Задача A. Игра «Bloxx city»

Имя входного файла:	`bloxx.in`
Имя выходного файла:	`bloxx.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	64 мегабайта

На новом мобильном телефоне фирмы *Mokia* установлена игра «Bloxx city». Действие этой игры происходит на прямоугольном поле размером n на m.

Каждая клетка этого поля может быть пустой или в ней может находиться дом некоторой высоты. За один ход игрок может в пустой клетке построить дом. При этом дом высоты h в некоторой клетке можно строить только, если в клетках, имеющих с рассматриваемой общую сторону, находятся дома всех высот от 1 до $h-1$ (соответственно, дом высоты 1 можно строить в любой свободной клетке поля).

Изначально на поле уже стоят некоторые дома. Цель игры — построить дополнительные дома так, чтобы их суммарная высота была как можно большей.

Напишите программу, которая позволяет добиться этой цели.

Формат входного файла

Первая строка входного файла содержит два целых числа: m и n ($1 \le m, n \le 10$, $m \cdot n \le 16$). Последующие m строк описывают игровое поле: каждая из них содержит по n чисел $h_{i,j}$ ($0 \le h_{i,j} \le 5$), задающих высоты домов, изначально стоящих на поле. Если некоторое из чисел $h_{i,j}$ равно нулю, то эта клетка пуста.

Формат выходного файла

В первой строке выходного файла выведите максимальную возможную сумму высот домов. Во второй строке выходного файла выведите k — число ходов, которые необходимо сделать, чтобы добиться такой суммы высот. В последующих k строках выведите описание этих ходов: каждая из них должна содержать по три числа: r, c, h — соответственно, координаты клетки, в которой строится дом, и его высоту ($1 \le r \le m$, $1 \le c \le n$).

Примеры

<table>
<tr><th>bloxx.in</th><th>bloxx.out</th></tr>
<tr><td><pre>3 4
0 2 4 0
0 2 0 3
0 0 0 0</pre></td><td><pre>29
8
1 4 1
2 1 1
3 3 1
3 1 2
3 4 2
1 1 3
3 2 3
2 3 5</pre></td></tr>
</table>

Задача B. Кодовый замок

Имя входного файла:	`code.in`
Имя выходного файла:	`code.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вася очень любит использовать кодовые замки. Как известно, кодовый замок состоит из барабана с несколькими кольцами. Причем, у каждого кольца есть несколько положений, каждое из которых соответствует некоторой цифре в k-ичной системе счисления. Таким образом, любое состояние всего кодового замка с n кольцами можно представить некоторым n-значным числом в k-ичной системе счисления. При этом замок открывается только в одном фиксированном состоянии.

Так как код замка очень легко забыть, для его запоминания Вася пользуется хитрым способом. После закрытия замка Вася поворачивает замок таким образом, чтобы число, кодирующее его состояние, было предыдущим (в порядке возрастания) числом, имеющим ровно такую же сумму цифр, что и то число, при котором замок открывается.

Тогда для открытия замка Васе достаточно найти следующее (в порядке возрастания) число с той же суммой цифр в k-ичной системе счисления, что и текущее, и это число откроет замок.

Но, после месяца использования такого замка, Васе надоело каждый раз решать вручную такую непростую задачу, и он попросил Вас помочь ему.

Требуется написать программу, которая будет по заданному числу m в k-ичной системе счисления находить следующее число (в порядке возрастания) в этой же системе счисления с такой же суммой цифр.

Формат входного файла

В первой строке входного файла задано два натуральных числа k и n ($2 \le k \le 36$; $n \le 100\,000$). Во второй строке задано число m в k-ичной системе счисления, при этом число m состоит из n цифр и, возможно, содержит ведущие нули. Цифрам от 10 до 35 соответствуют соответственно заглавные латинские буквы от A до Z.

Формат выходного файла

В выходной файл выведите ответ на задачу — искомое число в k-ичной системе счисления из n цифр с, возможно, несколькими ведущими нулями. Если Вася где-то ошибся, и следующего такого числа не существует, то выведите в выходной файл единственное слово «Impossible».

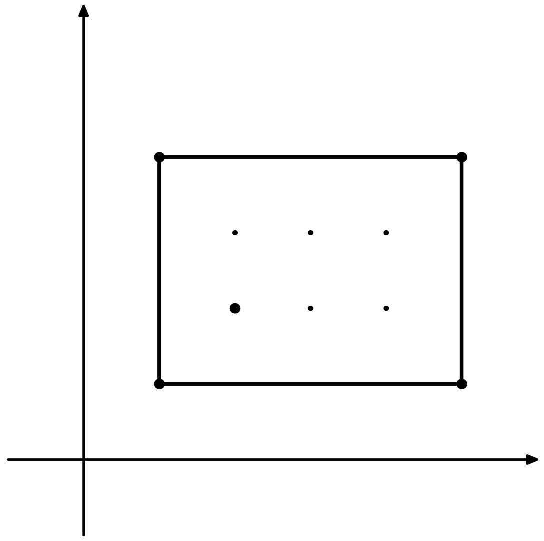
Примеры

code.in	code.out
`10 2` `23`	`32`
`16 2` `FF`	`Impossible`
`26 2` `HP`	`IO`

Задача C. Выпуклая оболочка

Имя входного файла:	`convex.in`
Имя выходного файла:	`convex.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Пусть задано некоторое множество точек на плоскости. *Выпуклая оболочка* — наименьший выпуклый многоугольник, содержащий данные точки.



Необходимо найти число точек с целыми координатами, которые находятся строго внутри выпуклой оболочки заданного множества точек.

Формат входного файла

Первая строка входного файла содержит целое число n ($3 \le n \le 100\,000$). Каждая из последующих n строк описывает одну точку и содержит по два целых числа x_i и y_i ($|x_i|, |y_i| \le 10^9$) — ее координаты. Никакие две точки не совпадают. Среди заданных точек есть три, не лежащие на одной прямой.

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

convex.in	convex.out
5 1 1 1 4 5 4 5 1 2 2	6

Задача D. Даты

Имя входного файла:	`dates.in`
Имя выходного файла:	`dates.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вася конструирует свой собственный автоматический электронный календарь, который будет отображать текущую дату в формате «ДД.ММ.ГГГГ». Для этого ему нужно уметь отображать различные цифры. На данный момент его календарь умеет отображать k различных цифр. Поскольку добавление поддержки каждой новой цифры — процесс очень трудоемкий, то Вася решил, что если его календарь будет отображать довольно большое количество дат, то он будет доволен. Поэтому на данный момент Васю крайне интересует следующий вопрос: сколько дней из данного промежутка его календарь будет отображать дату.

Например, если Васин календарь может отображать только цифры `0`, `2` и `9`, то он сможет отобразить дату «второе февраля 2009 года» и не сможет отобразить дату «третье февраля 2009 года».

Стоит отметить, что программировать Вася умеет неплохо, поэтому он не забывает, что существует такое понятие, как високосный год, в котором в феврале 29 дней. Напомним, что год является високосным, если его номер кратен 4 и при этом не кратен 100, либо кратен 400.

Формат входного файла

Первая строка входного файла состоит из одного целого числа k $(0 \le k \le 10)$. Во второй строке входного файла перечислены через пробел k различных цифр, которые Васин календарь умеет отображать. Следующие две строки содержат описание первой и последней даты интересующего его промежутка в формате «ДД.ММ.ГГГГ».

Формат выходного файла

В выходной файл выведите одно целое число — количество дней в промежутке включая концы, в которые календарь может показать дату.

Примеры

dates.in	dates.out
3 0 2 9 02.02.2009 03.02.2009	1
3 0 2 9 01.02.2009 28.02.2009	4

Задача E. Строки Фибоначчи — 2

Имя входного файла:	`fib2.in`
Имя выходного файла:	`fib2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В математике достаточно часто применяются так называемые рекуррентные соотношения. Обычно они применяются для задания числовых последовательностей, но могут применяться и для задания последовательностей строк.

Одним из примеров строк, задаваемы рекуррентным соотношением являются *строки Фибонначи* $F_0, F_1, \ldots$ Они задаются следующим образом: $F_0 = a$, $F_1 = b$, $F_i = F_{i-2}F_{i-1}, i > 1$. Первые семь строк Фибоначчи выглядят следующим образом: `a`, `b`, `ab`, `bab`, `abbab`, `bababbab`, `abbabbababbab`.

Дима занимается в кружке олимпиадного программирования и интересуется алгоритмами на строках. Недавно он узнал о строках Фибоначчи. Он быстро понял, что их длина с увеличением номера i растет очень быстро, поэтому задача нахождения всех символов строки F_i требует слишком большого объема памяти. В прошлый раз он ограничился задачей нахождения лишь некоторых символов строки F_i. В этот раз он поставил перед собой более сложную задачу — необходимо найти, сколько раз буква «a» встречается среди первых k символов строки F_i.

Напишите программу, решающую эту задачу.

Формат входного файла

Входной файл содержит несколько наборов входных данных. Первая строка входного файла содержит целое число T наборов входных данных ($1 \le T \le 100$). Каждая из последующих T строк описывает один набор входных данных и содержит по два целых числа: n и k ($0 \le n \le 45$, $1 \le k \le |F_n|$, как $|F_n|$ обозначена длина строки F_n, позиции символов в строке нумеруются с единицы).

Формат выходного файла

Выведите в выходной файл T строк, каждая из которых должна содержать одно число — ответ для соответствующего набора входных данных.

Примеры

fib2.in	fib2.out
4 0 1 1 1 3 2 7 7	1 0 1 3

Задача F. Резиновый рюкзак

Имя входного файла:	`knapsack.in`
Имя выходного файла:	`knapsack.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Турист Геннадий готовится к очередному походу. Его старый, видавший виды рюкзак уже износился, да и вещей в него помещается не так уж много. Не стоит и говорить о том, что все возможные задачи об этом рюкзаке он уже прорешал.

Новый рюкзак Геннадия сделан из сверхпрочной резины. Если резину не растягивать, то в этот рюкзак можно положить предметов суммарным объемом не более V_0 кубических сантиметров. Однако, в этот рюкзак можно положить, запихать или в крайнем случае утрамбовать и больше предметов. Проблема в том, что если суммарный объем предметов будет равен $V > V_0$, то на все эти предметы будет действовать давление $P = V - V_0$.

Среди различных предметов, которые Геннадий хочет взять в поход, есть прочные, а есть и хрупкие. Например, внешняя клавиатура для ноутбука Геннадия не сможет вынести того же, что выдерживала проверенная годами палатка. Однако, даже хрупкие предметы, как показывает практика, могут быть ценными в походе. Поэтому Геннадий для каждого предмета, помимо занимаемого им объема v_i, определил его стоимость c_i как меру того, насколько он хочет взять его в поход, а также вычислил максимальное давление p_i, которое он может выдержать.

Таким образом, перед Геннадием встала непростая задача — как выбрать предметы таким образом, чтобы они, находясь все вместе в рюкзаке, смогли выдержать образующееся давление, и при этом стоимость получившегося набора была максимальна?

Геннадий — турист бывалый, поэтому написанная им программа менее, чем за полсекунды справилась с этой задачей. А вы сможете повторить его достижение?

Формат входного файла

В первой строке входного файла находятся два целых числа N $(1 \le N \le 100)$ и V_0 $(0 \le V_0 \le 10^9)$ — число предметов и начальный объем рюкзака.

Следующие N строк содержат тройки целых чисел v_i, c_i и p_i — объем, стоимость и максимальное давление, выдерживаемое i-тым предметом. $1 \le v_i \le 1000$, $0 \le c_i \le 10^6$, $0 \le p_i \le 10^9$.

Формат выходного файла

В первой строке выходного файла выведите число предметов K, которые необходимо взять, и максимальную достигнутую стоимость.

Во второй строке выведите K чисел — номера предметов, которые необходимо взять. Предметы нумеруются, начиная с единицы, в том порядке, в котором они даны во входном файле.

Если существует несколько вариантов ответа, выведите один из них.

Примеры

knapsack.in	knapsack.out
3 10 3 1 2 4 1 2 5 1 2	3 3 1 3 2
3 10 3 1 1 4 1 2 5 1 3	2 2 2 3

Задача G. MP3-плеер

Имя входного файла:	`mp3.in`
Имя выходного файла:	`mp3.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вася недавно купил себе новый MP3-плеер *doPi Shuffle*. Этот плеер отличается от всех других, которые использовал Вася, тем, что у него есть функция «перемешивания» списка проигрываемых песен (плейлиста).

Вася загрузил в плеер n своих любимых песен. До этого он их постоянно слушал на компьютере и очень хорошо запомнил порядок, в котором они были в плейлисте.

Разумеется, плеер переставил песни в случайном порядке. Прослушав плейлист несколько раз на плеере, Вася с удивлением обнаружил, что никакие две, шедшие подряд друг за другом в старом плейлисте, не идут подряд в новом. Он задумался, насколько часто возможна такая ситуация.

Пронумеруем песни числами от 1 до n в том порядке, в котором они идут в плейлисте на компьютере. Напишите программу, которая вычисляет число способов их переставить так, чтобы никакие две песни с последовательными номерами не шли непосредственно друг за другом. Так как это число может быть достаточно большим, выведите его по модулю m.

Формат входного файла

Входной файл содержит два целых числа: n и m ($2 \le n \le 2000$, $2 \le m \le 10^9$).

Формат выходного файла

В выходной файл выведите остаток от деления числа способов переставить песни требуемым образом на m.

Примеры

mp3.in	mp3.out
2 1000000	1
5 1000000	53

Задача H. Оптическое распознавание символов

Имя входного файла:	`ocr.in`
Имя выходного файла:	`ocr.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Всемирно известная компания *Maple* готовится к выпуску новейшего мобильного телефона *myPhone*. Среди прочих возможностей в myPhone будет функция сканирования текста с помощь встроенной фотокамеры.

Телефон должен распознавать n различных символов. Образец символа представляется в виде прямоугольной таблицы размера $w \times h$ ячеек. Каждая ячейка содержит 1, если в этом месте написания символа должны быть чернила, и 0 в обратном случае.

Сканирующая программа с помощью сложных алгоритмов разбивает сфотографированное изображение на прямоугольники размера $w \times h$ пикселей (при этом каждый пиксель считается либо черным, и тогда там стоит 1, либо белым, чему соответствует 0) и сравнивает их с образцами символа.

Для оценки сравнения специальный отдел компании Maple по *Критериям Оценки Шрифтов Конечными Автоматами* разработал величину, называемую *похожестью* двух таблиц. Похожесть — число ячеек, таких что в образце и изображении в соответствующей ячейке наблюдаются одинаковые значения.

Вам необходимо написать программу, которая по набору символов и изображению находила бы символ, наиболее похожий на изображение по критерию *КОШКА*, то есть символ с наибольшей похожестью.

Формат входного файла

В первой строке входного файла три натуральных числа n, w и h ($n, w, h \le 100$). Далее следует n блоков, описывающих образцы символов. Каждый блок состоит из h строк из нулей и единиц по w символов в каждой. Далее следует изображение в аналогичном формате.

Формат выходного файла

В вывходном файле должно быть одно число — номер наиболее похожего символа. Символы нумеруются с единицы в порядке появления во входном файле. Если ответов несколько, выведите любой из них.

Примеры

ocr.in	ocr.out
2 2 2 11 11 00 00 00 01	2
2 2 2 11 11 00 00 01 10	1

Задача I. Налогообложение

Имя входного файла:	taxes.in
Имя выходного файла:	taxes.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Одна малоизвестная, но амбициозная компания разрабатывает масштабный проект, цель которого упростить подсчет бюджета рядового гражданина страны Флатландии. Вы участвуете в этом проекте и вам поручили написать программу, которая вычисляет подоходный налог, то есть налог, взимаемый с получаемого дохода.

Правила вычисления подоходного налога довольно сложны и состоят из $k+1$ части:

- Если доход (X) менее U_1, то налог составляет p_1 процентов, то есть $X \cdot \frac{p_1}{100}$.
- Если X не менее U_1 и меньше U_2, то налог с U_1 составляет p_1, а с оставшейся суммы — p_2. Таким образом суммарный налог составляет $U_1 \cdot \frac{p_1}{100} + (X - U_1) \cdot \frac{p_2}{100}$.
- ...
- Если X не менее U_k, то налог с предыдущих частей остается прежним, а с остатка составляет p_{k+1} процентов, то есть суммарный налог составляет $U_1 \cdot \frac{p_1}{100} + (U_2 - U_1) \cdot \frac{p_2}{100} + \ldots + (X - U_k) \cdot \frac{p_{k+1}}{100}$.

Ваша задача — по данному доходу X вычислить подоходный налог.

Формат входного файла

Первая строка входного файла содержит два целых числа X и k ($0 \le X \le 10^9$, $1 \le k \le 100$). Вторая строка входного файла содержит k целых чисел $U_1, U_2, \ldots, U_k$ ($1 \le U_1 < U_2 < \ldots < U_k \le 10^9$). Третья строка входного файла содержит $k+1$ целое число $p_1, p_2, \ldots, p_k, p_{k+1}$ ($0 \le p_1 \le p_2 \le \ldots \le p_{k+1} \le 100$).

Формат выходного файла

В единственной строке выходного файла выведите подоходный налог, который должен заплатить честный гражданин Флатландии, заработавший X единиц флатландской валюты. Ответ следует выводить ровно с двумя знаками после запятой.

Примеры

taxes.in	taxes.out
1500 2 1000 2000 10 20 30	200.00

Задача J. Обобщенные числа-близнецы

Имя входного файла:	`twins.in`
Имя выходного файла:	`twins.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В теории чисел *простыми числами-близнецами* называют пару таких простых чисел (p, q), что $q-p = 2$. Например, пары $(3, 5)$ и $(11, 13)$ являются парами простых чисел-близнецов. Назовем *обобщенными числами-близнецами* пару простых чисел (p, q), где $q - p = k$, k — некоторое натуральное число. Например, для $k = 4$ пара $(3, 7)$ является парой обобщенных чисел-близнецов.

Существует предположение, что пар простых чисел-близнецов бесконечно много, однако это не доказано. Безусловно, выяснить по заданному k, сколько пар обобщенных близнецов содержит множество всех натуральных чисел, не менее сложная задача, чем аналогичная о простых близнецах.

Ваша же задача несколько проще — выяснить по заданному k, сколько пар обобщенных близнецов содержит множество натуральных чисел от 1 до n.

Формат входного файла

Во первой строке входного файла через пробел заданы два натуральных числа n и k $(1 \le n, k \le 10^6)$.

Формат выходного файла

В выходной файл выведите число пар простых чисел (p, q), таких, что $1 \le p < q \le n$ и $q - p = k$.

Примеры

twins.in	twins.out
17 2	3
100000 1	1
20 7	0