Задача 1. Наносборка

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 МиБ

Одной из проблем, возникающих при создании деталей наноконструкций, является колоссальная затрата времени, которая требуется для сборки нанодеталей из отдельных атомов. Перемещение каждого из огромного числа атомов требует использования манипулятора. В то время, как манипулятор занят перемещением некоторого атома, он не может перемещать другие атомы. Использование большого числа манипуляторов одновременно является невозможным, так как манипуляторы имеют внушительный размер и не помещаются в окрестности пространства, где проводится сборка нанодетали. В лаборатории нанотехнологического государственного университета (НГУ) был предложен метод, призванный решить указанную проблему.

Идея предложенного метода заключается в отказе от использования манипуляторов и замене их на разработанный в лаборатории полуплоскостной отражатель. Этот прибор совершает манипуляции над атомами, расположенными на одной стороне листа графена, которую можно считать плоскостью. Принцип действия полуплоскостного отражателя заключается в том, что лист графена сгибается по некоторой заданной прямой и все атомы, которые находились по одну (определённую) сторону относительно прямой сгиба перемещаются на противоположную сторону, занимая места, полученные в результате отражения их исходных мест относительно этой прямой. Все атомы, находившиеся по другую сторону от прямой сгиба, остаются на своих местах. После того, как атомы переместились, лист графена разгибается. Таким образом, в результате нескольких последовательных операций сгиба, на поверхности листа графена может быть перемещено большое количество атомов.

По начальному положению атомов на поверхности листа графена и описанию последовательности операций, совершенных полуплоскостным отражателем, вам требуется определить положение атомов после произведённых операций.

Формат входного файла

В первой строке входного файла задано два целых числа N и M — количество атомов и количество операций ($1 \le N \le 10^5$, $1 \le M \le 10$).

В следующих N строках заданы начальные позиции атомов. Каждая позиция описывается двумя записанными через пробел целыми координатами x и y, модуль которых не превосходит 10^4.

Затем в следующих M строках описаны операции, совершенные полуплоскостным отражателем. Для описания каждой операции используются четыре записанных через пробел целых числа x_1, y_1 и x_2, y_2 — координаты начала и конца вектора, лежащего на прямой сгиба (модуль каждой из этих координат также не превосходит 10^4). Гарантируется, что начало и конец вектора не совпадают. При выполнении операции перемещаются атомы, находящиеся в полуплоскости, лежащей справа от заданного вектора.

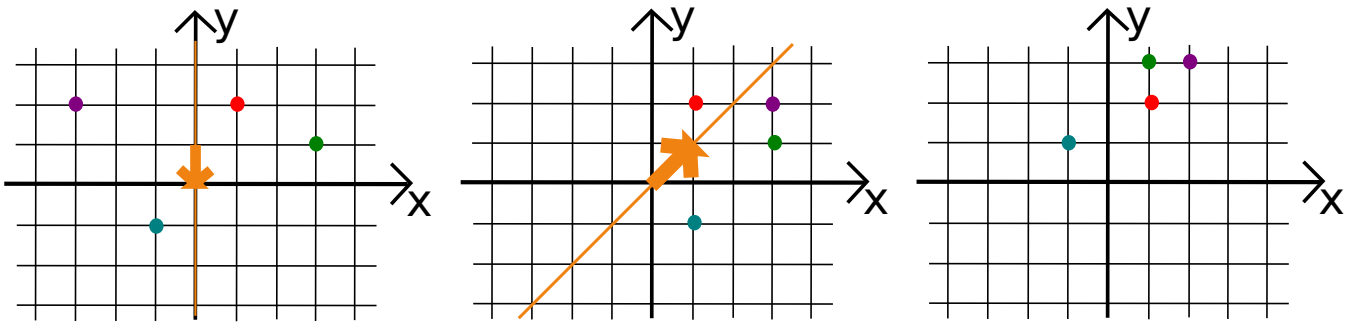
Формат выходного файла

В выходной файл необходимо вывести N строк, содержащих координаты атомов в том же порядке, что и во входных данных, с абсолютной ошибкой, не превосходящей 10^{-5}.

Пример

input.txt	output.txt
4 2 1 2 3 1 -1 -1 -3 2 0 1 0 0 0 0 1 1	1 2 1 3 -1.0 1 2.0 3.0

Пояснение к примеру



Задача 2. Плэй-офф

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 МиБ

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:

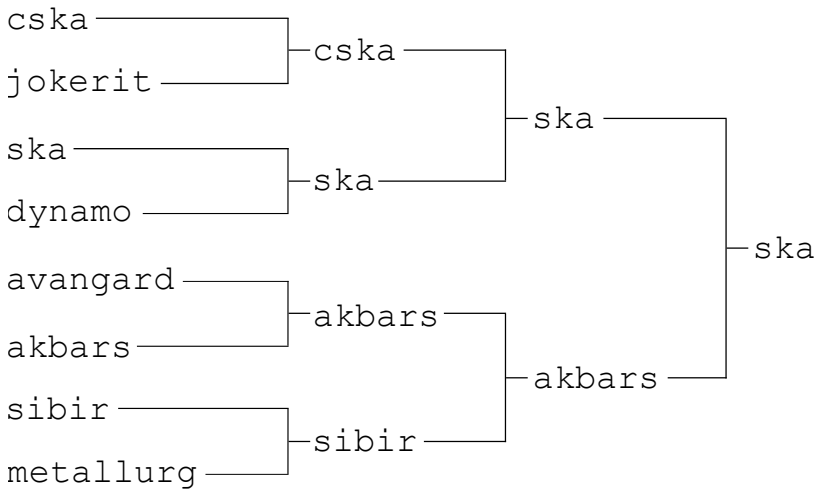
- `Win`, если первая команда *сильнее* второй;
- `Lose`, если вторая команда *сильнее* первой;
- `Unknown`, если про данные две команды нельзя сказать, что одна из них *сильнее* другой.

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

Пояснение к примеру

Турнирная сетка в примере выглядит следующим образом:



Задача 3. Неравенства

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	5 секунд (для Java)
Ограничение по памяти:	256 МиБ

Дана система неравенств. Неравенства имеют вид $s < t$ и $s \le t$, где каждое из s и t может быть целым числом или переменной x_i.

Необходимо найти какое-нибудь решение этой системы в целых числах, или же выяснить, что таких решений нет.

Формат входного файла

В первой строке входного файла записаны два целых числа n и k — количество неравенств и количество переменных соответственно ($0 \le n \le 10^5$, $1 \le k \le 10^5$).

В следующих n строках описаны неравенства, в каждой по одному. Описание неравенства состоит из пяти целых чисел. Первое — это тип неравенства: `0` — нестрогое ($\le$), `1` — строгое ($<$). Второе и третье число кодируют левую часть неравенства: «`0` i» означает переменную x_i ($1 \le i \le k$), «`1` a» — целое число a ($-10^9 \le a \le 10^9$). Правая часть неравенства аналогичным образом кодируется с помощью четвёртого и пятого чисел.

Формат выходного файла

Если система неравенств не имеет решений в целых числах, то в выходной файл нужно вывести слово `NO`.

В противном случае, в первую строку необходимо вывести слово `YES`, а в следующие k строк — любое решение системы, а именно, в i-ую из них нужно поместить целое число a_i — значение x_i ($1 \le i \le k$, $-2 \cdot 10^9 \le a_i \le 2 \cdot 10^9$).

Гарантируется, что если система разрешима, то существует решение, удовлетворяющее ограничениям.

Пример

input.txt	output.txt
2 2 1 0 1 0 2 0 0 2 0 1	NO
3 2 1 0 1 0 2 0 1 5 0 1 0 0 2 1 7	YES 5 7

Пояснение к примеру

Во втором примере описана следующая система неравенств:

$$\begin{cases} x_1 < x_2 \\ 5 \le x_1 \\ x_2 \le 7 \end{cases}$$

Одним из её решений является решение $x_1 = 5$, $x_2 = 7$.

Задача 4. Как измерить океан?

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

Козьма Прутков как-то задался вопросом, чем можно измерить глубину Восточного океана. Для Остапа ответ очевиден — конечно же, астролябией. Сама меряет, было бы что мерить. Проблема только одна: как бы эту астролябию на дно океана спустить.

Для этого у Остапа есть мотки проволоки различного сечения из различных материалов, при этом запас проволоки каждого вида неограничен. В каждом мотке вся проволока имеет одинаковое сечение. Куски проволоки из различных мотков можно скреплять друг с другом в их концевых точках очень прочным клеем, т.е. проволока может порваться где угодно, но не в месте скрепления различных кусков.

Требуется нарезать из мотков проволоки куски, затем выстроить их в каком-либо порядке и склеить в одну цепочку, а к концу приклеить астролябию. В частности, нельзя склеивать больше двух кусков проволоки в одном месте или прикреплять две или более проволоки к грузу. Вся конструкция должна доставать до дна океана, и проволока не должна нигде порваться. Проволока сечения S, сделанная из материала с пределом прочности σ может порваться в любом месте, если общий вес астролябии и проволоки ниже этого места превышает $\sigma \cdot S$.

Определите, какой максимальный вес может иметь астролябия, чтобы не порвать проволоку.

Формат входного файла

В первой строке входного файла записано два целых числа d и n, где d — глубина океана, n — количество видов проволоки ($1 \le d \le 20000$, $1 \le n \le 10^5$).

Далее в n строках приведены по три целых числа s_i, p_i и σ_i — характеристики i-го мотка проволоки, где s_i — площадь его поперечного сечения, p_i — плотность материала i-го мотка, т.е. вес единицы объема, σ_i — предел прочности материала этого мотка, т.е. максимальный допустимый вес на единицу площади, который может выдержать материал, при превышении его проволока рвется ($1 \le s_i \le 10^3$, $1 \le p_i \le 20$, $10 \le \sigma_i \le 10^3$).

Формат выходного файла

В выходной файл необходимо вывести одно число — максимально возможный вес астролябии, который мы можем опустить на глубину d, так чтобы проволока не порвалась. Если проволока не может достичь дна, то вывести число ноль. Абсолютная погрешность ответа не должна превышать 10^{-5}.

Примеры

input.txt	output.txt
10 1 2 8 200	240.00
30 1 2 8 200	0.0

Задача 5. Спортивное ориентирование

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1.5 секунды
Ограничение по памяти:	256 МиБ

В лесу Академгородка есть полянки, соединенные прямолинейными тропинками, по которым можно ездить на велосипеде в любую сторону. Все полянки пронумерованы числами от 1 до N. Переезжать с одной тропинки на другую разрешается только на полянках. Любая тропинка соединяет две различные полянки и не проходит ни через какую третью полянку.

В этом лесу проводятся соревнования по спортивному ориентированию. Каждый спортсмен должен стартовать с полянки номер 1 и финишировать на полянке номер N. При этом ему требуется отметиться на K полянках с номерами $a_1, a_2, \ldots, a_K$ в заданном порядке.

В ходе соревнования участник может бегать прямиком по лесу, а может ездить по тропинкам на велосипеде. На каждой полянке есть пункт проката велосипедов, где можно брать и сдавать велосипеды. В каждом пункте проката всегда есть велосипеды в наличии. Сдавать велосипед можно в любом пункте проката, а не только там, где его выдали. Оставлять велосипед посреди тропинки строго запрещается, и бегать по лесу с велосипедом тоже нельзя. Пересекать тропинки на бегу **не** запрещается.

По каждой тропинке можно проехать сколько угодно раз, и на каждой полянке можно оказаться сколько угодно раз. Отмечаться на полянке не обязательно в первое ее посещение.

Необходимо определить последовательность номеров полянок, через которые требуется проехать или пробежать, чтобы иметь возможность отметиться в заданной последовательности. Нужно так составить маршрут, чтобы время прохождения пути было минимальным.

Формат входного файла

В первой строке входного файла записаны пять целых чисел N, M, K, V_r, V_f, где N — количество полянок в лесу, M — количество тропинок, K — количество полянок, на которых нужно отметиться, V_r — скорость езды на велосипеде по тропинке, V_f — скорость бега по лесу в метрах в секунду ($3 \le N \le 1\,600$, $0 \le M \le 100\,000$, $1 \le K \le 20$, $1 \le V_f \le V_r \le 1\,000$).

В следующих N строках описываются полянки в порядке от 1 до N. Каждая полянка задается своими x и y координатами ($|x|, |y| \le 10\,000$). Все координаты целые, задаются в метрах. Нет полянок с одинаковыми координатами.

В следующих M строках описываются тропинки. Каждая тропинка задается двумя различными целыми числами — номерами полянок, которые она соединяет. Гарантируется, что каждую пару полянок соединяет не более одной тропинки.

В последней строке через пробел записано K целых чисел — номера полянок, на которых следует отметиться в указанном порядке. Все эти числа различны и лежат в диапазоне от 2 до $N - 1$.

Формат выходного файла

В первую строку выходного файла необходимо вывести одно вещественное число — минимальное время в секундах, которое требуется для прохождения заданной дистанции.

В следующей строке нужно выдать последовательность номеров полянок, через которые проезжает или пробегает участник соревнования. Если между двумя рядом стоящими в этой последовательности полянками нет тропинки, то считается, что спортсмен пробегает от одной до другой напрямую через лес, а если есть между ними тропинка, то спортсмен ее проезжает на велосипеде.

Абсолютная или относительная погрешность вашего ответа не должна превышать 10^{-7}.

Пример

input.txt	output.txt
4 3 2 10 4 0 0 100 0 0 400 100 400 1 3 3 4 2 4 2 3	85.00 1 2 4 3 4

Задача 6. Ставки

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

Наиболее популярными ставками в букмекерских конторах являются ставки на итоговый результат матча. Так, например, в футболе (вид спорта, в котором в одном матче две команды играют между собой), можно делать ставки на все три возможных исхода матча: «победа первой команды», «ничья» и «победа второй команды». На каждое из событий букмекер выставляет некоторый коэффициент k: если матч закончился с исходом, на который была сделана ставка, то возвращается сумма в k раз бо̀льшая, чем ставка, иначе не возвращается ничего.

Популярны также и «live-ставки» — ставки, сделанные во время матча. По ходу игры, в зависимости от текущего результата и оставшегося до конца матча времени, изменяются и коэффициенты. Можно сделать ставку сразу на несколько разных исходов, в том числе и на все возможные. Одним из способов зарабатывать на ставках являются, так называемые, «вилки» — ставки одновременно на все три события таких сумм, что, независимо от исхода матча, ставка, сделанная на победивший исход матча, окупит затраты.

Конечно, букмекерам допускать такие ситуации невыгодно, и сделать вилку просто так не получится: при выставлении коэффициентов букмекеры руководствуются не только своими предпочтениями, но и вынуждены оглядываться на других букмекеров. Однако, существует несколько разных способов сделать «вилку» на «live-ставках», например, когда коэффициенты разных контор меняются после забитого гола. В этот небольшой интервал времени, когда одна контора уже изменила коэффициенты, а другая — нет, можно попытаться сделать вилку. Есть и более рискованные варианты, не гарантирующие положительного результата: поставить в матче явно неравных команд на более слабую, и в случае, если она забьёт, или ничейный счёт продержится достаточно долго, коэффициенты на два других исхода поднимутся до такого уровня, что вилка станет возможной.

В данной задаче за игрока уже выбраны лучшие коэффициенты по всем букмекерским конторам для каждого исхода. Необходимо определить, является ли текущая ситуация «вилкой»: возможно ли сделать такие ставки на все три исхода, чтобы независимо от результата матча оказаться строго в выигрыше.

Формат входного файла

Первая строка входного файла содержит целое число Q — количество запросов ($1 \le Q \le 10^4$).

Следующие Q строк содержат описания запросов. Каждый запрос состоит из трёх вещественных чисел w, d, l, заданных не более, чем с 5 знаками после точки — коэффициентов на победу первой команды, на ничью и на победу второй команды, соответственно ($1 \le w$, d, $l \le 10$).

Формат выходного файла

В выходной файл на каждый запрос нужно вывести ровно по одной строке: если вилка при данных в запросе коэффициентах возможна, то выведите слово `YES`, иначе — `NO`.

Пример

input.txt	output.txt
3 2.5 3.5 4.5 3.0 3.0 3.0 2.0 3.0 4.0	YES NO NO

Пояснение к примеру

В первом запросе можно сделать ставки \$250, \$200 и \$150: в случае выигрыша можем получить соответственно \$625, \$700 и \$675, что в любом случае будет больше суммы ставок \$600.

Во втором запросе, если на все исходы сделать одинаковую ставку, то мы гарантированно ничего не проиграем, но и не выиграем. При разных ставках существует вероятность проигрыша.

В третьем запросе при любых ставках возможен проигрыш.

Задача 7. Муравей на дороге

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

Муравей ночью вышел к дороге. Дорога представляет собой бесконечную полосу, ограниченную прямыми $y = 0$ и $y = 1$. На дороге есть два бесконечных в обе стороны горизонтальных ряда фонарей: южные и северные. Каждый фонарь освещает параболический участок дороги. Фонари в каждом ряду находятся на одном и том же равном расстоянии d друг от друга.

k-й южный фонарь (k — произвольное целое число) освещает все точки с координатами (x, y), удовлетворяющими неравенству:

$$0 \le y \le a(x - b_k)^2 + c, \text{ где } b_k = b + kd.$$

Аналогично, для k-ого северного фонаря зона освещённости задаётся неравенством:

$$A(x - B_k)^2 + C \le y \le 1, \text{ где } B_k = B + kd.$$

Муравей находится южнее дороги (в зоне $y < 0$) и хочет перейти дорогу (попасть в зону $y > 1$) по кратчайшему пути, не попадая под свет фонарей. Он может начать и закончить свой путь из любой точки на соответствующем краю дороги.

Вам требуется определить длину этого пути.

Формат входного файла

В первой строке входного файла записано вещественное число d — расстояние между фонарями в ряду ($0.01 \le d \le 1$).

Во второй строке записаны три вещественных числа a, b, c, где a — квадратичный коэффициент парабол южных фонарей ($a < 0$), b — x-координата одного из южных фонарей, c — y-координата всех южных фонарей. В третьей и последней строке записаны три вещественных числа A, B, C — аналогичные параметры северных фонарей ($A > 0$).

Для параметров фонарей выполняются ограничения: $0.001 \le -a, A \le 10$, $0 \le b, B \le 1$, $0.001 \le c, C \le 0.999$.

Гарантируется, что зоны освещённости любых двух фонарей не касаются с погрешностью 10^{-6} (но могут пересекаться).

Формат выходного файла

В выходной файл необходимо вывести одно вещественное число — минимальное расстояние, которое должен пройти муравей, чтобы перейти дорогу, не попадая под свет фонарей. Если он не может этого сделать, выведите -1.

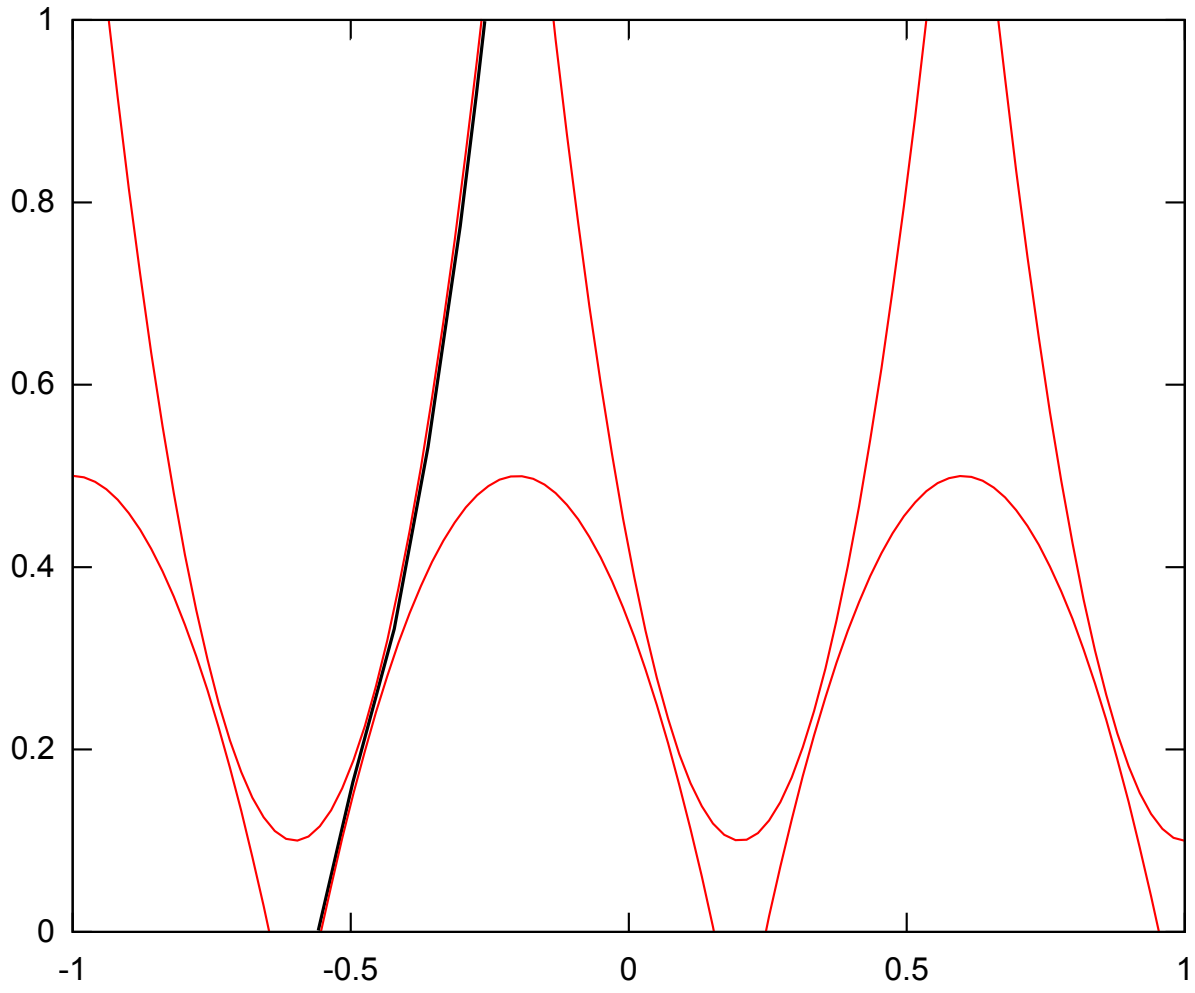
Абсолютная или относительная погрешность вашего ответа не должна превышать 10^{-6}.

Примеры

input.txt	output.txt
1.0 -1.0 0.0 0.1 1.0 0.0 0.9	1
1.0 -1.0 0.0 0.3 1.0 0.0 0.7	-1
0.8 -4.0 0.6 0.5 8.0 0.2 0.1	1.043157963

Пояснение к примеру

Границы освещенности фонарей в третьем примере выглядят следующим образом:



Задача 8. Букет

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1.5 секунды
	3 секунды (для Java)
Ограничение по памяти:	256 МиБ

Нежная телячья душа командора требует жертвоприношения любимой женщине. Правда, денег после крушения конторы «Рога и копыта» осталось всего ничего, и все они будут истрачены на шикарный букет. Шикарность же букета, по мнению Остапа, есть количество видов цветов, в этот букет входящих. Правда, экономный Михаил Самуэльевич настаивает, что вовсе не обязательно тратить все деньги, можно ведь составить более шикарную композицию и за меньшую сумму, но — шиковать, так шиковать! Деньги надо потратить обязательно все.

Необходимо найти максимально возможное количество видов цветов, составляющих букет наперед заданной стоимости.

Формат входного файла

В первой строке входного файла записано два целых числа N и S, где N — количество цветов, S — имеющаяся сумма денег ($1 \le N \le 10^3$, $1 \le S \le 5 \cdot 10^4$).

В каждой из следующих N строк описан имеющийся в продаже цветок двумя целыми числами a и b — номером вида цветка и его стоимостью соответственно ($1 \le a \le 10^3$, $1 \le b \le S$). Каждый цветок имеется в продаже в единственном экземпляре, однако параметры (вид и стоимость) различных цветков могут совпадать.

Формат выходного файла

В выходной файл необходимо вывести одно целое число — максимально возможное количество видов цветов, составляющих букет стоимости S. Если составить букет указанной стоимости невозможно, то вывести слово `Impossible`.

Примеры

input.txt	output.txt
6 20 2 5 1 5 2 6 3 3 4 4 1 5	3
1 10 1 8	Impossible

Комментарий

В первом примере нельзя покупать цветок 3-его вида, потому что все оставшиеся деньги не получится потратить на другие цветы. Однако можно купить цветы оставшихся трёх видов. Для этого нужно купить цветы стоимостью 4 и 6, а также любые два из трёх цветов стоимостью 5.

Задача 9. Хэш-функция

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 МиБ

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 ($1 \le Q \le 100$).

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

Задача 10. Civilization

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 МиБ

Действия пошаговой компьютерной стратегии Civilization V происходят на гексагональном поле — двумерной сетке, состоящей из правильных одинаковых шестиугольников. Клетками поля могут быть зелёные равнины, на которых стоят города и по которым передвигаются юниты, и непроходимые горы.

Игроки ходят по очереди. Во время своего хода игрок может перемещать юниты в соответствии с описанными далее правилами.

- У каждого юнита в начале хода есть несколько очков движения (ОД). За один шаг снимается одно ОД. Юнит может ходить, пока он имеет положительное значение ОД.
- Каждый юнит за одно ОД может шагнуть в смежную по стороне клетку.
- Во время хода игрока в одной клетке может оказаться несколько юнитов, однако, необходимо чтобы к концу хода в каждой клетке оказалось не более одного юнита.
- Юниты не могут перемещаться по горам.
- Нельзя заходить в (и проходить через) клетки, на которых стоят города противника.

Каждый юнит занимает ровно одну клетку поля. Гарантируется, что начальное положение юнитов удовлетворяет описанными выше правилам: в одной клетке не более одного юнита и никакие юниты не стоят в горах или городах.

Имеются юниты, которые называются катапультами. Они могут обстреливать города. У каждого города имеется несколько очков здоровья (ОЗ). У разных городов может быть различное количество ОЗ.

Катапульты могут обстреливать города по следующим правилам:

- Обстрелять город может только катапульта с положительным значением ОД.
- После обстрела города количество ОД катапульты обнуляется.
- Обстрелять можно только тот город, который находится на расстоянии не дальше двух клеток от места катапульты. В зону обстрела попадают:
  1. клетки, смежные по стороне с клеткой катапульты;
  2. клетки, смежные по стороне с клетками из первого пункта.

  Катапульта может обстрелять город даже в том случае, когда между ней и городом стоит гора, другой город или катапульта.
- За каждый обстрел одной катапультой у города снимается одно ОЗ.
- Значение ОЗ у города не может быть ниже нуля. В то же время, даже если у города ноль ОЗ, то обстреливать его по-прежнему можно, но это не даст никакого эффекта.

В данной задаче участнику дано поле, на котором расположены города противника и дружественные катапульты. Сейчас ход участника. Требуется за этот ход снять защиту, т.е. довести значение ОЗ до нуля, с как можно большего числа городов.

Формат входного файла

В первой строке входного файла дано два целых числа W и H — ширина и высота поля ($1 \le W, H \le 10^9$). Все боевые действия совершаются в пределах игрового поля: в клетках, координаты (x, y) которых лежат в пределах $0 \le x < W$ и $0 \le y < H$ — и катапульты не могут уезжать за его пределы.

Поле состоит из гексагональных ячеек, которые образуют H рядов. Каждый ряд состоит из W ячеек, при этом все ячейки ориентированы так, что сверху и снизу расположены вершины шестиугольника, а слева и справа рёбра, которыми ячейка соединена с соседними ячейками в ряду. Каждый следующий ряд смещён относительно предыдущего на половину ячейки. Ось Ox направлена вдоль нижнего ряда ячеек слева направо. Ось Oy направлена под углом 60 градусов к оси Ox. Также в пояснении к примеру приложена иллюстрация, на которой изображена нумерация ячеек.

Во второй строке дано целое число N — количество городов ($1 \le N \le 4$).

Следующие N строк содержат по три числа x, y и p, которые описывают, соответственно, координаты города и его очки здоровья ($1 \le p \le 100$).

В следующей строке дано целое число C — количество катапульт ($1 \le C \le 100$).

Следующие C строк описывают катапульты: в каждой строке находится по три числа x, y и v — соответственно, координаты катапульты и её очки движения ($1 \le v \le 5$).

В следующей строке дано целое число M — количество клеток, являющихся горами ($0 \le M \le 1\,000$). Следующие M строк содержат по два числа x и y и описывают координаты гор. Все клетки, не являющиеся горами, являются проходимыми равнинами.

Гарантируется, что все координаты попарно различны. Для всех координат x и y выполняются ограничения $0 \le x < W$ и $0 \le y < H$.

Формат выходного файла

В первую строку выходного файла необходимо вывести одно целое число — максимальное количество городов, у которых можно снять защиту.

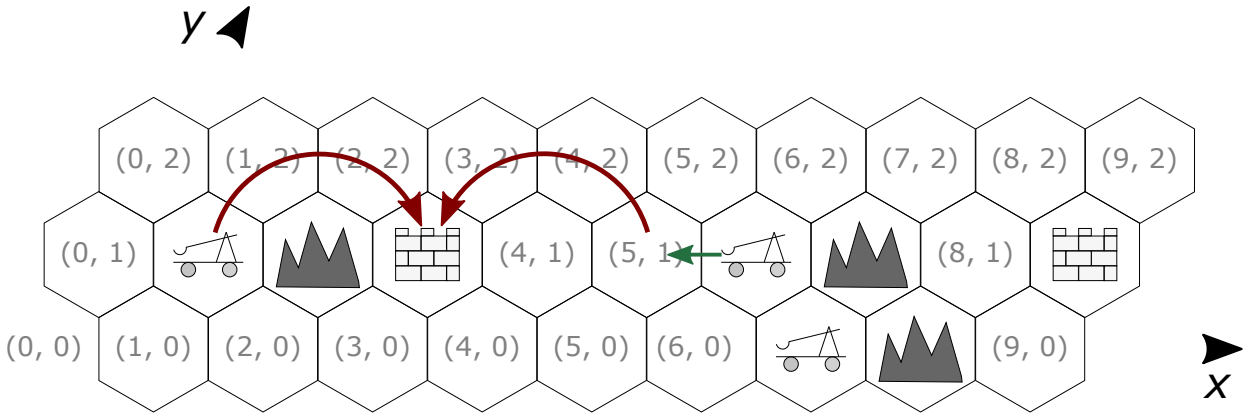
В следующие C строк для каждой катапульты необходимо вывести через пробел по три числа: координаты клетки, куда её следует переместить, а также номер города, который данная катапульта будет обстреливать. Города занумерованы числами, начиная с 1, в том же порядке, в каком они даны во входном файле. Если катапульте не требуется обстреливать город, то нужно вывести 0 (ноль) вместо номера города.

Пример

input.txt	output.txt
10 3 2 3 1 2 9 1 1 3 1 1 1 6 1 2 7 0 3 3 2 1 7 1 8 0	1 1 1 1 5 1 1 7 0 0

Пояснение к примеру

Боевые действия в примере выглядят следующим образом:



Задача 11. Стулья мастера Гамбса

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

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

Пояснение к примеру

Отец Фёдор может купить один предмет из первого гарнитура, и один или два предмета из второго, а на другие виды товаров имеющихся денег не хватит.

Задача 12. Эрудит

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МиБ

Внимательно прочитайте приведённые здесь правила, они могут отличаться от оригинальных правил настольной игры!

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

Комментарий

В примере игроки сделали 6 ходов. Сначала первый игрок составил слова «МЕЧ» и «РЫЧАГ». На следующем ходу второй игрок составил слово «ВЕСНА». На третьем ходу первый игрок составил слова «КРУГ» и «КОРД». На четвёртом ходу второй игрок составил «СЕКАЧ», «ЯЩИК» и «НАЙ». Следующим сходил первый игрок, составив слова «ПОЧКА», «УЛАН», «ЩИ». В последний ход второй игрок составил слова «ЩИТ», «СРОК», «ДЕПО» и «КАПА». В таблице 4 приведено состояние игрового поля в конце.

Таблица 1: Игровое поле

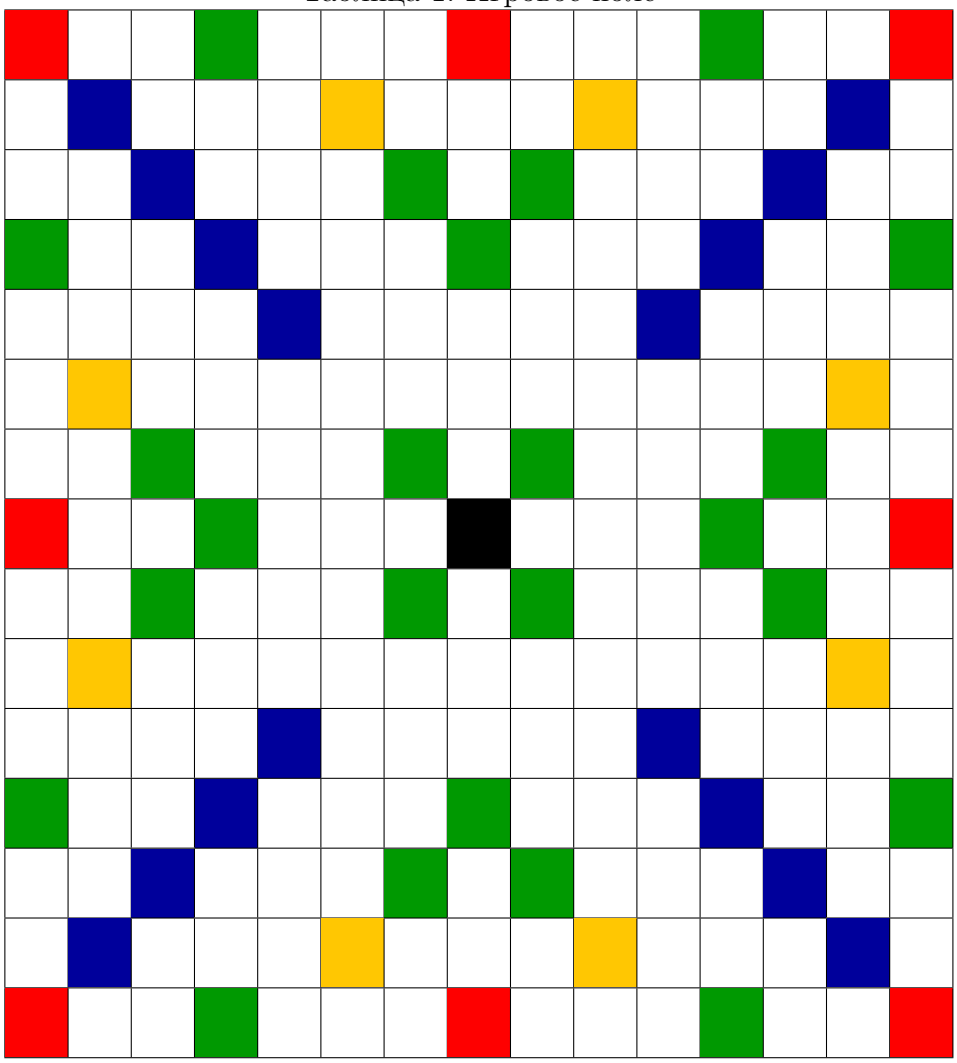


Таблица 2: Значения цветов клеток

Цвет клетки	Обозначение	Коэффициент буквы	Коэффициент слова
Белый	w	1	1
Черный (обозначает стартовую клетку)	s	1	1
Зелёный	g	2	1
Желтый	y	3	1
Синий	b	1	2
Красный	r	1	3

Таблица 3: Базовая цена букв

Используемый номер буквы	Буква	Базовая цена
1	А	1
2	Б	3
3	В	2
4	Г	3
5	Д	2
6	Е	1
7	Ж	5
8	З	5
9	И	1
10	Й	2
11	К	2
12	Л	2
13	М	2
14	Н	1
15	О	1
16	П	2
17	Р	2
18	С	2
19	Т	2
20	У	3
21	Ф	10
22	Х	5
23	Ц	10
24	Ч	5
25	Ш	10
26	Щ	10
27	Ъ	10
28	Ы	5
29	Ь	5
30	Э	10
31	Ю	10
32	Я	3

Таблица 4: Игровое поле после выполненных ходов

						Я								
						Щ	И	Т						
						И								
						К	О	Р	Д					
								Ы						
					С	М	Е	Ч						
				В	Е	С	Н	А	Й					
					К	Р	У	Г						
					А	О	Л							
	Д	Е	П	О	Ч	К	А	П	А					
							Н							