Задача A. Раскраска в три цвета

Имя входного файла:	`coloring.in`
Имя выходного файла:	`coloring.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Сколько способов покрасить n различных домов в три цвета — c_1 домов в красный, c_2 в синий и c_3 в зелёный?

Формат входных данных

В единственной строке ввода заданы через пробел три целых числа c_1, c_2 и c_3 ($0 \leqslant c_1, c_2, c_3 \leqslant 20$, кроме того, их сумма $n = c_1 + c_2 + c_3 \leqslant 20$).

Формат выходных данных

В единственной строке выведите искомое количество способов.

Примеры

coloring.in	coloring.out
1 1 1	6
2 1 1	12
0 1 2	3

Задача B. Строка Фибоначчи

Имя входного файла:	`fibstr.in`
Имя выходного файла:	`fibstr.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Строка Фибоначчи — это строка из нулей и единиц, в которой не встречается двух идущих подряд единиц.

Даны числа n и k. Нужно вывести строку Фибоначчи, которая состоит из n цифр и является k-ой в лексикографическом порядке из таких строк.

Формат входных данных

В первой строке входного файла записаны целые числа n и k через пробел ($0 \leqslant n \leqslant 44$, $0 \leqslant k \leqslant 2 \cdot 10^9$). Гарантируется, что k-ая строка из n символов существует. Строки нумеруются с нуля.

Формат выходных данных

Выведите лексикографически k-ую строку Фибоначчи длины n.

Примеры

fibstr.in	fibstr.out
3 0	000
3 1	001
3 2	010
3 3	100
3 4	101

Задача C. Психотренинг

Имя входного файла:	`psyche.in`
Имя выходного файла:	`psyche.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

На очередном психологическом тренинге n участников сборов по информатике играют в занимательную игру. Участники игры рассаживаются по кругу и получают номера от 1 до n против часовой стрелки. После этого главный психолог отсчитывает против часовой стрелки k-го участника игры, начиная с первого. Этот участник выходит из круга и может идти на ужин. А остальные продолжают участие в тренинге. Главный психолог отсчитывает ещё k участников, начиная со следующего после выбывшего. Участник, который оказался k-ым, тоже покидает тренинг, и так далее.

Участники сборов решили сесть в круг таким образом, чтобы один вредный тип пошёл ужинать последним. Для этого они хотят установить, какой номер он должен для этого получить. Помогите им.

Формат входных данных

Входной файл содержит два целых числа: n и k ($1 \leqslant n, k \leqslant 1\,000\,000$).

Формат выходных данных

Выведите в выходной файл одно число — номер участника, который пойдёт на ужин последним.

Пример

psyche.in	psyche.out
5 3	4

Задача D. Наилучшее приближение

Имя входного файла:	`nearest.in`
Имя выходного файла:	`nearest.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вам даны N целых чисел. Ваша задача — вставить ровно по одному знаку "`+`" или "`-`" между каждой парой соседних таким образом, чтобы сделать значение получившегося выражения максимально близким к заданному числу A.

Формат входных данных

Первая строка входного файла содержит два целых числа: N ($1 \leqslant N \leqslant 10\,000$) и A, которое по модулю не превосходит $10\,000$. Далее следуют N строк, в каждой из которых содержится ровно одно целое число X_i, не превосходящее по модулю $10\,000$. Кроме того, гарантируется, что сумма абсолютных величин всех N чисел также не превосходит $10\,000$.

Формат выходных данных

В первой строке необходимо вывести значение получившегося выражения (которое должно быть настолько близко к A, насколько это возможно). Во второй строке необходимо вывести само выражение, дающее такое значение, в форме $X_1[+|-]X_2[+|-]\ldots X_{N-1}[+|-]X_N$. Если оптимальных решений несколько, то разрешается выводить любое из них.

Пример

nearest.in	nearest.out
3 0 3 -2 1	0 3+-2-1

Задача E. Робот

Имя входного файла:	`robot.in`
Имя выходного файла:	`robot.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Роботу дали задание пройти по лабиринту. Этот лабиринт представляет из себя прямоугольное поле размера $m \times n$ клеток, в котором каждая клетка либо занята сплошной стеной, либо свободна. Каждая клетка описывается парой координат (x, y), где $1 \leqslant x \leqslant m$ и $1 \leqslant y \leqslant n$. Особенности конструкции робота таковы, что он из клетки (x, y) может попасть только в клетки $(x-1, y)$, $(x+1, y)$ и $(x, y+1)$, конечно, если они свободны, но не может уменьшить свою координату по y и перейти на клетку $(x, y-1)$.

Робот хочет провести в лабиринте как можно больше времени, так как после прохождения лабиринта его наверняка заставят заниматься чем-то более тяжёлым. Однако, если он остановится хоть на секунду или просто замедлит свое движение, могут возникнуть подозрения, что он сломался, и тогда ему грозит попадание на свалку металлолома. Поэтому робот хочет выбрать путь, который бы имел наибольшую длину, и пройти по нему с постоянной скоростью.

Задача робота усложняется тем, что ему нельзя возвращаться на клетку, в которой он уже побывал — иначе баг в программе, которая записывает его путь, заставит его крутиться на одном месте, а о последствиях такой ошибки страшно даже подумать!

Помогите роботу найти самый длинный путь по лабиринту, не имеющий самопересечений и такой, что в нём не уменьшается координата y. Путь может начинаться в любой свободной клетке с $y = 1$ и заканчиваться в любой свободной клетке с $y = n$; можно считать, что весь лабиринт огорожен сплошной стеной, и выходить за его пределы нельзя.

Формат входных данных

В первой строке ввода заданы целые числа m и n через пробел ($1 \leqslant m, n \leqslant 100$). В последующих n строках содержится по m символов в каждой; j-ый символ i-ой из этих строк равен «`X`» (икс большое), если соответствующая клетка занята стеной, и «`.`» (точка), если она свободна.

Формат выходных данных

Если пути с указаными свойствами не существует, выведите в выходной файл число `-1`. В противном случае в первую строку выведите число посещённых роботом клеток k, а в последующие k строк по паре целых чисел x_i и y_i через пробел — координаты клеток пути в порядке их посещения. Число k должно быть максимально; если оптимальных ответов несколько, разрешается вывести любой из них.

Примеры

robot.in	robot.out
2 2	4 1 1 2 1 2 2 1 2
3 4 .X. ... X.X ..X	6 3 1 3 2 2 2 2 3 2 4 1 4
1 1 X	-1

Задача F. Лекция

Имя входного файла:	lecture.in
Имя выходного файла:	lecture.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

> Образование — это то, что остаётся, когда забываешь всё, чему учился в школе.
>
> Эйнштейн

Константин Игоревич читает завтра лекцию по алгебре. Сегодня же ему надо решить, какие из теорем, которые он будет рассказывать, следует доказать. С одной стороны, лекция не может длиться более T минут, так что на доказательство вообще всех теорем времени, скорее всего, не хватит. С другой стороны, хочется, чтобы лекция оказалась для студентов как можно полезнее.

Полезность лекции определяется как сумма полезностей всех доказанных теорем, в то время как *полезность теоремы* — какое-то заранее известное для каждой теоремы число. Теоремы делятся на три класса по сложности, и их доказательство занимает одну, две или три минуты в зависимости от класса.

Конечно же, прилежные студенты заранее выучили все определения и формулировки, так что всё время лекции можно потратить на доказательство теорем.

Формат входных данных

В первой строке ввода заданы через пробел два числа N и T ($1 \leqslant N \leqslant 100\,000$, $1 \leqslant T \leqslant 300\,000$) — количество теорем и продолжительность лекции (в минутах). Далее идут N строк, в каждой два числа t_i и a_i ($1 \leqslant t_i \leqslant 3$, $0 \leqslant a_i \leqslant 10^9$), обозначающих класс сложности i-й теоремы (то есть, время в минутах, требуемое на её доказательство) и её полезность.

Формат выходных данных

Выведите одно число — максимальную полезность завтрашней лекции Константина Игоревича.

Примеры

lecture.in	lecture.out
3 3 1 3 2 10 2 14	17
3 4 2 5 2 10 3 14	15

Задача G. Гиперкуб

Имя входного файла:	cube.in
Имя выходного файла:	cube.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Гиперкуб — это обобщение понятия трёхмерного куба на N измерений. Нуль-мерным гиперкубом является точка, одномерным — отрезок, двумерным — квадрат. В общем же случае N-мерный гиперкуб — это правильный N-мерный многогранник, каждая из $2 \cdot N$ граней которого является $(N-1)$-мерным гиперкубом. Например, для $N = 2$ квадрат — это правильный многоугольник, каждая из $2 \cdot 2 = 4$ сторон которого — отрезок, то есть

одномерный гиперкуб. Отметим, что N-мерный гиперкуб имеет 2^N вершин.

Старшеклассник Петя долго разбирался, что же такое гиперкуб, но наконец понял, как этот объект устроен, и ему настолько понравилось, что он даже придумал свою собственную игру на гиперкубе. Игра заключается в следующем.

Рассмотрим N-мерный единичный гиперкуб. Расположим его таким образом, чтобы одна из вершин находилась в начале координат — точке $(0, 0, \ldots, 0)$ в N-мерном пространстве, а для любой из остальных вершин каждая координата равнялась бы нулю или единице. В каждой из 2^N вершин запишем по целому неотрицательному числу. Игрок начинает свой путь в начале координат. За один ход можно переместиться из текущей вершины по любому ребру при условии, что сумма координат новой вершины строго больше суммы координат старой. Игра заканчивается, когда игрок попадает в вершину $(1, 1, \ldots, 1)$, имеющую максимальную сумму координат — N. Результат игры — сумма чисел во всех посещённых игроком вершинах. Цель игры — пройти по гиперкубу таким образом, чтобы эта сумма (количество очков за игру) оказалась как можно больше.

Петя довольно быстро понял, что между двумя вершинами гиперкуба A и B ребро есть тогда и только тогда, когда все координаты этих вершин $(A_1, A_2, \ldots, A_N)$ и $(B_1, B_2, \ldots, B_N)$ совпадают, кроме одной, которая равна нулю у одной из вершин (скажем, A) и единице у другой (B). Поскольку при этом $A_1 + A_2 + \ldots + A_N + 1 = B_1 + B_2 + \ldots + B_N$, то по такому ребру можно перемещаться из A в B, но не наоборот. Однако, сыграв в свою игру, Петя не может с уверенностью сказать, является ли полученная им сумма максимальной или можно на данном гиперкубе сыграть по-другому и набрать больше очков.

Напишите программу, которая по данному гиперкубу находит максимальную сумму, которую можно получить, сыграв в эту игру.

Формат входных данных

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 10$) — размерность гиперкуба. В следующих 2^N строках содержится по одному числу в каждой; в $(k+2)$-ой строке записано C_k ($0 \leqslant C_k \leqslant 1000$) — число в вершине с номером k.

Номер вершины вычисляется так: вершина A с координатами $(A_1, A_2, \ldots, A_N)$ имеет номер, равный $A_1 \cdot 2^{N-1} + A_2 \cdot 2^{N-2} + \ldots + A_{N-1} \cdot 2 + A_N$, то есть координаты просто интерпретируются как двоичная запись номера вершины. В этой нумерации начальная вершина имеет номер 0, а конечная — номер $2^N - 1$.

Формат выходных данных

В выходной файл выведите одно число — максимальную сумму, которую можно получить при игре на данном гиперкубе.

Пример

cube.in	cube.out
3 1 2 3 4 5 6 7 8	21

Пояснение к примеру

Наш маршрут таков:

- вершина 0 (число 1, координаты $(0, 0, 0)$) — начальная
- вершина 4 (число 5, координаты $(1, 0, 0)$)
- вершина 6 (число 7, координаты $(1, 1, 0)$)
- вершина 7 (число 8, координаты $(1, 1, 1)$) — конечная

Наше количество очков: $1 + 5 + 7 + 8 = 21$.

Любой другой маршрут с соблюдением правил игры даёт меньшее количество очков.

Задача H. Редакционное расстояние

Имя входного файла:	`editdist.in`
Имя выходного файла:	`editdist.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В информатике *редакционным расстоянием* между двумя строками называется минимальное количество добавлений, удалений и замен символов, при помощи которых можно из одной строки получить другую. К примеру, редакционное расстояние между строками "`ab`" и "`ab`" равно нулю, так как строки равны между собой безо всяких изменений; расстояние между строками "`short`" и "`ports`" равно трём: в слове "`short`" нужно удалить из

начала букву 's', заменить 'h' на 'p' и добавить в конец букву 's'. Редакционное расстояние также называют *расстоянием Левенштейна*.

Найдите редакционное расстояние между двумя заданными строками.

Формат входных данных

В первой строчке входного файла задана одна строка, во второй — другая. Длины обеих строк — от 1 до 100 символов, включительно.

Формат выходных данных

В выходной файл выведите единственное число — редакционное расстояние между двумя заданными строками.

Примеры

editdist.in	editdist.out
ab ab	0
short ports	3

Задача I. Шаблоны

Имя входного файла:	patterns.in
Имя выходного файла:	patterns.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Многие операционные системы используют шаблоны для ссылки на группы объектов: файлов, пользователей и т. д. Ваша задача — реализовать простейший алгоритм проверки шаблонов для имён файлов.

В этой задаче алфавит состоит из маленьких букв английского алфавита и точки ('.'). Шаблоны могут содержать произвольные символы алфавита, а также два специальных символа: '?' и '*'. Знак вопроса ('?') соответствует ровно одному произвольному символу. Звёздочка '*' соответствует подстроке произвольной длины (возможно, нулевой). Символы алфавита, встречающиеся в шаблоне, отображаются на ровно один такой же символ в проверяемой строчке. Строка считается подходящей под шаблон, если символы шаблона можно последовательно отобразить на символы строки таким образом, как описано выше. Например, строчки "ab", "aab" и "beda." подходят под шаблон "*a?", а строчки "bebe", "a" и "ba" — нет.

Формат входных данных

Первая строка входного файла определяет шаблон P. Вторая строка S состоит только из символов алфавита. Её необходимо проверить на соответствие шаблону. Длины обеих строк не превосходят 10 000. Строки могут быть пустыми — будьте внимательны!

Формат выходных данных

Если данная строка подходит под шаблон, выведите "YES". Иначе выведите "NO".

Примеры

patterns.in	patterns.out
k?t*n kitten	YES
k?t?n kitten	NO

Задача J. Два шаблона

Имя входного файла:	patterns2.in
Имя выходного файла:	patterns2.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Многие операционные системы используют шаблоны для ссылки на группы объектов: файлов, пользователей и т. д. Ваша задача — найти строку минимально возможной длины, которая подходит под два заданных шаблона.

Алфавит в этой задаче состоит из маленьких букв латинского алфавита и точки ('.'). Шаблоны могут содержать любые символы алфавита, а также специальные символы '?' и '*'. Под '?' подходит любой символ алфавита, а под '*' — любая строка символов алфавита (возможно, пустая). Под символы алфавита, встречающиеся в шаблоне, подходят только такие же символы алфавита. Строка считается подходящей под шаблон, если символы шаблона можно последовательно отобразить в строку вышеуказанным способом. Например, строки "ab", "aab" и "beda." подходят под шаблон "*a?", а строки "bebe", "a" и "ba" — нет.

Формат входных данных

Входной файл состоит из одного или нескольких тестов. Первая строка входного файла — это количество тестов в нем.

Каждый тест состоит из двух строк, содержащих шаблоны P_1 и P_2. Длина любого из шаблонов не превосходит 100 символов.

Формат выходных данных

Для каждого из тестов ответ задается одной строкой:

- Если строка, подходящая под оба шаблона, существует, выведите такую строку минимально возможной длины (если таких несколько, разрешается выводить любую).
- В противном случае выведите строку "NO".

Пример

patterns2.in	patterns2.out
2 *k*tt*n* *i*e* haha hihi	kitten NO

Задача K. Выбор вершин взвешенного дерева

Имя входного файла:	selectw.in
Имя выходного файла:	selectw.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дан граф, являющийся деревом. В вершинах графа написаны целые числа. Множество вершин графа называется *допустимым*, если никакие две вершины этого множества не соединены ребром.

Рассмотрим все допустимые множества вершин графа. Для каждого такого множества вычислим сумму чисел, написанных в его вершинах. Какова максимальная из этих сумм?

Формат входных данных

Граф в этой задаче задан в виде *корневого дерева*. В графе выделена вершина — *корень дерева*. Для каждой вершины i, не являющейся корнем, задан номер вершины-предка p_i в корневом дереве. Дерево, заданное таким образом, состоит из рёбер $i - p_i$ для всех вершин i, кроме корня.

В первой строке входного файла записано целое число n — количество вершин в графе ($1 \leqslant n \leqslant 100$). В следующих n строках задан граф. В i-й из этих строк записаны через пробел два целых числа p_i и q_i; здесь p_i — номер вершины-предка i-ой вершины, а q_i — число, записанное в этой вершине. Для корня дерева $p_i = 0$; для всех остальных вершин $1 \leqslant p_i \leqslant n$. Числа q_i не превосходят по модулю 10 000.

Гарантируется, что заданный во входном файле граф является деревом.

Формат выходных данных

В первой строке выходного файла выведите одно число — максимальную сумму чисел в допустимом множестве.

Примеры

selectw.in	selectw.out
5 0 1 1 2 1 3 2 4 3 5	10
6 5 8 6 0 5 -1 1 1 0 3 1 2	8

На рисунке показаны графы, заданные в примерах. В каждом графе выделено допустимое множество с максимальной суммой чисел в вершинах.