

Задача А. Минёр

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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 5×5) 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

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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 $[0..360)^\circ$.

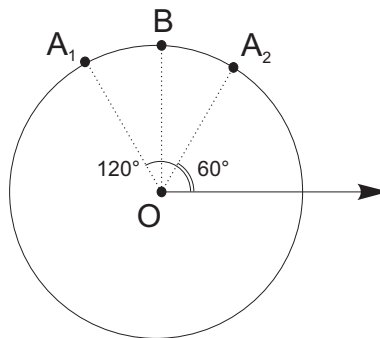
Формат выходных данных

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

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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 8×8 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 чёрные). Порядок начального размещения фигур на строках 1 и 8 (от **a** к **h**): ладья, конь, слон, ферзь, король, слон, конь, ладья. Горизонтали 2 и 7 занимают пешки.

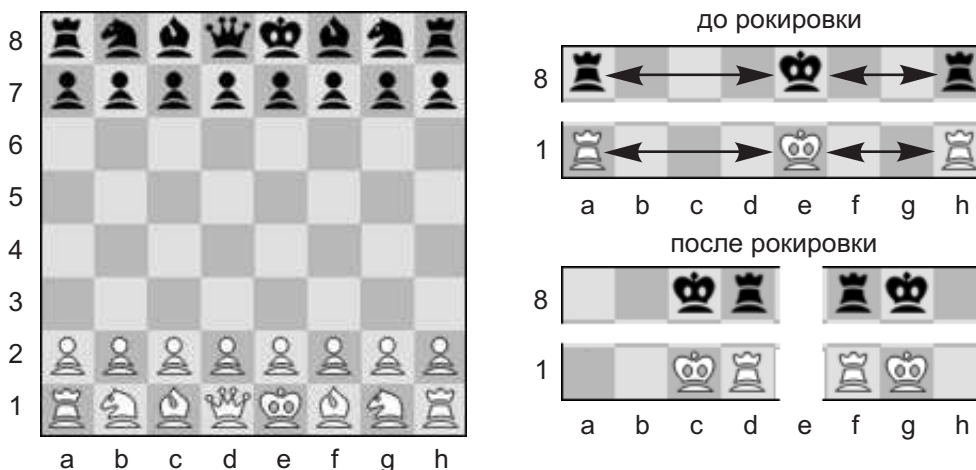
Игроки делают ходы по очереди, сначала ходят белые, затем черные, и так далее. В свой ход игроки перемещают ровно одну фигуру по правилам хода этой фигуры.

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, B — 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 чёрная 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 $pe2-e1q$ 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 выведите 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 a 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.pppp..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

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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

Задача E. Квадрат

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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.

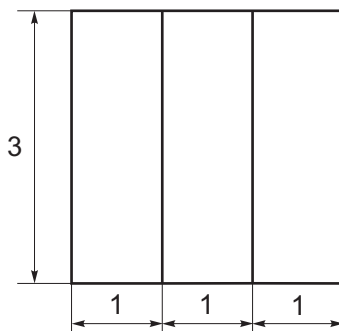
Формат выходных данных

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 F. Два капитана

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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 G. Игра

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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 Н. Крестики-нолики

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

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 картине 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

<code>xo.in</code>	<code>xo.out</code>
<code>.X ..X ..OXO ...XO</code>	<code>CORRECT</code>
<code>.... .XO... .XO. .XO... .XO. .XO...</code>	<code>INCORRECT</code>

Задача I. Муравьи на кубической Земле

Ограничение по времени: 2 секунды
Ограничение по памяти: 64 Мб

В условиях данной задачи будем пользоваться предположением, что Земля имеет форму куба, и каждая грань — квадрат $m \times m$, расчерченный на клетки размером 1×1 .

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.

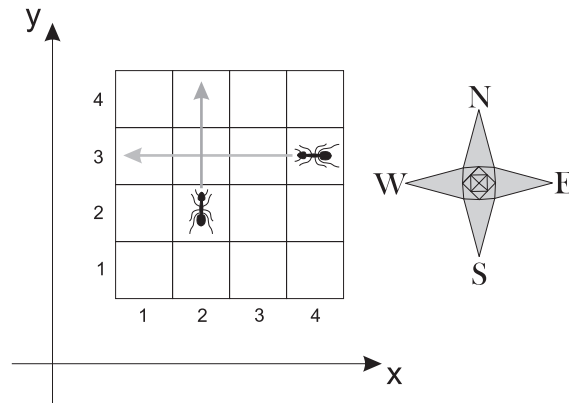
Несколько муравьев как в начальный момент времени, так и в любой другой, могут оказаться на одной клетке. Это никак не влияет на траектории их движения.

Формат выходных данных

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

Пример

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



Верхняя грань куба, вид сверху