть на f_i-й этаж. К сожалению, в здании только один лифт, способный вместить k человек (иными словами, одновременно лифт могут использовать не более k человек). Изначально лифт расположен на первом этаже. Лифту требуется $|a - b|$ секунд для того, чтобы доехать от a-го до b-го этажа (время, необходимое пассажирам лифта на вход и выход, не учитывается).

Какое минимальное количество секунд необходимо для того, чтобы развезти всех людей по необходимым этажам и затем вернуть лифт на первый этаж?

Входные данные

В первой строке записано два целых числа n и k $(1 \le n, k \le 2000)$ — количество людей и максимальная вместимость лифта.

В следующей строке записано n целых чисел: $f_1, f_2, \ldots, f_n$ $(2 \le f_i \le 2000)$, где f_i обозначает этаж, до которого хочет доехать i-й человек.

Выходные данные

Выведите единственное целое число — минимальное время, необходимое для достижения поставленной цели.

Примеры тестов

входные данные
3 2 2 3 4
выходные данные
8

Алгоритм

Чтобы доставить всех людей на нужные этажи за наименьшее количество времени, отсортируем всех по номеру этажа и будем доставлять их порциями по вместимости лифта. Сложность $O(nlog(n))$.

Исходный код

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#define ll long long
using namespace std;
int main() {
    ll n, k;
    cin >> n >> k;
    vector<ll> v(n);
    for(ll i=0; i<n; ++i)
        cin >> v[i];
    sort(v.begin(), v.end());
    ll i = n-1;
    ll time = 0;
    while(i>=0) {
        time += v[i]*2-2;
        i -= k;
    }
    cout << time;
    return 0;
}
```

4.6 Codeforces Round 272 Div 2

Результаты

Задачи

№	Название			
A	Dreamoon и ступеньки	стандартный ввод/вывод 1 с, 256 МБ		x5308
B	Dreamoon и WiFi	стандартный ввод/вывод 1 с, 256 МБ		x3709
C	Dreamoon и суммы	стандартный ввод/вывод 1,5 с, 256 МБ		x1584
D	Dreamoon и множества	стандартный ввод/вывод 1 с, 256 МБ		x847
E	Dreamoon и строки	стандартный ввод/вывод 1 с, 256 МБ		x280

Ссылка на контест: http://codeforces.com/contest/476

Задача A - Dreamoon и ступеньки

Dreamoon хочет подняться по лестнице, состоящей из n ступенек. За один шаг он может преодолеть 1 или 2 ступеньки. При этом, Dreamoon хочет, чтобы количество шагов было кратно целому числу m.

Какое минимальное количество шагов ему придётся сделать, чтобы подняться, выполнив своё условие?

Входные данные
В единственной строке записано два целых числа через пробел — n, m ($0 < n \le 10000$, $1 < m \le 10$).

Выходные данные
Выведите единственное число — минимальное количество шагов, кратное m. Если способа взобраться по лестнице, выполнив условие задачи, не существует, выведите -1.

Примеры тестов

входные данные
10 2
выходные данные
6

входные данные
3 5
выходные данные
-1

Алгоритм

Сначала поделим количество ступенек на два, затем будем прибавлять по единице, пока d не делится на m без остатка. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, const char * argv[]) {
    int n, m;
    cin >> n >> m;
    if(m>n)
    {
        cout << "-1";
        return 0;
    }
    int d = n/2+(n%2?1:0);
    while(d%m) ++d;
    cout << d;
    return 0;
}
```

Задача B - Dreamoon и WiFi

Dreamoon стоит на отметке 0 на числовой прямой. Drazil отправляет ему на смартфон список команд через Wi-Fi, а Dreamoon следует им.

Каждая команда относится к одному из двух типов:

1. Пройти на 1 единицу в положительном направлении, обозначается как «+»
2. Пройти на 1 единицу в отрицательном направлении, обозначается как «−»

Но так как уровень сигнала Wi-Fi очень слабый, смартфон Dreamoon'a сообщает, что некоторые команды не поддаются распознаванию. Более того, Dreamoon знает, что даже некоторые из рапознанных команд могут быть неверными. Dreamoon решил выполнять каждую распознанную команду, а для нераспознанных команд бросать монетку (это значит, что он пойдёт на 1 единицу в положительном или отрицательном направлении с одинаковой вероятностью 0.5).

Вам дан изначальный список команд, которые выслал Drazil, и список, который получил Dreamoon. Какова вероятность того, что Dreamoon окажется в том положении, к которому привели бы команды Drazil'a?

Входные данные
В первой строке записана строка s_1 — команды, которые Drazil посылает Dreamoon'y, строка состоит только из символов из набора {«+», «−»}.

Во второй строке записана строка s_2 — команды, которые распознал смартфон Dreamoon'a, эта строка состоит только из символов из набора: {«+», «−», «?»}. Символом «?» обозначается нераспознанная команда.

Обе строки одинаковой длины и не превышают 10.

Выходные данные
Выведите требуемую вероятность. Ответ будет сраниваться с относительной или абсолютной погрешностью 10^{-9}.

Примеры тестов

входные данные
++-+- +-+-+
выходные данные
1.000000000000

входные данные
+-+- +-??
выходные данные
0.500000000000

входные данные
+++ ??-
выходные данные
0.000000000000

Алгоритм

Для подсчёта вероятности используем формулу перестановок. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
unsigned long factorial(unsigned long a)
{
    unsigned long res = 1;
    if(!a)
        return 1;
    for(long i=1; i<=a; ++i)
        res*=i;
    return res;
}
int main(int argc, const char * argv[]) {
    char c;
    int aPlus=0, aMinus=0, bPlus=0, bMinus=0, bQuest=0;
    while((c=cin.get())!='\n')
    {
        if(c=='+')
            ++aPlus;
        else if(c=='-')
            ++aMinus;
    }
    for(int i=0; i<aPlus+aMinus; ++i)
    {
        c=cin.get();
        if(c=='+')
            ++bPlus;
        else if(c=='-')
            ++bMinus;
        else if(c=='?')
            ++bQuest;
    }
    cout << fixed;
    cout << setprecision(10);
    if(aPlus<=bPlus+bQuest && aMinus<=bMinus+bQuest &&  aPlus+aMinus-(bPlus+bMinus)-bQuest == 0)
    {
        if(!bQuest)
            cout << 1;
        else
        {
            unsigned long pDiff=aPlus-bPlus;
            cout << factorial(bQuest)/(factorial(pDiff)*factorial(bQuest-pDiff))/pow(2, bQuest);
        }
    }
    else
        cout << 0;
    return 0;
}
```

4.7 Codeforces Round 273 Div 2

Результаты

Задачи

№	Название				
A	Начальная ставка	стандартный ввод/вывод 1 с, 256 МБ			x6305
B	Случайные команды	стандартный ввод/вывод 1 с, 256 МБ			x4483
C	Украшение столов	стандартный ввод/вывод 1 с, 256 МБ			x2677
D	Красно-зеленые башни	стандартный ввод/вывод 2 с, 256 МБ			x1171
E	Волнистые числа	стандартный ввод/вывод 1,5 с, 256 МБ			x66

Ссылка на контест: http://codeforces.com/contest/478

Задача A - Начальная ставка

В игру «Щедрость» играют пять человек. Каждый из них вносит некоторое ненулевое количество монет b в качестве начальной ставки. После того, как все игроки сделали ставку в b монет, некоторое число раз повторяется следующая операция: у одного из игроков берется одна монета и отдается какому-то другому игроку.

Ваша задача — написать программу, которая по количеству монет у каждого из игроков в конце игры определит размер начальной ставки b или определит, что такой итог игры не мог быть получен ни при каком положительном количестве монет b в начальной ставке.

Входные данные

Ввод состоит из единственной строки, содержащей пять целых чисел c_1, c_2, c_3, c_4 и c_5 — количество монет в конце игры у первого, второго, третьего, четвертого и пятого игрока соответственно ($0 \le c_1, c_2, c_3, c_4, c_5 \le 100$).

Выходные данные

В единственной строке требуется вывести единственное положительное целое число b — количество монет в начальной ставке каждого из игроков. Если не существует такого значения b, то в единственной строке выходных данных требуется вывести «`-1`» (без кавычек).

Примеры тестов

входные данные
`2 5 4 0 4`
выходные данные
`3`

Алгоритм

В этой задаче достаточно заметить, что суммарное количество монет должно без остатка делиться на 5. Иначе исход игры невозможен. Сложность $O(1)$.

Исходный код

```
#include <iostream>
using namespace std;
int main(int argc, const char * argv[]) {
    int a;
    int s = 0;
    for(int i=0; i<5; ++i)
    {
        cin >> a;
        s+=a;
    }
    if(s%5 || !s)
        cout << -1;
    else
        cout << s/5;
    return 0;
}
```

Задача B - Случайные команды

Для участия в соревнованиях n участников были разбиты некоторым образом на m команд так, чтобы в каждой команде был хотя бы один участник. После соревнований каждая пара участников из одной команды стала друзьями.

Ваша задача — написать программу, которая определит, какое минимальное и какое максимальное количество пар друзей могло образоваться после соревнования.

Входные данные

В единственная строке содержатся два целых числа n и m, разделенных одним пробелом ($1 \le m \le n \le 10^9$) — количество участников и количество команд соответственно.

Выходные данные

Требуется вывести два целых числа k_{min} и k_{max} — минимальное возможное количество пар друзей и максимальное возможное количество пар друзей соответственно.

Примеры тестов

входные данные
5 1
выходные данные
10 10

Алгоритм

Задача на комбинаторику. Минимальное количество получается если всех разделить на команды поровну, а максимальное если в одну из команд поместить максимально возможное количество участников. Сложность $O(nlog(1))$.

Исходный код

```
#include <iostream>
using namespace std;

int main(int argc, const char * argv[]) {
    unsigned long long m, n;
    cin >> m >> n;
    unsigned long long min, max;
    unsigned long long k = (m-n+1);
    max = k*(k-1)/2;
    unsigned long long r = n%n;
    if(r)
    {
        k = m/n;
        min = k*(k-1)/2*(n-r)+(k+1)*k/2*r;
    }
    else
    {
        k = m/n;
        min = k*(k-1)/2*n;
    }
    if(min>max)
    {
        cout << max << ' ' << min;
    }
    else
        cout << min << ' ' << max;
    return 0;
}
```

4.8 Codeforces Round 274 Div 2

Результаты

Задачи

№	Название			
A	Выражение	стандартный ввод/вывод 1 с, 256 МБ		x6249
B	Башни	стандартный ввод/вывод 1 с, 256 МБ		x2943
C	Экзамены	стандартный ввод/вывод 1 с, 256 МБ		x2947
D	Прыжки в длину	стандартный ввод/вывод 1 с, 256 МБ		x1197
E	Катаемся на лифте	стандартный ввод/вывод 2 с, 256 МБ		x685

Ссылка на контест: http://codeforces.com/contest/480

Задача A - Выражение

Петя учится в школе и очень любит математику. Уже несколько занятий они с классом проходят арифметические выражения. На последнем уроке учительница написала на доске три положительных целых числа a, b, c. Задание заключалось в том, чтобы расставить между этими числами знаки операций '+' и '*', а также, возможно, скобки. Значение получившегося выражения должно быть как можно больше. Рассмотрим пример: пусть учительница выписала на доску числа 1, 2 и 3. Вот некоторые варианты расстановки знаков и скобок:

- 1+2*3=7
- 1*(2+3)=5
- 1*2*3=6
- (1+2)*3=9

Обратите внимание на то, что знаки операций можно вставлять только между a и b, а также между b и c, то есть нельзя менять числа местами. Так, в приведенном примере нельзя получить выражение (1+3)*2.

Легко убедиться, что максимальное значение, которое можно получить, — это 9.

Ваша задача — по заданным a, b и c вывести, какое максимальное значение выражения можно получить.

Входные данные
Во входных данных записаны три целых числа a, b и c, каждое в отдельной строке ($1 \le a, b, c \le 10$).

Выходные данные
Выведите максимальное значение выражения, которое можно получить.

Примеры тестов

входные данные
1 2 3
выходные данные
9

Алгоритм

Напишем условия для каждого случая. Таким образом программа будет работать за $O(1)$.

Исходный код

```
#include <iostream>
using namespace std;
int main(int argc, const char * argv[]) {
    int a, b, c;
    cin >> a >> b >> c;
    if(a==1) {
        if(c==1) cout << a+b+c;
        else cout << (a+b)*c;
    } else {
        if(b==1) {
            if(a>c) cout << a*(b+c);
            else cout << (a+b)*c;
        } else {
            if(c==1) cout << a*(b+1);
            else cout << a*b*c;
        }
    }
    return 0;
}
```

Задача B - Башни

Как известно, все дети в Берляндии любят играть с кубиками. У маленького Пети имеется n башен, состоящих из кубиков одинакового размера. Башня под номером i представляет собой a_i кубиков, поставленных друг на друга. *Неустойчивостью* набора башен Петя называет величину, равную разности высот самой высокой и самой низкой башни. К примеру, если Петя построил из кубиков пять башен с высотами (8, 3, 2, 6, 3), то неустойчивость этого набора равна 6 (самая высокая башня имеет высоту 8, самая низкая — высоту 2).

Мальчик хочет, чтобы неустойчивость его набора башен была как можно меньше. Все, что он может сделать, это несколько раз проделать следующую операцию: взять верхний кубик с какой-то из башен и положить его сверху на какую-то другую башню из своего набора. Обратите внимание, что Петя никогда не будет класть кубик на ту же башню, с которой тот был снят, потому что считает это пустой тратой времени.

Прежде чем отправиться в школу, мальчик успеет произвести не более k таких операций. Петя не хочет опоздать на урок, поэтому вам придется помочь ему выполнить эту задачу.

Входные данные

В первой строке через пробел записаны два целых положительных числа n и k ($1 \le n \le 100$, $1 \le k \le 1000$) — количество башен в имеющемся наборе и максимальное число операций, которые Петя может произвести. Во второй строке через пробел записаны n целых положительных чисел a_i ($1 \le a_i \le 10^4$) — исходные высоты башен.

Выходные данные

В первой строке выведите через пробел два целых неотрицательных числа s и m ($m \le k$). Первое из чисел — это величина минимально возможной неустойчивости, которую можно достичь, применив не более k операций, а второе — количество операций, необходимых для этого.

Затем в m строках выведите описание каждой из операций в виде двух целых положительных чисел i и j, каждое из которых лежит в пределах от 1 до n. Они обозначают, что Петя переложил верхний кубик с i-й башни на j-ю ($i \ne j$). Обратите внимание, что в процессе выполнения операций высоты некоторых башен могут стать равны нулю.

Если существует несколько корректных последовательностей операций, при которых достигается минимально возможная неустойчивость, разрешается вывести любую из них.

Примеры тестов

входные данные
3 2 5 8 5
выходные данные
0 2 2 1 2 3

Алгоритм

Отсортируем башни по высоте и будем перекладывать кубик с самой высокой на самую низкую. После этого нужно снова отсортировать новые высоты. Это нужно проделать k раз. Сложность алгоритма $O(knlog(n))$

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
struct myPair {
    int from, to;
};
struct Tower {
    int id, h;
};
bool compareByH(const Tower &a, const Tower &b) {
    return a.h < b.h;
}
int main(int argc, const char * argv[]) {
```

```
    int n, k;
    vector<Tower> m(101);
    myPair mP[1001];
    bool u[101];
    cin >> n >> k;
    for(int i=0; i<n; ++i)
    {
        cin >> m[i].h;
        m[i].id = i+1;
        u[i] = 0;
    }
    sort(m.begin(), m.begin()+n, compareByH);

    int minK = k, diff = -1;
    for(int i=0; i<k; ++i)
    {
        int min, max;
        int minIndex = 0, maxIndex = n-1;
        while(minIndex<n-1 && m[minIndex].h>=m[minIndex+1].h && !u[minIndex+1])++
    minIndex;
        while(maxIndex>0 && m[maxIndex].h<=m[maxIndex-1].h)--maxIndex;
        min = m[minIndex].h;
        max = m[maxIndex].h;
        u[maxIndex] = 1;
        if(u[minIndex])
        {
            minK = i;
            break;
        }
        if(max-min == 1)
        {
            diff = 1;
            minK = i;
            break;
        }
        else if(max-min == 0)
        {
            diff = 0;
            minK = i;
            break;
        }
        mP[i].from = m[maxIndex].id;
        mP[i].to = m[minIndex].id;
        --m[maxIndex].h;
        ++m[minIndex].h;
    }
    if(diff == -1)
    {
        int minIndex = 0, maxIndex = n-1;
        while(minIndex<n-1 && m[minIndex].h>=m[minIndex+1].h)++minIndex;
        while(maxIndex>0 && m[maxIndex].h<=m[maxIndex-1].h)--maxIndex;
        diff = m[maxIndex].h - m[minIndex].h;
    }
    cout << diff << ' ' << minK << '\n';
    for(int i=0; i<minK; ++i)
    {
        cout << mP[i].from << ' ' << mP[i].to << '\n';
    }
    return 0;
}
```

Задача C - Экзамены

Студент Валера учится на первом курсе университета. Скоро у него сессия, и ему предстоит сдать ровно n экзаменов. Валера — умный парень, поэтому он сможет сдать любой экзамен с первого раза. Кроме того, он может сдавать несколько экзаменов в один день и в любом порядке.

Согласно расписанию, экзамен по i-му предмету нужно сдать в день с номером a_i. Однако Валера договорился с каждым преподавателем, и преподаватель i-го предмета разрешил организовать досрочную сдачу своего экзамена в день b_i ($b_i < a_i$). Таким образом, Валера может сдать экзамен по i-му предмету либо в день a_i, либо в день b_i. Все преподаватели ставят запись о сдаче экзамена в зачетную книжку в день фактической сдачи экзамена и датируют эту запись числом a_i.

Валера считает, что будет достаточно странно, если записи в зачетной книжке будут идти не в порядке неубывания даты. Поэтому Валера просит вас помочь ему. Найдите минимально возможный номер дня, когда Валера сможет сдать последний экзамен, если он будет сдавать экзамены так, чтобы все записи в его зачетной книжке шли в порядке неубывания даты.

Входные данные

В первой строке записано единственное целое положительное число n ($1 \le n \le 5000$) — количество экзаменов, которые будет сдавать Валера.

В каждой из следующих n строк записано по два целых положительных числа через пробел a_i и b_i ($1 \le b_i < a_i \le 10^9$) — дата сдачи по расписанию и досрочная дата сдачи i-го экзамена соответственно.

Выходные данные

Выведите единственное целое число — минимально возможный номер дня, когда Валера сможет сдать последний экзамен, если он будет сдавать экзамены так, чтобы все записи в его зачетной книжке шли в порядке неубывания даты.

Примеры тестов

входные данные
3 5 2 3 1 4 2
выходные данные
2

Алгоритм

Для решения этой задачи нужно отсортировать экзамены сначала по дате сдачи по расписанию, а затем устойчиво отсортировать по датам досрочной сдачи. Потом нужно пройти по отсортированным датам и если у текущего экзамена досрочная дата раньше чем досрочная предыдущего, то текущий нужно сдавать по расписанию. Сложность алгоритма $O(nlog(n))$.

Исходный код

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
bool compareFirst(const pair<int, int> &a, const pair<int, int> &b)
{
    return a.first < b.first;
}
bool compareSecond(const pair<int, int> &a, const pair<int, int> &b)
{
    return a.second < b.second;
}
int main(int argc, const char * argv[]) {
    int n;
    cin >> n;
    vector< pair<int,int> > m(n);
    for(int i=0; i<n; ++i)
        cin >> m[i].first >> m[i].second;
    stable_sort(m.begin(), m.end(), compareSecond);
    stable_sort(m.begin(), m.end(), compareFirst);
    int res = m[0].second;
    for(int i=1; i<n; ++i)
    {
        if(m[i].second >= res)
            res = m[i].second;
        else
            res = m[i].first;
    }
    cout << res;
    return 0;
}
```

4.9 Codeforces Round 275 Div 2

Результаты

Задачи

№	Название			
A	Опровержение гипотез	стандартный ввод/вывод 1 с, 256 МБ		x4242
B	Друзья и подарки	стандартный ввод/вывод 1 с, 256 МБ		x1627
C	Разнообразная перестановка	стандартный ввод/вывод 1 с, 256 МБ		x2212
D	Интересный массив	стандартный ввод/вывод 1 с, 256 МБ		x402
E	Игра со строками	стандартный ввод/вывод 1 с, 256 МБ		x28

Ссылка на контест: http://codeforces.com/contest/483

Задача A - Опровержение гипотез

Перестановкой p называется упорядоченный набор чисел $p_1, p_2, \ldots, p_n$, состоящий из n различных целых положительных чисел, каждое из которых не больше чем n. Число n будем называть длиной перестановки $p_1, p_2, \ldots, p_n$.

Ваша задача — найти такую перестановку p длины n, что среди чисел $|p_1 - p_2|, |p_2 - p_3|, \ldots, |p_{n-1} - p_n|$ ровно k различных.

Входные данные

В единственной строке входных данных находятся два разделённых пробелом целых положительных числа n, k ($1 \le k < n \le 10^5$).

Выходные данные

Выведите n целых чисел — искомую перестановку. Если существует несколько ответов, разрешается вывести любой.

Примеры тестов

входные данные
3 2
выходные данные
1 3 2

Алгоритм

Используем обычный перебор. Сложность $O(n^3)$.

Исходный код

```
#include <iostream>
using namespace std;
unsigned long long min(unsigned long long a, unsigned long long b){return (a>b)?a:b;}
unsigned long long gcd (unsigned long long a, unsigned long long b) {
    while (b) {
        a %= b;
        swap (a, b);
    }
    return a;
}
int main(int argc, const char * argv[]) {
    unsigned long long l,r;
    cin >> l >> r;
    for(unsigned long long a=l; a<=r; ++a)
    {
        for(unsigned long long b=a+1; b<r; ++b)
        {
            for(unsigned long long c=b+1; c<=r; ++c)
            {
                if(gcd(a,b)==1 && gcd(b,c)==1 && gcd(a,c)>1)
                {
                    cout << a << ' ' << b << ' ' << c;
                    return 0;
                }
            }
        }
    }
    cout << -1;
    return 0;
}
```

Задача C - Разнообразная перестановка

Перестановкой p называется упорядоченный набор чисел $p_1, p_2, \ldots, p_n$, состоящий из n различных целых положительных чисел, каждое из которых не больше чем n. Число n будем называть длиной перестановки $p_1, p_2, \ldots, p_n$.

Ваша задача — найти такую перестановку p длины n, что среди чисел $|p_1 - p_2|, |p_2 - p_3|, \ldots, |p_{n-1} - p_n|$ ровно k различных.

Входные данные

В единственной строке входных данных находятся два разделённых пробелом целых положительных числа n, k ($1 \le k < n \le 10^5$).

Выходные данные

Выведите n целых чисел — искомую перестановку. Если существует несколько ответов, разрешается вывести любой.

Примеры тестов

входные данные
3 2
выходные данные
1 3 2

Алгоритм

Получить искомую перестановку можно записывая поочередно $b = 1$ и $e = n$. После записи b нужно инкрементировать, а e декрементировать. В результате будет получена искомая перестановка. Сложность алгоритма $O(n)$.

Исходный код

```
#include <iostream>
#include <cmath>
#define MAX 100002
using namespace std;
int main(int argc, const char * argv[]) {
    long n, k;
    cin >> n >> k;
    cout << "1 ";
    long a = k+1, b=2;
    for(unsigned long long i=1; i<=k; ++i)
    {
        if(i%2)
            cout << a--<< ' ';
        else
            cout << b++ << ' ';
    }
    for(long i=k+2; i<=n; ++i)
    {
        cout << i << ' ';
    }
    return 0;
}
```

4.10 Codeforces Round 276 Div 2

Результаты

Задачи

№	Название				
A	Фабрика	стандартный ввод/вывод 1 с, 256 МБ			x3184
B	Ценные ресурсы	стандартный ввод/вывод 1 с, 256 МБ			x3551
C	Биты	стандартный ввод/вывод 1 с, 256 МБ			x1499
D	Максимальное значение	стандартный ввод/вывод 1 с, 256 МБ			x468
E	Странная сортировка	стандартный ввод/вывод 2 с, 256 МБ			x32

Ссылка на контест: http://codeforces.com/contest/485

Задача A - Фабрика

На одной промышленной фабрике проходит реформа рабочего дня. Директор предложил установить норму производства мифических деталей. Если к началу рабочего дня на заводе находилось x деталей, то по плану к концу должно быть произведено еще $x \bmod m$ (остаток от деления x на m) деталей. К сожалению, пока не нашлось ни одного покупателя на мифические детали, поэтому все произведенные детали остаются на заводе.

Совет директоров обеспокоен, не остановится ли производство по такому плану в один прекрасный день (а именно, не настанет ли момент, когда текущее количество деталей на заводе делится на m).

По заданному количеству деталей a в первый день и числу m проверьте, остановится ли производство.

Входные данные

В первой строке содержатся два целых числа a и m ($1 \le a, m \le 10^5$).

Выходные данные

Выведете «`Yes`» (без кавычек), если производство может остановиться и «`No`» иначе.

Примеры тестов

входные данные
1 5
выходные данные
No

входные данные
3 6
выходные данные
Yes

Алгоритм

Производство не остановится тогда, когда t будет повторяться. Чтобы это проверить нужно создать булевский массив и заполнить его нулями. В цикле присваивать единицу элементу с индексом t, если элемент уже равен единице, то вывести No.

Исходный код

```
#include <cctype>
#include <iostream>
#include <cstring>
#include <cmath>
#include <vector>
#include <algorithm>
#include <set>
#include <stack>
using namespace std;
long min(long a, long b){return (a<b?a:b);}
bool prime(long long n){
    for(long long i=2;i<=sqrt(n);i++)
        if(n%i==0)
            return false;
    return true;
}
int main()
{
    long long a, m;
    cin >> a >> m;
    vector<bool> v(m+1, 0);
    while(a)
    {
        long long t = a%m;
        if(!t)
        {
            cout << "Yes";
            return 0;
        }
        if(v[t])
        {
            cout << "No";
            return 0;
        }
        v[t] = 1;
        a += t;
    }
    cout << "No";
    return 0;
}
```

Задача B - Ценные ресурсы

Во многих компьютерных играх стратегического жанра нужно строить города, создавать армию, захватывать племена, накапливать ресурсы. Иногда это приводит к интересным задачам.

Предположим, вам требуется построить город квадратной формы. На карте мира введена декартова система координат. Стороны города должны быть параллельны осям координат. На карте присутствуют рудники с ценными ресурсами, расположенные в точках с целочисленными координатами. Размерами рудников можно пренебречь, то есть их можно считать точками. Город должен быть построен так, чтобы все рудники были внутри или на границе квадрата, которым он образован.

Так как строительство города требует затрат, зависящих от его размера, то необходимо построить город с минимальной площадью. По заданному расположению рудников найдите минимальную возможную площадь города.

Входные данные

Первая строка ввода содержит целое число n — количество рудников на карте ($2 \le n \le 1000$). В каждой из n последующих строк содержится по паре целых чисел x_i и y_i — координаты соответствующего рудника ($-10^9 \le x_i, y_i \le 10^9$). Все точки попарно различны.

Выходные данные

Выведите минимальную площадь города, которым можно покрыть все рудники с ценными ресурсами.

Примеры тестов

входные данные

```
2
0 0
2 2
```

выходные данные

```
4
```

входные данные

```
2
0 0
0 3
```

выходные данные

```
9
```

Алгоритм

Нужно найти минимальную и максимальную координаты и посчитать площадь прямоугольника между ними. Сложность $O(n)$.

Исходный код

```
#include <cctype>
#include <iostream>
#include <cstring>
#include <cmath>
#include <vector>
#include <algorithm>
#include <set>
#include <stack>
using namespace std;
long min(long a, long b){return (a<b?a:b);}

int main()
{
    long long n;
    cin >> n;
    long long minX, minY, maxX, maxY;
    cin >> minX >> minY;
    maxX = minX;
    maxY = minY;
    for(int i=1; i<n; ++i)
    {
        long long a, b;
        cin >> a >> b;
        if(a<minX)
            minX = a;
        else if(a>maxX)
            maxX = a;
        if(b<minY)
            minY = b;
        else if(b>maxY)
            maxY = b;
    }
    long long t = max(abs(maxX-minX), abs(maxY-minY));
    cout << t*t;
    return 0;
}
```

Задача C - Биты

Обозначим за $\mathrm{popcount}(x)$ количество единиц в двоичной записи целого неотрицательного числа x.

Вам дано несколько запросов, каждый в виде пары целых чисел l и r. Для каждого запроса найдите такое x, что $l \le x \le r$, а $\mathrm{popcount}(x)$ максимально. Если таких чисел несколько, то требуется найти наименьшее из них.

Входные данные
Первая строка содержит целое число n — количество запросов ($1 \le n \le 10000$).

Следующие n строк содержат по два числа l_i, r_i — параметры для очередного запроса ($0 \le l_i \le r_i \le 10^{18}$).

Выходные данные
На каждый запрос выведите требуемое число в отдельной строке.

Примеры тестов

входные данные
3 1 2 2 4 1 10
выходные данные
1 3 7

Алгоритм

Заполняем число, изначально равное нижней границе, в двоичном представлении единицами и до тех пор пока оно не приблизится к верхней. Сложность $O(log(n))$.

Исходный код

```
#include <cctype>
#include <iostream>
#include <vector>
using namespace std;
long min(long a, long b){return (a<b?a:b);}
int main() {
    long long n;
    cin >> n;
    long long minX, minY, maxX, maxY;
    cin >> minX >> minY;
    maxX = minX; maxY = minY;
    for(int i=1; i<n; ++i) {
        long long a, b;
        cin >> a >> b;
        if(a<minX)
            minX = a;
        else if(a>maxX)
            maxX = a;
        if(b<minY)
            minY = b;
        else if(b>maxY)
            maxY = b;
    }
    long long t = max(abs(maxX-minX), abs(maxY-minY));
    cout << t*t;
    return 0;
}
```

4.11 Codeforces Round 277 Div 2

Результаты

Задачи

№	Название			
A	Подсчёт функции	стандартный ввод/вывод 1 с, 256 МБ		x6702
B	OR в матрице	стандартный ввод/вывод 1 с, 256 МБ		x3235
C	Преобразование в палиндром	стандартный ввод/вывод 1 с, 256 МБ		x2332
D	Допустимые множества	стандартный ввод/вывод 1 с, 256 МБ		x903
E	Наибольшие возрастающие подпоследовательности	стандартный ввод/вывод 2 с, 256 МБ		x531

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

Задача A - Подсчёт функции

Для положительного целого числа n определим функцию f:

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

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

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

Выходные данные
Выведите $f(n)$ в единственной строке.

Примеры тестов

входные данные
4
выходные данные
2

входные данные
5
выходные данные
-3

Алгоритм

Выводим входное число делённое на два, округленное в большую сторону и умноженное на -1, если входное число нечётное. Сложность $O(1)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <climits>
#define ll long long
#define ull unsigned long long
using namespace std;
int main()
{
    ll n = 0;
    cin >> n;
    ll r = n-n/2;
    cout << (n&1?-r:r);
    return 0;
}
```

Задача B - OR в матрице

Определим логический OR как операцию над двумя логическими значениями (то есть, значениями, которые принадлежат множеству $\{0, 1\}$), которая даёт 1, если один или оба операнда равны 1, в противном случае операция даёт 0. Можно определить логическое OR трех или более логических значений аналогичным образом:

$a_1 \text{ OR } a_2 \text{ OR} \ldots \text{OR } a_k$, где $a_i \in \{0, 1\}$, равняется 1, если какое-то из $a_i = 1$, в противном случае результат равняется 0.

У Нама есть матрица A, состоящая из m строк и n столбцов. Строки пронумерованы от 1 до m, столбцы пронумерованы от 1 до n. Элемент в строке i ($1 \le i \le m$) и столбце j ($1 \le j \le n$) обозначается как A_{ij}. Все элементы A равны либо 0, либо 1. По матрице A Нам строит матрицу B того же размера, используя следующую формулу:

$$B_{ij} = A_{i1} \text{ OR } A_{i2} \text{ OR} \ldots \text{OR } A_{in} \text{ OR } A_{1j} \text{ OR } A_{2j} \text{ OR} \ldots \text{OR } A_{mj}.$$

(B_{ij} — OR всех элементов в строке i со всеми элементами столбца j матрицы A)

Нам дает вам матрицу B и бросает вызов: сможете ли вы угадать матрицу A? Хотя Нам умен, он вполне мог ошибиться в подсчете матрицы B, так как размеры матрицы A могут быть очень большими.

Входные данные

В первой строке записано два целых числа m и n ($1 \le m, n \le 100$) — количество строк и колонок матриц соответственно.

В следующих m строках записано по n целых чисел через пробелы — строки матрицы B (каждый элемент B равен либо 0, либо 1).

Выходные данные

В первой строке выведите «NO», если Нам ошибся при подсчитывании B, в противном случае выведите «YES». Если первая строка равна «YES», то выведите также m строк по n целых чисел — матрицу A, по которой могла быть получена данная матрица B. Если решений несколько, выведите любое.

Примеры тестов

входные данные

```
2 2
1 0
0 0
```

выходные данные

```
NO
```

входные данные

```
2 3
1 1 1
1 1 1
```

выходные данные

```
YES
1 1 1
1 1 1
```

входные данные

```
2 3
0 1 0
1 1 1
```

выходные данные

```
YES
0 0 0
0 1 0
```

Алгоритм

Сложность $O(n^3)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <climits>
#define ll long long
#define ull unsigned long long
using namespace std;
int main()
{
    ll m, n;
    cin >> m >> n;
    bool matrix[101][101];
    bool newMatrix[101][101];
    for(ll i=0; i<m; ++i)
    {
        for(ll j=0; j<n; ++j)
        {
            cin >> matrix[i][j];
            newMatrix[i][j] = 0;
        }
    }
    vector<bool> markx(100, 0);
    vector<bool> marky(100, 0);
    for(ll i=0; i<m; ++i)
    {
        for(ll j=0; j<n; ++j)
        {
            if(matrix[i][j])
            {
                bool ok1 = 1, ok2 = 1, okok1=0, okok2=0;
                for(ll ii=0; ii<m; ++ii)
                {
                    bool f = 1;
                    for(ll jj = 0; jj<n; ++jj)
                    {
                        if(!matrix[ii][jj])
                        {
                            f = 0;
                            break;
                        }
                    }
                    if(f)
                    {
                        okok1 = 1;
                    }
                    if(!matrix[ii][j])
                    {
                        ok1 = 0;
                        break;
                    }
                }
                for(ll jj=0; jj<n; ++jj)
                {
                    bool f = 1;
                    for(ll ii = 0; ii<m; ++ii)
                    {
                        if(!matrix[ii][jj])
```

```
                        {
                            f = 0;
                            break;
                        }
                    }
                    if(f)
                    {
                        okok2 = 1;
                    }
                    if(!matrix[i][jj])
                    {
                        ok2 = 0;
                        break;
                    }
                }
                if(!((ok1&&okok1) || (ok2&&okok2)))
                {
                    cout << "NO";
                    return 0;
                }
            }
        }
    }
    for(ll i=0; i<m; ++i)
    {
        for(ll j=0; j<n; ++j)
        {
            bool ok1 = 1, ok2 = 1;
            for(ll ii=0; ii<m; ++ii)
            {
                if(!matrix[ii][j])
                {
                    ok1 = 0;
                    break;
                }
            }
            for(ll jj=0; jj<n; ++jj)
            {
                if(!matrix[i][jj])
                {
                    ok2 = 0;
                    break;
                }
            }
            if(ok1 && ok2)
                newMatrix[i][j] = 1;
        }
    }
    cout << "YES\n";
    for(ll i=0; i<m; ++i)
    {
        for(ll j=0; j<n; ++j)
        {
            cout << newMatrix[i][j] << ' ';
        }
        cout.put('\n');
    }
    return 0;
}
```

4.12 Codeforces Round 277.5 Div 2

Результаты

Задачи

№	Название				
A	SwapSort	стандартный ввод/вывод 1 с, 256 МБ			x3847
B	Бал в БерлГУ	стандартный ввод/вывод 1 с, 256 МБ			x4322
C	Даны длина и сумма цифр...	стандартный ввод/вывод 1 с, 256 МБ			x4116
D	Невыносимая запутанность бытия	стандартный ввод/вывод 1 с, 256 МБ			x1822
E	Поход	стандартный ввод/вывод 1 с, 256 МБ			x381
F	Особые матрицы	стандартный ввод/вывод 1 с, 256 МБ			x771

Ссылка на контест: http://codeforces.com/contest/489

Задача A - SwapSort

В этой задаче вам требуется отсортировать массив из n чисел, используя не более чем n обменов. Для заданного массива найдите последовательность попарных обменов элементов, после которых массив становится отсортированным по неубыванию. Обмены производятся последовательно, один за одним.

Обратите внимание, что в задаче не требуется минимизировать число обменов — требуется найти любой способ длины не более n.

Входные данные

В первой строке входных данных содержится целое число n ($1 \le n \le 3000$) — количество элементов массива. Вторая строка содержит элементы массива $a_0, a_1, \ldots, a_{n-1}$ ($-10^9 \le a_i \le 10^9$), где a_i — это i-й элемент массива. Элементы пронумерованы от 0 до $n - 1$ слева направо. Среди элементов массива могут быть равные.

Выходные данные

В первую строку выведите k ($0 \le k \le n$) — количество обменов. Следующие k строк должны содержать описания k искомых обменов по одному в строке. Каждый обмен выводится в виде пары целых чисел i, j ($0 \le i, j \le n - 1$), которая обозначает обмен элементов a_i и a_j. Индексы в парах можно выводить в любом порядке. Обмены производятся в порядке их вывода сверху вниз. Допускается вывод $i = j$ и обмен пары элементов несколько раз.

Если ответов несколько, выведите любой из них. Гарантируется, что хотя бы один ответ существует.

Примеры тестов

входные данные

```
5
5 2 5 1 4
```

выходные данные

```
2
0 3
4 2
```

входные данные

```
6
10 20 20 40 60 60
```

выходные данные

```
0
```

входные данные

```
2
101 100
```

выходные данные

```
1
0 1
```

Алгоритм

На каждом шаге будем находить минимальный элемент и обменивать его с элементом под текущим индексом. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
#define ll long long
#define ull unsigned long long

int main()
{
    ll n;
    cin >> n;
    ll m[3000];
    for(int i=0; i<n; ++i)
        cin >> m[i];
    cout << n << '\n';
    for(int i=0; i<n; ++i)
    {
        ll min = 1000000001, minIndex = -1;
        for(int j=i; j<n; ++j)
        {
            if(m[j] < min)
            {
                min = m[j];
                minIndex = j;
            }
        }
        swap(m[i], m[minIndex]);
        cout << i << ' ' << minIndex << '\n';
    }
    return 0;
}
```

Задача B - Бал в БерлГУ

По случаю 100500-летия Берляндского государственного университета совсем скоро состоится бал! Уже n юношей и m девушек во всю репетируют вальс, менуэт, полонез и кадриль.

Известно, что на бал будут приглашены несколько пар юноша-девушка, причем уровень умений танцевать партнеров в каждой паре должен отличаться не более чем на единицу.

Для каждого юноши известен уровень его умения танцевать. Аналогично, для каждой девушки известен уровень ее умения танцевать. Напишите программу, которая определит наибольшее количество пар, которое можно образовать из n юношей и m девушек.

Входные данные

В первой строке записано целое число n ($1 \le n \le 100$) — количество юношей. Вторая строка содержит последовательность $a_1, a_2, ..., a_n$ ($1 \le a_i \le 100$), где a_i — умение танцевать i-го юноши.

Аналогично, третья строка содержит целое m ($1 \le m \le 100$) – количество девушек. В четвертой строке содержится последовательность $b_1, b_2, ..., b_m$ ($1 \le b_j \le 100$), где b_j — умение танцевать j-й девушки.

Выходные данные

Выведите единственное число — искомое максимальное возможное количество пар.

Примеры тестов

входные данные

```
4
1 4 6 2
5
5 1 5 7 9
```

выходные данные

```
3
```

входные данные

```
4
1 2 3 4
4
10 11 12 13
```

выходные данные

```
0
```

входные данные

```
5
1 1 1 1 1
3
1 2 3
```

выходные данные

```
2
```

Алгоритм

Отсортируем парней и девушек. Будем брать парня и девушку, если между ними разница не больше единицы, если же пара не получается, то переходим к другому парню или девушке (в зависимости от того, у кого индекс больше). Сложность после сортировки $O(n+m)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <map>
#include <cmath>
#include <algorithm>
#include <iomanip>
using namespace std;
#define ll long long
#define ull unsigned long long

int main()
{
    ll n, m;
    cin >> n;
    vector<ll> boys(n);
    for(ll i=0; i<n; ++i)
        cin >> boys[i];
    sort(boys.begin(), boys.end());
    cin >> m;
    vector<ll> girls(m);
    for(ll i=0; i<m; ++i)
        cin >> girls[i];
    sort(girls.begin(), girls.end());
    ll count = 0;
    for(ll i=0, j=0; i<n && j<m;)
    {
        if(boys[i] >= girls[j]-1 && boys[i] <= girls[j]+1)
        {
            ++i;
            ++j;
            ++count;
        }
        else if(boys[i] < girls[j])
            ++i;
        else
            ++j;
    }
    cout << count;
    return 0;
}
```

Задача C - Даны длина и сумма цифр...

Вам задано положительное целое число m и неотрицательное целое число s. Ваша задача найти наименьшее и наибольшее из чисел, которые имеют длину m и сумму цифр s. Искомые числа должны быть неотрицательными целыми, записанными в десятичной системе счисления без ведущих нулей.

Входные данные

В единственной строке входных данных записана пара целых чисел m, s ($1 \le m \le 100$, $0 \le s \le 900$) — длина и сумма цифр искомых чисел.

Выходные данные

В выходные данные выведите пару искомых неотрицательных целых чисел — сначала минимальное из возможных, потом — максимальное. Если ни одного числа, удовлетворяющего условию, не существует, то выведите пару чисел «`-1 -1`» (без кавычек).

Примеры тестов

входные данные
2 15
выходные данные
69 96

входные данные
3 0
выходные данные
-1 -1

Алгоритм

Будем собирать число из девяток, если же нужно больше цифр, то раскладываем девятки на слагаемые, если это невозможно, то выводим $1 - 1$". Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
#define ll long long
#define ull unsigned long long

int main()
{
    ll m, s;
    cin >> m >> s;
    if(m>1 && !s)
    {
        cout << "-1 -1";
        return 0;
    }
    else if(m == 1 && !s)
    {
        cout << "0 0";
        return 0;
    }
    char v[101];
    ll i = 0;
    while(s>0)
    {
        if(s >= 9)
```

```
            v[i++] = 9;
        else
            v[i++] = s;
        s -= 9;
    }
    ll n = i-1;
    if(i > m)
    {
        cout << "-1 -1";
        return 0;
    }
    for(; i<m; ++i)
    {
        v[i] = 0;
    }
    --v[n];
    ++v[m-1];
    for(ll i=m-1; i>=0; --i)
        cout << (int)v[i];
    cout.put(' ');
    ++v[n];
    --v[m-1];
    for(ll i=0; i<m; ++i)
        cout << (int)v[i];
    return 0;
}
```

Задача D - Невыносимая запутанность бытия

Томаш постоянно плутает и теряется, гуляя по улицам столицы Берляндии. Еще бы, ведь он привык, что в его родном городе для любой пары перекрестков существует ровно один способ пройти от одного к другому. В столице Берляндии это совсем не так!

Томаш подметил, что даже простые случаи неоднозначности перемещения его путают. Так, четверку различных перекрестков a, b, c и d таких, что существует два пути из a в c — один через b, а другой через d, он называет «чёртов ромб». Обратите внимание, пары перекрёстков (a, b), (b, c), (a, d), (d, c) должны быть непосредственно соединены дорогами. Схематично «чёртов ромб» изображен на рисунке ниже:

Наличие других дорог между любыми из перекрестков не делают ромб более привлекательным для Томаша, поэтому четверка перекрестков остается для него «чёртовым ромбом».

Принимая во внимание, что в столице Берляндии n перекрестков и m дорог, все из которых являются односторонними и известны наперед, найдите количество «чёртовых ромбов» в городе.

При сравнении ромбов порядок перекрестков b и d значения не имеет.

Входные данные

В первой строке входных данных записана пара целых чисел n, m ($1 \le n \le 3000$, $0 \le m \le 30000$) — количество перекрестков и дорог соответственно. Далее в m строках перечислены дороги по одной в строке. Каждая из дорог задана парой целых чисел a_i, b_i ($1 \le a_i, b_i \le n; a_i \ne b_i$) — номером перекрестка откуда она ведет и номером перекрестка куда она ведет. Между парой перекрестков существует не более одной дороги в каждом из двух направлений.

Не гарантируется, что из любого перекрестка можно добрать до любого другого.

Выходные данные

Выведите искомое количество «чёртовых ромбов».

Примеры тестов

входные данные

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

выходные данные

```
1
```

входные данные

```
4 12
1 2
1 3
1 4
2 1
2 3
2 4
3 1
3 2
3 4
4 1
4 2
4 3
```

выходные данные

```
12
```

Алгоритм

Из каждой вершины пробуем попасть в другую через промежуточные вершины, подсчитываем удачные попытки. Сложность $O(n^2 * log(n))$.

Исходный код

```
#include <iostream>
#include <vector>
#include <map>
#include <cmath>
#include <algorithm>
#include <iomanip>
#include <list>
#include <iterator>
using namespace std;
#define ll long long
#define ull unsigned long long

int main()
{
    ll n, m;
    cin >> n >> m;
    vector< pair< list<int>, list<int> > > l(n+1);
    vector< vector<bool> > matrix(n+1, vector<bool>(n+1, 0));
    for(ll i=0; i<m; ++i)
    {
        int a, b;
        cin >> a >> b;
        l[b].first.insert(l[b].first.end(), a);
        l[a].second.insert(l[a].second.end(), b);
        matrix[a][b] = 1;
    }
    ll count = 0;
    for(int i=1; i<=n; ++i)
    {
        for(int j=i+1; j<=n; ++j)
        {
            list<int>::iterator it = l[i].first.begin();
            ll a = 0, b = 0, c = 0;
            for(;it!=l[i].first.end(); ++it)
                if(*it!=j && matrix[*it][j])
                {
                    ++a;
                    if(matrix[j][*it] && matrix[i][*it])
                        ++c;
                }
            it = l[i].second.begin();
            for(;it!=l[i].second.end(); ++it)
                if(*it!=j && matrix[j][*it])
                    ++b;
            count += a*b-c;
        }
    }
    cout << count;
    return 0;
}
```

4.13 VK Cup 2015 Квалификация 2

Результаты

Задачи

№	Название			
A	Поворот, отражение и масштабирование	стандартный ввод/вывод 2 с, 256 МБ		x1298
B	Усреднение обращений	стандартный ввод/вывод 4 с, 256 МБ		x598
C	В поисках имени	стандартный ввод/вывод 2 с, 256 МБ		x1012
D	Статистика обработки роликов	стандартный ввод/вывод 3 с, 256 МБ		x699

Ссылка на контест: http://codeforces.com/contest/523

Задача A - Поворот, отражение и масштабирование

Поликарп пишет прототип графического редактора. Он уже определил для себя, что основные преобразования изображений в его редакторе будут: поворот изображения на 90 градусов по часовой стрелке, отражение изображения по горизонтали (симметрия относительно вертикальной прямой, то есть правая часть изображения переходит в левую и наоборот) и его масштабирование. Он уверен, что существует большое количество преобразований, которые можно выразить через эти три.

Недавно он завершил реализацию всех трех преобразований для монохромных изображений. Для тестирования этой функциональности он просит вас написать программу, которая последовательно выполнит с монохромным изображением три действия: сначала повернет его на 90 градусов по часовой стрелке, затем отразит его по горизонтали и, наконец, отмасштабирует вдвое (увеличит все линейные размеры в два раза).

Реализуйте эту функциональность, чтобы помочь Поликарпу протестировать его редактор.

Входные данные

В первой строке записаны два целых числа w и h ($1 \le w, h \le 100$) — ширина и высота изображения в пикселях. Сама картинка задана в h строках по w символов в каждой из них — каждый символ кодирует цвет соответствующего пикселя изображения. В качестве символов строк используются только символы «.» и «*», так как изображение монохромное.

Выходные данные

Выведите $2w$ строк по $2h$ символов в каждой — результат последовательного применения трех описанных выше преобразований.

Примеры тестов

входные данные

```
3 2
.*.
.*.
```

выходные данные

```
....
....
****
****
....
....
```

входные данные

```
9 20
**.......
****.....
******...
*******..
..******.
....****.
......***
*.....***
*********
*********
*********
*********
....**...
...****..
..******.
.********
****..***
***...***
**.....**
*.......*
```

выходные данные

```
********......**********........********
********......**********........********
********........********......********..
********........********.....********..
..********......********....********....
..********.....********...********....
..********.....********..********......
..********.....********.********.......
....********....****************........
....********....****************........
....********....****************........
....********....****************........
......*****************..**********....
......*****************..**********....
........****************....**********..
........****************....**********..
............************.....**********
............************.....**********
```

Алгоритм

Считываем картинку и выводим её просто поменяв индексы местами и индекс i поделив на два. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <cmath>
#include <algorithm>
#include <iomanip>
#include <list>
#include <iterator>
#include <climits>
#include <stack>
#include <cstring>
#include <cctype>
#include <queue>
#include <set>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}

int main()
{
    ll w, h;
    cin >> w >> h;
    char map[101][101];
    for (int i=0; i<h; ++i) {
        cin >> map[i];
    }
    for (int i=0; i<w*2; ++i) {
        for (int j=0; j<h; ++j) {
            cout.put(map[j][i/2]);
            cout.put(map[j][i/2]);
        }
        cout.put('\n');
    }
    return 0;
}
```

Задача C - В поисках имени

Марсианского мальчика зовут s — это имя ему совсем недавно дали родители на совершеннолетие. Теперь ему всюду нравится искать свое имя. Если он видит, что из какой-то строки может быть получено его имя путем удаления нуля или более букв (при этом, оставшиеся буквы остаются в том же порядке), то он очень радуется. Например, если s=«aba», то при виде строк «baobab», «aabbaa», «helloabahello» он очень радуется, а при виде строк «aab», «baaa» и «helloabhello» — нет.

Однако вдвое больше чем радоваться один раз, он любит радоваться два раза! Поэтому, получив в подарок строку t, он хочет разрезать ее на две части (левую и правую) так, чтобы каждая из них его очень радовала.

Помогите s определить количество различных способов разрезать заданную строку t на две части искомым образом.

Входные данные
В первой строке записана строка s, которая состоит из строчных букв латинского алфавита. Длина строки s — от 1 до 1000 букв.

Во второй строке записана строка t, которая тоже состоит из строчных букв латинского алфавита. Длина строки t — от 1 до 10^6 букв.

Выходные данные
Выведите искомое количество способов разрезать строку t на две части так, что каждая из них очень радует s.

Примеры тестов

входные данные
aba baobababbah
выходные данные
2

входные данные
mars sunvenusearthmarsjupitersaturnuranusneptune
выходные данные
0

Алгоритм

Находим подстроку слева и справа и считаем расстояние между ними. Сложность $O(n)$.

Исходный код

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <cmath>
#include <algorithm>
#include <iomanip>
#include <list >
#include <iterator>
#include <climits>
#include <stack>
#include <cstring>
#include <cctype>
#include <queue>
#include <set>
using namespace std;
#define ll long long
#define ull unsigned long long
#define eps 0.00001
ll min(ll a, ll b){return (a<b?a:b);}
ll max(ll a, ll b){return (a>b?a:b);}
```

```

int main()
{
    char s[1001], t[1000001];
    cin >> s >> t;
    ll s_len = strlen(s);
    ll t_len = strlen(t);
    ll sLCursor = 0, sRCursor = s_len - 1, l = -1, r = -1;
    for (ll i=0; i<t_len; ++i) {
        if (s[sLCursor] == t[i]) {
            ++sLCursor;
            if (sLCursor == s_len) {
                l = i;
                break;
            }
        }
    }
    for (ll i=t_len-1; i>=0; --i) {
        if (s[sRCursor] == t[i]) {
            --sRCursor;
            if (sRCursor == -1) {
                r = i;
                break;
            }
        }
    }
    if (l == -1 || r == -1 || l >= r) {
        cout << 0;
    }
    else {
        cout << r-l;
    }
    return 0;
}
```

4.14 VK Cup 2015 - Раунд 1

Результаты

Задачи

№	Название				
A	Возможно, вы знаете этих людей?	стандартный ввод/вывод 2 с, 256 МБ			x740
B	Фото на память - 2 (round version)	стандартный ввод/вывод 2 с, 256 МБ			x695
C	Искусство обращения с бакноматом	стандартный ввод/вывод 2 с, 256 МБ			x526
D	Социальная сеть	стандартный ввод/вывод 2 с, 256 МБ			x338
E	Ладьи и прямоугольники	стандартный ввод/вывод 2 с, 256 МБ			x242
F	И снова правильная скобочная последовательность	стандартный ввод/вывод 5 с, 256 МБ			x88

Ссылка на контест: http://codeforces.com/contest/524

Задача A - Возможно, вы знаете этих людей?

Основой любой социальной сети является отношение дружбы между двумя пользователями в том или ином смысле. В одной известной социальной сети дружба симметрична, то есть если a является другом b, то b также является другом a.

В этой же сети есть функция, которая демонстрирует множество людей, имеющих высокую вероятность быть знакомыми для пользователя. Эта функция работает следующим образом. Зафиксируем пользователя x. Пусть некоторый другой человек y, не являющийся другом x на текущий момент, является другом не менее, чем для $k\%$ друзей x. Тогда он является предполагаемым другом для x.

У каждого человека в социальной сети есть свой уникальный идентификатор — это целое число от 1 до 10^9. Вам дан список пар пользователей, являющихся друзьями. Определите для каждого упомянутого пользователя множество его предполагаемых друзей.

Входные данные

В первой строке следуют два целых числа m и k ($1 \le m \le 100$, $0 \le k \le 100$) — количество пар друзей и необходимый процент общих друзей для того, чтобы считаться предполагаемым другом.

В последующих m строках записано по два числа a_i, b_i ($1 \le a_i, b_i \le 10^9$, $a_i \ne b_i$), обозначающих идентификаторы пользователей, являющихся друзьями.

Гарантируется, что каждая пара людей фигурирует в списке не более одного раза.

Выходные данные

Для всех упомянутых людей в порядке возрастания id выведите информацию о предполагаемых друзьях. Информация должна иметь вид "id: k id_1 id_2 ... id_k", где id — это id самого человека, k — количество его предполагаемых друзей, а id_1, id_2, ..., id_k — идентификаторы его предполагаемых друзей в возрастающем порядке.

Примеры тестов

входные данные

```
5 51
10 23
23 42
39 42
10 39
39 58
```

выходные данные

```
10: 1 42
23: 1 39
39: 1 23
42: 1 10
58: 2 10 42
```

входные данные

```
5 100
1 2
1 3
1 4
2 3
2 4
```

выходные данные

```
1: 0
2: 0
3: 1 4
4: 1 3
```

Алгоритм

Создаём граф друзей и затем, для каждого друга находим возможных друзей, если процент общих друзей больше заданного. Сложность $O(n^2)$.

Исходный код

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <cmath>
#include <algorithm>
#include <iomanip>
#include <list>
#include <iterator>
#include <climits>
#include <stack>
#include <cstring>
#include <cctype>
#include <queue>
#include <set>
using namespace std;
#define ll long long

map<ll, pair<ll, set<ll>>> p;

void add_friend(ll a, ll b) {
    map<ll, pair<ll, set<ll>>>::iterator findIter = p.find(a);
    if (findIter == p.end()) {
        set<ll> t;
        t.insert(b);
        p[a] = make_pair(a, t);
    }
    else {
        p[a].second.insert(b);
    }
}

int main()
{
    ll m;
    double k;
    cin >> m >> k;
    for (ll i=0; i<m; ++i) {
        ll a, b;
        cin >> a >> b;
        add_friend(a, b);
        add_friend(b, a);
    }
    map<ll, pair<ll, set<ll>>>::iterator iter;
    for ( iter = p.begin(); iter != p.end(); ++iter) {
        ll a = iter->first;
        cout << a << ": ";
        map<ll, pair<ll, set<ll>>>::iterator iter2;
        list<ll> p_friends;
        for ( iter2 = p.begin(); iter2 != p.end(); ++iter2) {
            if (iter2->first == a) {
                continue;
            }
```

```
            set<ll>::iterator findIter = find(p[a].second.begin(), p[a].second.end(), iter2->first);
            if (findIter == p[a].second.end()) {
                ll c = iter2->first;
                set<ll>::iterator setIter = p[a].second.begin();
                ll o = 0;
                for ( ; setIter != p[a].second.end(); ++setIter) {
                    ll b = *setIter;
                    if (p[b].second.find(c) != p[b].second.end()) {
                        ++o;
                    }
                }
                ll kokoko = o*100/p[a].second.size();
                if (kokoko >= k) {
                    p_friends.push_back(c);
                }

            }
        }
        cout << p_friends.size() << ' ';
        list<ll>::iterator it = p_friends.begin();
        for ( ; it != p_friends.end(); ++it) {
            cout << *it << ' ';
        }
        cout << '\n';
    }
    return 0;
}
```

4.15 VK Cup 2015 - Уайлд-кард раунд 1

Результаты

Задачи

№	Название				
A	Квадратное уравнение	стандартный ввод/вывод 2 с, 256 МБ			x298
B	Строка наизнанку	стандартный ввод/вывод 2 с, 256 МБ			x240
C	Диофантово уравнение	стандартный ввод/вывод 2 с, 256 МБ			x190
D	Разность множеств	стандартный ввод/вывод 2 с, 256 МБ			x195
E	Сумма и произведение	стандартный ввод/вывод 2 с, 256 МБ			x118
F	Прыгающие лягушки	стандартный ввод/вывод 2 с, 256 МБ			x29
G	Расстояние Левенштейна	стандартный ввод/вывод 2 с, 256 МБ			x29
H	Точки в треугольнике	стандартный ввод/вывод 2 с, 256 МБ			x48
I	Разные переменные	стандартный ввод/вывод 2 с, 256 МБ			x13

Ссылка на контест: http://codeforces.com/contest/530

Задача A - Квадратное уравнение

Вам дано квадратное уравнение с целыми коэффициентами $A * X^2 + B * X + C = 0$. Гарантируется, что $A \neq 0$ и уравнение имеет хотя бы один действительный корень. Найдите корни уравнения.

Входные данные

Единственная строка входных данных содержит целые числа A, B и C (- $1000 \leq A, B, C \leq 1000, A \neq 0$).

Выходные данные

Выведите корни уравнения в порядке их возрастания. Если уравнение имеет один корень кратности 2, выведите его только один раз. Корень считается правильным, если его абсолютная или относительная погрешность не превосходит 10^{-4}.

Примеры тестов

входные данные
1 -2 1
выходные данные
1

Алгоритм

Сложность этой задачи состоит в том, что написать ее нужно на эзотерическом языке *Picat*.

Исходный код

```
main =>
    A = read_int(),
    B = read_int(),
    C = read_int(),
    D = B*B-4*A*C,
        if (D == 0) then
            writef("%w%n",-B/(2*A))
        elseif ((-B+sqrt(D))/(2*A) < (-B-sqrt(D))/(2*A)) then
                writef("%w %w%n", (-B+sqrt(D))/(2*A), (-B-sqrt(D))/(2*A))
        else
                writef("%w %w%n", (-B-sqrt(D))/(2*A), (-B+sqrt(D))/(2*A))
        end.
```

Задача B - Строка наизнанку

Вам дана строка S четной длины $s_1..s_{2n}$. Выполните следующие преобразования:

- разбейте ее на две половины: $s_1..s_n$ и $s_{n+1}..s_{2n}$
- разверните каждую из них: $s_n..s_1$ и $s_{2n}..s_{n+1}$
- сконкатенируйте полученные строки: $s_n..s_1 s_{2n}..s_{n+1}$

Выведите строку, полученную в результате этих преобразований.

Входные данные
В единственной строке входных данных записана строка, состояща из строчных латинских символов. В строке будет от 2 до 20 символов, включительно. Строка содержит чётное количество символов.

Выходные данные
Выведите строку, полученную в результате описанных преобразований.

Примеры тестов

входные данные
codeforces
выходные данные
fedocsecro

входные данные
qwertyasdfgh
выходные данные
ytrewqhgfdsa

Алгоритм

Сложность этой задачи состоит в том, что написать ее нужно на эзотерическом языке *Picat*.

Исходный код

```
main =>
    S = read_line(),
    L = S.length,
    X1 = to_integer(L/2),
    X2 = X1+1,
    S1 = reverse(slice(S, 1, X1)),
    S2 = reverse(slice(S, X2, L)),
    printf("%s%s", S1,S2).
```

4.16 Vekua Cup 2015 Личный этап

Так как соревнование проводилось в центре 1С, исходные коды программ не доступны.

Результаты

155.	[9246-29] Антон Якименко (МАИ)	-	-	-	-	-	-	-	+2 2:04	1	164	66%		0.09

Ссылка на контест: http://vekua.snarknews.info