

## Задача А. Минёр

Имя входного файла: mines.in  
Имя выходного файла: mines.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Каждый сапёр знает, что вероятность подорваться на mine очень сильно зависит от того, насколько профессионально они расставлены. Широко известная игра “Сапёр” проходит на прямоугольном поле из  $w \times h$  клеток. В каждой клетке либо стоит мина, либо записано количество мин, располагающихся в восьми соседних клетках (число от 0 до 8). В этой задаче Вам необходимо узнать, какое минимальное количество мин потребуется, чтобы не оказалось ни одной клетки, в которой или рядом с которой не будет мины.

### Формат входных данных

В первой строке входного файла находятся два целых числа  $w$  и  $h$  ( $1 \leq w, h \leq 1000$ ), разделенные пробелом — ширина и высота поля, соответственно.





### Формат выходных данных

В единственной строке выходного файла выведите минимальное количество мин, достаточное для того, чтобы в каждой клетке игрового поля либо стояла мина, либо было записано число больше нуля.

### Примеры

mines.in	mines.out
3 3	1
5 5	4

Второй пример (поле  $5 \times 5$ ) проиллюстрирован на следующем рисунке:

1	1	1	1	
1		1	1	1
1	1	2	1	1
1	1	2		1
1		2	1	1

## Задача В. Всемирное собрание

Имя входного файла: meeting.in  
Имя выходного файла: meeting.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

В одном плоском мире, на одной плоской планете в форме круга живут плоские люди. Недавно власти решили провести Всемирное собрание, на которое пригласили всех жителей планеты. Они хотят выбрать такое место на поверхности планеты, чтобы суммарное расстояние, пройденное всеми приглашенными жителями до него, было минимальным. Помогите им! Заметьте, что поверхность планеты представляет собой окружность с центром в начале координат и жители могут перемещаться только по поверхности.

### Формат входных данных

В первой строке входного файла находится натуральное число  $n$  ( $1 \leq n \leq 20000$ ) — количество жителей планеты. Во второй строке через пробел заданы  $n$  десятичных

вещественных чисел в интервале  $[0..360)$ .  $i$ -ое число соответствует полярному углу дома  $i$ -го человека (в градусах). Углы даны в порядке возрастания, среди них нет совпадающих. Полярным углом точки  $A$  называется угол между осью абсцисс и отрезком  $OA$ , где  $O$  — точка начала координат. Полярный угол отсчитывается против часовой стрелки и может принимать значения в интервале  $[0..360)^\circ$ .

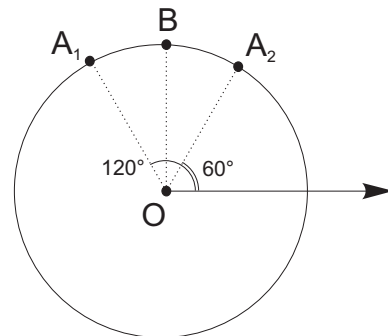
### Формат выходных данных

В единственной строке выходного файла выведите одно десятичное вещественное число, лежащее в пределах  $[0..360)$  — полярный угол предполагаемого места проведения собрания (в градусах). Суммарное пройденное расстояние должно отличаться от лучшего не более чем на  $10^{-6}$ . Если возможно несколько вариантов ответа, выведите любой.

### Пример

meeting.in	meeting.out
2 60 120	90

Пример проиллюстрирован на следующем рисунке. Точками  $A_1$  и  $A_2$  обозначены жители. Точка  $B$  соответствует месту Всемирного собрания. Точкой  $O$  обозначен центр планеты.



## Задача С. Партия в шахматы

Имя входного файла: chess.in  
Имя выходного файла: chess.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

*Шахматы — настольная логическая игра, сочетающая в себе элементы искусства, науки и спорта. Считается одной из древнейших игр на Земле...*

Википедия, свободная энциклопедия  
(<http://ru.wikipedia.org>).

Игра происходит на доске, поделенной на равные квадратные клетки (размер доски  $8 \times 8$  клеток). Традиционно клетки нумеруются по горизонтали латинскими буквами от  $a$  до  $h$ , по вертикали цифрами от 1 до 8 (каждая клетка имеет соответственное обозначение, например,  $c4$ ). Клетки раскрашены в чёрный и белый цвета, так что соседние по вертикали и горизонтали клетки раскрашены в разные цвета, а клетка  $a8$  — белая.

Играют два игрока. Каждый имеет набор фигур, в который входят: один король, один ферзь, два слона, два коня, две ладьи, восемь пешек.

Один игрок играет белыми фигурами, другой — чёрными. В начале игры фигуры размещаются на фиксированных позициях, занимая ровно две строки (1 и 2 белые, 7 и 8 черные). Порядок начального размещения фигур на строках

1 и 8 (от а к h): ладья, конь, слон, ферзь, король, слон, конь, ладья. Горизонтالي 2 и 7 занимают пешки.

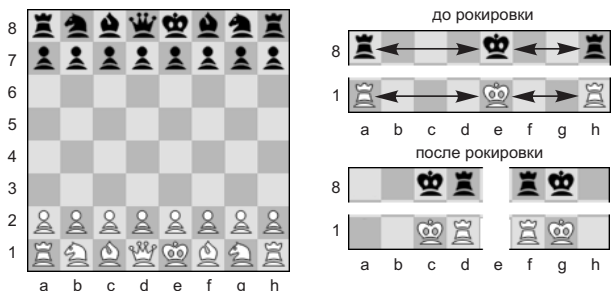
Игроки делают ходы по очереди, сначала ходят белые, затем черные, и так далее. В свой ход игроки перемещают ровно одну фигуру по правилам хода этой фигуры.

Естественно, что партию, когда она играет, имеет смысл записывать (например, чтобы иметь возможность воспроизвести ее позже). Для этого разработана специальная нотация. Ваша задача — восстановить положение фигур в конце партии, если известна запись всех её ходов.

В данной задаче Вам вовсе не обязательно знать, как ходит та или иная фигура. Но стоит сосредоточить внимание на двух специальных ситуациях.

Первая ситуация, требующая специального рассмотрения в данной задаче, называется *рокировкой*. Рокировкой называется одновременное перемещение короля и одной из ладей того же цвета по крайней горизонтали (считается одним ходом короля) и она выполняется следующим образом: король перемещается с его исходного поля на два поля по направлению к ладье, затем ладья переставляется через короля на последнее поле, которое только что король пересек. Бывает два вида рокировок: длинная и короткая.

На следующем рисунке показаны вид поля с начальной расстановкой (слева), а также два вида рокировок (справа).



Вторая ситуация возникает тогда, когда пешка, пройдя поле, достигает крайней горизонтали. Для белой пешки это восьмая горизонталь, а для черной — первая. В конце такого хода пешка может превратиться в любую фигуру такого же цвета, но не в пешку и не короля.

### Формат входных данных

В первой строке входного файла записано натуральное число  $n$  ( $n \geq 1$ ) — количество ходов в записи партии. Последующие  $n$  строк имеют формат  $Fx_1y_1 - x_2y_2$ .

В этой строке символ  $F$  обозначает фигуру, которая делает ход (К — король, Q — ферзь, R — ладья, В — слон, N — конь, P — пешка). При этом белые фигуры обозначаются заглавными буквами, а черные — строчными. К примеру, К — это белый король, а r — черная ладья. Пары символов  $x_1y_1$  и  $x_2y_2$  — это координаты фигуры до хода и после того, как ход был совершен, соответственно. В этой записи  $x_1$  и  $x_2$  — символы столбцов, а  $y_1$  и  $y_2$  — номера строк.

Короткая рокировка обозначена 0-0, длинная 0-0-0, где 0 — ноль, при этом черная и белая рокировка не отличаются по записи. Если во время хода пешка добралась до противоположной горизонтали, то добавляется символ  $F'$  в конец записи хода.  $F'$  — это символ фигуры, в которую превращается пешка (по описанной выше спецификации).

Некоторые примеры: запись **Pe2-e4** означает, что белая пешка сделала ход с e2 на e4, запись **re2-e1Q** означает, что черная пешка сделала ход с e2 на e1 и превратилась в ферзя.

Считайте, что все ходы сделаны по правилам шахмат, более того, во входном файле никогда не будет ситуации, когда пешка берется на проходе.

### Формат выходных данных

В выходной файл выведете восемь строк по восемь символов в каждой. Строки описывают горизонтали шахматного поля, в порядке с восьмой по первую. Столбцы описывают вертикали шахматного поля, в порядке с а по h. Символы соответствуют фигурам, находящимся в соответствующих позициях, и их надо выводить в том же формате, который используется во входном файле. Для пустых клеток поля необходимо выводить символ . (точка).

### Примеры

	chess.in	chess.out
	<pre>9 Pe2-e4 pe7-e5 Pd2-d3 ng8-f6 Bc1-g5 nf6-e4 Bg5-d8 ke8-d8 Qd1-h5</pre>	<pre>rnbk.b.r pppp.ppp ..... ...p..Q ...n... ...P.... PPP..PPP RN..KBNR</pre>
	<pre>13 Pe2-e4 pe7-e5 Pd2-d3 ng8-f6 Bc1-g5 nf6-e4 Bg5-d8 ke8-d8 Qd1-h5 pa7-a6 Nb1-a3 pa6-a5 0-0-0</pre>	<pre>rnbk.b.r .ppp.ppp ..... p...p..Q ...n... N..P.... PPP..PPP ..KR.BNR</pre>

### Задача D. Пинбол в треугольнике Паскаля

Имя входного файла: pinball.in  
Имя выходного файла: pinball.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Пинбол в треугольнике Паскаля — игра для одного человека с незаурядными арифметическими способностями. Поле для игры выглядит следующим образом:

0-я строка:										1
1-я строка:									1	1
2-я строка:								1	2	1
3-я строка:							1	3	3	1
4-я строка:						1	4	6	4	1
5-я строка:				1	5	10	10	5	1	
6-я строка:		1	6	15	20	15	6	1		

В каждом ряду слева и справа стоят единицы, а всякое внутреннее число получается как сумма двух чисел, стоящих над ним.

В начале игры фишка ставится на верхнюю единицу. За ход игрок передвигает фишку на одну строчку вниз и либо на полстолбца влево, либо на полстолбца вправо. Например, с четверки игрок может походить либо на пятерку, либо на

десятку. Сделав этот ход, игрок получает количество очков, равное сумме цифр в том числе, на которое он походил. Для удобства будем считать, что за начальное положение фишки (за самую верхнюю единицу) игрок также получает бесплатное первое очко.

Задача игрока - дойти фишкой до  $n$ -й строки и заработать при этом максимальное количество очков. Точнее, это задача для Вашей программы.

### Формат входных данных

Во входном файле содержится целое неотрицательное число  $n$  — номер строки, до которой надо добраться ( $n \leq 30$ ).

### Формат выходных данных

В выходной файл выведите максимальное количество очков, которое можно заработать в этой игре.

### Примеры

pinball.in	pinball.out
2	4
6	22

### Примеры

spy.in	spy.out
1 1 0 2 0	YES
5 3 4 5 -1	NO
5 -3 -4 0 -10	YES

### Задача F. Квадрат

Имя входного файла: square.in  
Имя выходного файла: square.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Дано три прямоугольника. Необходимо определить, существует ли квадрат, который может быть составлен из заданных прямоугольников. Прямоугольники не должны накладываться друг на друга, при этом их можно сдвигать, не меняя их начальной ориентации.

### Формат входных данных

Во входном файле содержатся три строки, в каждой из которых содержится описание одного прямоугольника. Описание прямоугольника состоит из двух целых чисел  $w$  и  $h$ , записанных через пробел ( $0 < w, h \leq 10^6$ ) — ширины и высоты, соответственно.

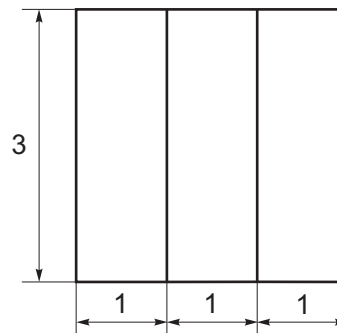
### Формат выходных данных

Выведите слово YES, если такой квадрат существует, иначе выведите NO.

### Примеры

square.in	square.out
1 3 1 3 1 3	YES
1 5 5 3 2 5	NO

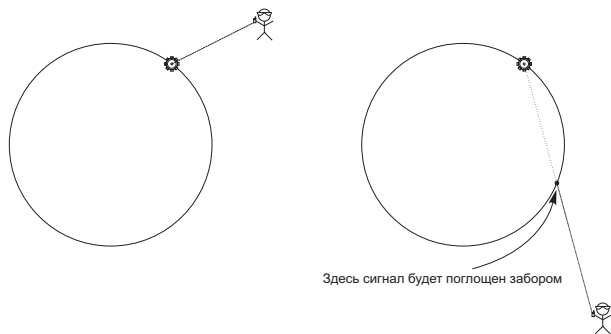
Первый пример проиллюстрирован на следующем рисунке.



### Задача E. Шпион

Имя входного файла: spy.in  
Имя выходного файла: spy.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Секретный объект обнесен забором, выполненным в форме окружности радиуса  $r$  с центром в точке  $(0, 0)$ . Шпион, пытающийся разведать точное местоположение секретного объекта, для преодоления забора установил бомбу точно на заборе в точке  $(a_x, a_y)$ . Бомба активизируется дистанционным взрывателем, сигнал от которого распространяется по прямой, причём качество его приёма не зависит от пройденного расстояния. Из секретных источников известно, что забор полностью поглощает проходящий сквозь него сигнал (на сигнал, проходящий по касательной, забор не влияет). Шпион, находясь вне области, окруженной забором, в точке  $(b_x, b_y)$ , решил взорвать бомбу. Помогите ему узнать, дойдет ли сигнал до бомбы или будет поглощен.



### Формат входных данных

В первой строке входного файла записано число  $r$  ( $r > 0$ ). Во второй строке входного файла содержатся два числа  $(a_x, a_y)$ . В третьей строке входного файла содержатся два числа  $(b_x, b_y)$ . Все числа во входном файле целые и не превышают по модулю  $10^3$ .

### Формат выходных данных

В единственной строке выходного файла выведите слово YES, если сигнал дойдет до бомбы, и NO в противном случае.

### Задача G. Два капитана

Имя входного файла: captains.in  
Имя выходного файла: captains.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

*Корабли без капитанов, капитан без корабля...*

Несколько приятелей решили поиграть в футбол. Известно, что не все они играют одинаково хорошо, у каждого есть сила, которую можно выразить некоторым числом (чем больше число, тем сильнее игрок). Поэтому было решено выделить двух капитанов, которые будут набирать людей в команды. Сначала решили, что капитаны будут по очереди выбирать в свою команду по одному человеку. При этом изначально капитаны тянут жребий, и тот, кто его выигрывает, выбирает игрока первым. Однако получается, что при этом команда выигравшего жребий капитана оказывается гораздо сильнее другой команды, так как в каждой паре выбранных игроков игрок первой команды будет сильнее (по крайней мере, не слабее) игрока второй команды. Поэтому было решено поступить более справедливо. Капитаны, как и раньше, по очереди выбирают игроков, но при выборе первой пары (и каждой нечетной) выбирать начинает выигравший жребий капитан, а при выборе второй (и каждой четной) — первым выбирает другой, проигравший жребий, капитан. Однако в таком случае может получиться, что при оптимальном выборе игроков капитанами команда проигравшего жребий капитана окажется сильнее команды выигравшего жребий капитана. Вам необходимо определить, что выгодно — выиграть или проиграть жребий, чтобы набрать более сильную команду, при условии, что оба капитана будут действовать оптимально, то есть выберут стратегию, которая позволит им набрать настолько сильную команду, насколько это возможно, независимо от выбора капитана команды соперника.

### Формат входных данных

В первой строке входного файла дано количество игроков  $n$  ( $0 < n \leq 100$ ,  $n$  — четное). Во второй строке перечислены  $n$  чисел  $a_i$  — силы игроков ( $a_i \geq 0$ ). Капитанов среди них нет. Силы капитанов равны нулю. Все числа во входном файле целые и их сумма не превышает  $10^9$ .

### Формат выходных данных

В первой строке выходного файла выведите, получится ли команда выигравшего жребий сильнее (YES — если получится, и NO — если не получится). Во второй строке выведите через пробел, какая сила команды получится у выигравшего жребий капитана, и какая сила команды получится у проигравшего жребий капитана. Сила команды — это сумма сил всех игроков этой команды.

### Примеры

captains.in	captains.out
4 10 6 7 1	NO 11 13
2 100 99	YES 100 99

### Задача H. Игра

Имя входного файла: `game.in`  
Имя выходного файла: `game.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Однажды два математика придумали математическую игру и решили в нее сыграть. Они попросили постороннего человека написать на разных листах бумаги два натуральных числа, причем одно должно быть больше другого ровно в 2 раза. Затем один математик взял один лист, а второй взял оставшийся. В чужие листы они не заглядывали и единственное, что знал каждый из них — на другом листе записано число либо в 2 раза большее, либо в 2 раза меньшее, чем у него. Затем началась игра. Игра заключается в том, что

игроки по очереди отвечают на вопрос, знают ли они, какое число записано на листе соперника, до тех пор, пока один из них не сможет назвать это число. В процессе игры игроки всегда учитывают всю информацию, которую им дает ответ соперника, и говорят только правду. Например, могло быть так: первому игроку достался лист с числом 1, а второму — с числом 2. Тогда игра развивалась бы следующим образом:

1 игрок: Я знаю, твое число — 2.

А вот другая ситуация: первому игроку достался лист с числом 8, а второму — с числом 16.

1 игрок: Я не знаю, какое число записано у тебя на листе.

2 игрок: Я не знаю, какое число записано у тебя на листе.

1 игрок: Я не знаю, какое число записано у тебя на листе.

2 игрок: Я не знаю, какое число записано у тебя на листе.

1 игрок: Я знаю, твое число — 16.

Ваша задача — определить, с какого хода игроки смогут угадать, какое число записано на листе у соперника. Если они так и не смогут догадаться — выведите 0.

### Формат входных данных

В первой строке входного файла через пробел записаны два целых числа  $a$  и  $b$  — числа первого и второго игроков соответственно. ( $0 < a, b \leq 10^6$ ).

### Формат выходных данных

В выходной файл выведите номер хода, на котором один из игроков сможет с уверенностью сказать, какое число записано на листе у соперника, либо 0, если оба игрока не смогут этого сделать при любом количестве ходов.

### Примеры

game.in	game.out
1 2	1
8 16	5

### Задача I. Футбольные парадоксы

Имя входного файла: `paradox.in`  
Имя выходного файла: `paradox.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

После того, как сборная Бразилии стала Чемпионом Мира по футболу, она проиграла сборной Италии, которая, в свою очередь, проиграла сборной Болгарии, а та — сборной Люксембурга. Так что же, получается, что Люксембург сильнее Бразилии? Такие парадоксы происходят довольно часто, и удивляться этому не стоит, так как победа в матче не обладает свойством транзитивности, то есть описанное выше не означает, что при встрече сборных Бразилии и Люксембурга обязательно выиграет Люксембург.

Это заинтересовало Петю, и он решил проанализировать результаты всех матчей, которые он знает, и выбрать из них самую длинную цепочку игр, обладающую следующим свойством — победитель любого матча этой цепочки (кроме последнего) должен проиграть в следующем матче этой цепочки. Заметьте, что хронологический порядок игр должен сохраниться, то есть следующий матч в цепочке должен быть сыгран позже предыдущего. Пете не важно, заканчивается ли цепочка игр той же командой, с которой начинается, или нет. Прежде всего, его волнует количество игр в ней.

### Формат входных данных

Во входном файле содержится отсортированная хронологически последовательность игр, то есть каждая следующая игра в этой последовательности была сыграна позже предыдущей. В первой строке входного файла записано целое число  $n$  — количество сыгранных игр ( $0 < n \leq 10000$ ). В каждой из следующих  $n$  строк содержится описание одной игры. Каждая игра описывается строкой из семи символов. Первые три символа — идентификатор выигравшей команды, четвертый символ — тире, символы с пятого по седьмой — идентификатор проигравшей команды. Идентификатор команды всегда является трехбуквенным и состоит только из заглавных латинских букв. Количество различных идентификаторов команд во входном файле не превышает 200.

### Формат выходных данных

В первую строку выходного файла выведите максимальное количество матчей в искомой цепочке. Во вторую строку выведите цепочку команд, начиная с команды, победившей в последнем матче цепочки, и заканчивая командой, проигравшей в первом. Если вариантов наиболее длинных цепочек несколько, выведите любой из них.

### Примеры

paradox.in	paradox.out
5 FRA-ITA GER-ITA ITA-GER ITA-LUX RUS-ITA	3 RUS-ITA-GER-ITA
3 BLR-UKR LAT-BLR UKR-LAT	3 UKR-LAT-BLR-UKR

**Пояснение:** оптимальная цепочка в первом примере состоит из матчей GER-ITA, ITA-GER, RUS-ITA, а во втором примере все матчи входят в цепочку.

### Задача J. Шахматные снайперы

Имя входного файла: `snipers.in`  
Имя выходного файла: `snipers.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

В связи с обострением ситуации на первом и третьем фронтах шахматной доски  $n \times m$  клеток, главнокомандующим белой армии было решено ввести новую боевую единицу — шахматного снайпера. Шахматный снайпер — фигура, которая бьет в каком-то определенном направлении (вперед, назад, влево или вправо), при этом направление фиксируется уже при установке фигуры на клетку поля. Однако перед непосредственным вступлением снайперов в бой необходимо выяснить: какое максимальное количество таких фигур можно разместить на пустом шахматном поле размером  $n \times m$  клеток так, чтобы они не били друг друга.

### Формат входных данных

В первой строке входного файла записаны два целых числа  $n$  и  $m$  ( $1 \leq n, m \leq 1000$ ).

### Формат выходных данных

В первую строку выходного файла выведите одно число — максимальное количество не бьющих друг друга снайперов.

### Пример

snipers.in	snipers.out
2 2	4

### Задача K. Крестики-нолики

Имя входного файла: `xo.in`  
Имя выходного файла: `xo.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Крестики-нолики на бесконечном поле — это игра для двух человек. Поле представляет собой бесконечный клетчатый лист бумаги. Игроки ходят по очереди. За один ход игрок ставит в какую-нибудь свободную клетку игрового поля свой символ. Символ первого игрока — крестик, а символ второго игрока — нолик.

Если после очередного хода игрока на поле появляются **пять** стоящих в ряд (по горизонтали, вертикали или диагонали) символов этого игрока, то он объявляется победителем партии, и игра заканчивается.

Однажды одаренный школьник Антон обнаружил у себя в тетради некоторую картинку из крестиков и ноликов. Он не может вспомнить, что это такое: поле его поединка с другом Лёшей или просто произвольная картинка из крестов и нулей.

Напишите программу, которая по заданной картинке из крестиков и ноликов определяет, могла ли такая позиция возникнуть на поле в результате игры — либо законченной, либо незаконченной. Предполагается, что игроки всегда соблюдают все правила.

### Формат входных данных

Во входном файле находится картина из тетради Антона. Пустые клетки обозначаются символом ‘.’ (точка). Символы игроков обозначаются символами ‘X’ и ‘O’ (заглавные латинские буквы ‘икс’ и ‘о’).

Количество строчек во входном файле не превосходит 100. Количество символов в каждой строке также не превосходит 100. Пустых строчек в файле нет. Гарантируется, что во входном файле будет хотя бы один крестик или нолик.

### Формат выходных данных

В выходной файл выведите слово `CORRECT`, если данная позиция могла возникнуть в результате игры в крестики-нолики на бесконечном поле (в конце игры или в середине). В противном случае выведите слово `INCORRECT`.

### Примеры

xo.in	xo.out
.X ..X ..OXO ...X .....O	CORRECT
.... .XO.. .XO. .XO.. .XO. .XO.. ....	INCORRECT

### Задача L. Муравьи на кубической Земле

Имя входного файла: `ants.in`  
Имя выходного файла: `ants.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

В условиях данной задачи будем пользоваться

предположением, что Земля имеет форму куба, и каждая грань — квадрат  $m \times m$ , расчерченный на клетки размером  $1 \times 1$ .

В начальный момент времени  $n$  муравьев стоят на верхней грани этого куба. Каждый муравей направлен в одну из четырех сторон — на север, на юг, на запад или на восток.

В определенный момент муравьи начинают двигаться по прямой, каждый в своем направлении. Когда муравей доползает до ребра куба, он переползает через него и продолжает движение по следующей грани. При этом он все время движется перпендикулярно тому ребру, которое переполз.

Такое движение продолжается бесконечно долго. Выясните, сколько есть клеток на кубе, на которых ни разу во время этого процесса не побывает ни один муравей.

### Формат входных данных

В первой строчке входного файла содержатся два натуральных числа —  $n$  и  $m$  — количество муравьев на земле и длина стороны планеты ( $1 \leq n \leq 100000$ ;  $1 \leq m \leq 15000$ ).

В каждой из следующих  $n$  строчек находится описание начального положения очередного муравья. Сначала идут два натуральных числа  $x$  и  $y$  — координаты муравья на верхней грани, а затем символ, задающий направление муравья — 'N', 'S', 'W' или 'E'. Числа и символ разделяются ровно одним пробелом.

Оси координат и направления сторон света приведены на рисунке. Все координаты лежат в интервале от 1 до  $m$  включительно.

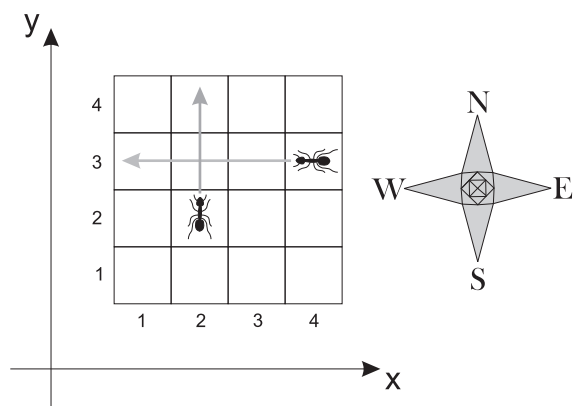
Несколько муравьев как в начальный момент времени, так и в любой другой, могут оказаться на одной клетке. Это никак не влияет на траектории их движения.

### Формат выходных данных

В выходной файл выведите одно число — количество клеток, никогда не посещаемых муравьями.

### Пример

ants.in	ants.out
2 4	66
2 2 N	
4 3 W	



Верхняя грань куба, вид сверху

стандартными, появилась новая задача, обобщающая все предыдущие. Дана матрица из нулей и единиц. Необходимо найти наилучшую “расческу”, которую можно построить на нулевых ячейках этой матрицы (наилучшей называется расческа, которая занимает наибольшее количество ячеек данной матрицы). Звенком расчески называется непустая прямоугольная область из нулей в матрице, то есть некоторая подматрица ненулевой площади, состоящая только из нулей.  $k$ -расческой называется фигура, получаемая соединением по вертикали  $k$  ( $0 \leq k \leq m$ ) звеньев. У всех звеньев должна совпадать  $x$ -координата левой стороны, соседние звенья должны иметь разную ширину, а все звенья вместе должны образовывать цельную фигуру, то есть должны быть связаны. При этом 0-расческа существует всегда и имеет размер 0.

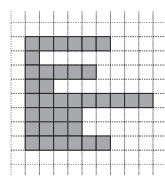


рис. 1

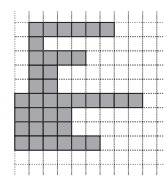


рис. 2

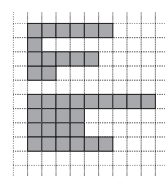


рис. 3

Фигура на рисунке 1 является правильной 7-расческой. Хотя фигура занимает 8 строк, она является 7-расческой, так как на второй и третьей снизу строках находится одно и то же звено (соседние звенья обязательно должны быть разной ширины, но звено может быть высотой более 1 строки). Фигура на рисунке 2 не является правильной расческой, так как левая  $x$ -координата звеньев не совпадает. Фигура на рисунке 3 также не является правильной расческой, так как не является связной фигурой. Можно заметить, что наилучший прямоугольник это то же самое что наилучшая 1-расческа (расческа высоты 1 звено). Очевидно, что для каждой длины  $k$  в матрице можно либо найти наилучшую расческу, либо заявить, что ее нет (то есть наилучшая расческа имеет размер 0). Например, на рисунке 4 обозначена лучшая 3-расческа (знаком X обозначены ненулевые элементы в матрице).

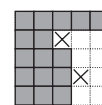


рис. 4

Ваша задача — найти в данной матрице наилучшую расческу среди всех  $k$ -расчесок матрицы (для всех  $k$  от 0 до  $m$ ).

## Задача М. Наилучшая расческа

Имя входного файла: `bestcomb.in`  
 Имя выходного файла: `bestcomb.out`  
 Ограничение по времени: 2 секунды  
 Ограничение по памяти:

После того, как задачи нахождения в числовой матрице наибольших квадратов и прямоугольников из нулей стали



### Формат входных данных

В первой строке входного файла даны два целых числа — высота ( $1 \leq m \leq 2000$ ) и ширина ( $1 \leq n \leq 2000$ ) матрицы соответственно. Далее, в  $m$  строках дано по  $n$  чисел через пробел — исходная матрица.

### Формат выходных данных

В выходной файл необходимо вывести размер наибольшей расчески среди всех  $k$ -расчесок, которую можно построить на нулевых ячейках входной матрицы.

### Примеры

bestcomb.in	bestcomb.out
3 4 0 0 1 0 0 1 0 0 0 0 0 0	7
5 5 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0	20

### Примеры

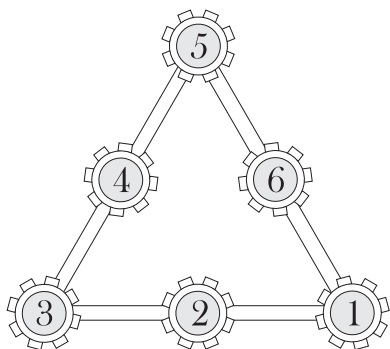
castle.in	castle.out
0 0	POSSIBLE
0 1	0.00 0.00
? ?	0.00 1.00
? ?	0.00 2.00
2 2	1.00 2.00
? ?	2.00 2.00
	1.00 1.00
0 1	IMPOSSIBLE
? ?	
? ?	
? ?	
-2 -1	
-1 0	

### Задача N. Крепость

Имя входного файла: castle.in  
Имя выходного файла: castle.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

В 2123 году на территории Ленинградской Области археологи обнаружили останки старинной крепости. К сожалению, некоторые фрагменты крепости не сохранились. Археологи точно знают, что крепость содержала шесть башен, три из которых являются вершинами треугольника, а остальные три — серединами сторон этого треугольника. Известно точное расположение только некоторых башен. Ваша задача определить расположение всех башен.

На рисунке приведен возможный вид крепости сверху:



### Формат входных данных

Входной файл содержит шесть строк, каждая строка представляет собой описание башни. Если расположение башни известно, то строка содержит два целых числа, разделенных одним пробелом, в противном случае два знака вопроса ('?'), разделенных одним пробелом. Башни даны в порядке обхода, начиная с любой угловой башни.

### Формат выходных данных

В выходной файл необходимо вывести IMPOSSIBLE, если однозначно восстановить расположение всех башен невозможно, в противном случае вывести в первой строке POSSIBLE, а в следующих шести строках расположение башен, в таком же порядке как во входном файле. Числа необходимо вывести по крайней мере с двумя знаками после точки.

### Задача O. Цифры

Имя входного файла: digits.in  
Имя выходного файла: digits.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Назовем *суммой цифр числовой последовательности* сумму цифр всех ее чисел. Например, для последовательности чисел 14, 22, 239 сумма цифр будет равна  $(1 + 4) + (2 + 2) + (2 + 3 + 9) = 23$ .

Ваша задача — при данном  $n$  найти сумму цифр следующей числовой последовательности:

$$1, 2, 3, \dots, 10^n - 1$$

### Формат входных данных

На первой строке входного файла находится целое число  $n$  ( $1 \leq n \leq 100000$ ).

### Формат выходных данных

Выведите в выходной файл одно число — искомую сумму цифр числовой последовательности.

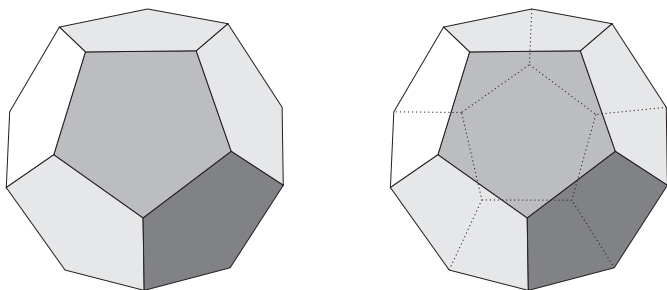
### Пример

digits.in	digits.out
1	45

### Задача P. Додекаэдр

Имя входного файла: dodeca.in  
Имя выходного файла: dodeca.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Додекаэдром называется правильный многогранник, состоящий из 12 граней. Каждая его грань является правильным пятиугольником. Двое полицейских, находящихся на некоторых гранях додекаэдра (возможно, на одной и той же), гоняются за З. Хуссейном, мировым террористом номер один, который также расположился на одной из граней додекаэдра. Полицейские и З. Хуссейн ходят последовательно — сначала двигается один из полицейских (любой), затем Хуссейн. Каждый ход заключается в перемещении на соседнюю грань, а соседней называется грань, имеющая с данной гранью общее ребро. Оставаться на месте в свой ход нельзя. Перемещения повторяются до тех пор, пока Хуссейн не будет пойман. Если злодей в свой ход встает на грань, где находится полицейский, то он будет пойман в следующий же ход полицейского. Если полицейский в свой ход встает на грань, где находится злодей — то злодей сразу считается пойманным.



### Формат входных данных

Дано расстояние  $n$  между полицейскими на додекаэдре. Расстоянием называется минимальное количество ходов, которое потребуется одному из них, чтобы оказаться на одной грани с другим.

### Формат выходных данных

Выведите, какое максимальное количество ходов потребуется, чтобы гарантированно поймать З. Хуссейна, где бы на додекаэдре он ни находился. Координаты Хуссейна всегда известны полицейским, и наоборот. Полицейские также всегда знают координаты друг друга.

### Пример

dodeca.in	dodeca.out
1	5

### Задача Q. Эндшпиль

Имя входного файла: `endspiel.in`  
Имя выходного файла: `endspiel.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Партия в шахматы подходила к концу, и на поле оставалось всего две фигуры: белый и черный короли. Однако, вопреки шахматным правилам, игроки решили доиграть эту партию до самого конца, руководствуясь девизом “остаться должен только один!”. Но короли, в целях государственной безопасности, были заменены “охранниками”. Охранник — это фигура, которая в каждый свой ход перемещается ровно на одну клетку по горизонтали или вертикали, и бьет эти же клетки. Вам необходимо сказать, чем закончится партия при оптимальной игре обоих соперников.

### Формат входных данных

Во входном файле дано через пробел расположение охранников (сначала координаты белого, затем черного), а затем информация о том, кто будет ходить первым (символ ‘B’ — если первым будет ходить черный охранник, и символ ‘W’ в противном случае). Расположение фигур дано в стандартной шахматной записи — сначала указывается символ столбца шахматного поля (строчная латинская буква от a до h), затем указывается номер строки шахматного поля (цифра от 1 до 8).

### Формат выходных данных

Необходимо вывести, чем закончится партия при оптимальной игре обоих соперников: **White wins** — если выиграют белые, **Black wins** — если победят черные, и **Draw** — если будет ничья. Партия заканчивается ничьей, если ни один из соперников не может победить. Партия закончится победой одной из сторон, когда охранник “съедает” охранника соперника. Изначально охранники могут занимать абсолютно любую позицию на поле, за исключением того, что не могут находиться на одной клетке.

### Пример

endspiel.in	endspiel.out
g6 g7 B	Black wins

### Задача R. Маленький шахматный Ним

Имя входного файла: `smallnim.in`  
Имя выходного файла: `smallnim.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

В шахматной стране в последнее время стала очень популярна игра Ним. Правила игры просты. Перед началом игры на стол выкладываются несколько кучек камней. Два игрока ходят по очереди и за каждый ход берут из одной любой кучки произвольное число камней. Игрок, который берет последний камень из последней оставшейся кучки — проигрывает.

Черный и белый короли тоже решили сыграть в Ним, но игра оказалась слишком сложной, поэтому они решили немного изменить правила: камни теперь можно брать не из любой кучки, а только из такой, в которой содержится минимальное число камней.

После нескольких партий обнаружилось, что черный король очень хорошо освоил эту игру и каждый раз ходит наилучшим образом, то есть если у черного короля есть хотя бы один ход, ведущий к победе, то он его и делает. Таким образом, белый король стал подозревать, что исход каждой партии можно определить по начальной позиции в игре.

Теперь он хочет, чтобы вы, как главный мудрец шахматной страны, помогли ему определить по количеству камней в каждой кучке, может ли он выиграть, и если может, то сколько камней ему нужно взять из минимальной кучки для того, чтобы сохранить возможность победы.

### Формат входных данных

В первой строке входного файла записано целое число  $n$  — количество кучек ( $1 \leq n \leq 100$ ). Во второй строке входного файла записано  $n$  целых чисел  $b_i$  ( $1 \leq b_i \leq 1000$ ) — количество камней в  $i$ -ой кучке.

### Формат выходных данных

Если белый король может выиграть при наилучшей игре черного короля, то в первую строку выходного файла выведите слово **YES**, а во вторую строку — целое число  $s$ , которые определяет, сколько камней необходимо взять белому королю из минимальной кучки на первом ходе.

Если же белый король не может выиграть, то в первую строку файла выведите **NO**.

### Примеры

smallnim.in	smallnim.out
2	YES
2 3	1
1	NO
1	

### Задача S. Вычитание

Имя входного файла: `subtract.in`  
Имя выходного файла: `subtract.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Сотрудник одного известного банка неожиданно обнаружил, что в некоторых программах, обслуживающих работу банка, процедура вычитания одного числа из другого иногда выдает ошибочный результат, что, естественно, угрожает безопасности банка. Например, один клиент заявил, что, когда на его счету было \$296, и он снял с него \$125, его



счет почему-то оказался пустым, хотя на нем должен был остаться \$171. Вам нужно срочно написать простую, но очень ответственную программу, которая правильно реализует вычитание одного числа из другого, пока электронные воры не воспользовались дыркой в системе безопасности.

### Формат входных данных

Во входном файле даны два целых числа, каждое из которых лежит в пределах от  $-2^{63}$  до  $2^{63} - 1$ .

### Формат выходных данных

Необходимо вывести результат вычитания второго числа из первого.

### Примеры

subtract.in	subtract.out
296 125	171

### Пример

vars.in	vars.out
i, j, k : integer ;	boolean: 1
flag: boolean;	char: 3
length: integer;	double: 0
c1, c2: char;	integer: 4
c3: char;	

## Задача Т. Сколько же переменных?

Имя входного файла: `vars.in`  
Имя выходного файла: `vars.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти:

Маленький мальчик Петя очень любит программировать на языке Паскаль. К сожалению, в последнее время он стал замечать, что использует слишком много переменных. Так много, что сам не может толком сосчитать, сколько же в каждой программе их используется.

Поэтому он просит вас написать программу, которая поможет ему в этом нелегком деле.

Так как Петя еще не очень хорошо знает Паскаль, то использует он только четыре стандартных типа: `boolean`, `char`, `double`, `integer`.

### Формат входных данных

Во входном файле содержится несколько строк с объявлениями переменных.

Каждая строка входного файла является объявлением переменных одного типа.

Все объявления являются корректными с точки зрения синтаксиса языка Паскаль, названия переменных не повторяются, при объявлении используются только стандартные типы: `boolean`, `char`, `double`, `integer`.

Формально каждое объявление имеет следующий вид:

```
<идентификатор> {',' <идентификатор>} ':' <тип> ';' ;
```

где `<тип>` — это одна из строк `boolean`, `char`, `double`, `integer`; `<идентификатор>` — строка, состоящая из букв, цифр или символов `'_'` и начинающаяся с буквы или символа `'_'`.

Строки `<тип>` и `<идентификатор>`, а также символы `':'` и `','` и `','` являются неделимыми элементами, между которыми (а также до и после них) может находиться произвольное число пробелов.

Гарантируется, что во входном файле количество строк — не более 100, в каждой строке не более 20 переменных, и название каждой переменной не длинее 1000 символов.

### Формат выходных данных

В выходной файл необходимо вывести четыре строки, в следующем формате:

```
boolean: <количество объявленных переменных типа boolean>  
char: <количество объявленных переменных типа char>  
double: <количество объявленных переменных типа double>  
integer: <количество объявленных переменных типа integer>
```

Обратите внимание, что между двоеточием и числом переменных должен быть выведен один пробел.