Задача A. Архиватор

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

Вася решил покорить рынок лучших архиваторов мира. Совсем недавно он придумал очень нетривиальную идею для сжатия текста из маленьких латинских букв. А именно, он решил, что можно хранить текст как последовательность команд. Команды бывают двух типов:

- «`c`»: дописать к текущей строке символ c.
- «`i k`»: дописать к текущей строке k символов один за другим. При этом первый дописываемый символ совпадает с символом i текущей строки, второй с символом $i + 1$ и так далее, k-ый добавляемый символ совпадает с символом $i + k - 1$. Гарантируется, что i не превосходит текущей длины строки.

Например последовательность команд «`a, b, 1 3`» кодирует строку «`ababa`», а последовательность команд «`a, 1 3, b, 3 3`» кодирует строку «`aaaabaab`».

На хранение команды первого типа Васе требуется 1 байт, а второго типа 5 байт. К сожалению, пока Вася умеет только по командам восстановить исходную строку, а наоборот не умеет. Вам предлагается помочь бедному Васе в покорении архиваторного рынка. Найдите последовательность команд, которая архивирует заданную строку указанным способом, при этом потратив как можно меньше байт на ее хранение.

Формат входного файла

Во входном файле вам задана строка s из строчных латинских букв длиной не более 4000 символов.

Формат выходного файла

В первой строке выходного файла вы должны вывести количество байт, которое потребуется для хранения последовательности команд и количество команд в последовательности. На следующих строках выведите саму последовательность, по одной команде на строке. Если команда первого типа, то выведите просто букву, иначе выведите два числа: позиция символа (символы нумеруются начиная с единицы) в строке s, начиная с которого надо начать копирование, и количество символов, которое надо скопировать.

Примеры

archiver.in	archiver.out
abcdqwertyqwertyu	16 12 a b c d q w e r t y 5 6 u

Задача B. Get the Duck to the Sink

Имя входного файла:	`getduck.in`
Имя выходного файла:	`getduck.out`
Ограничение по времени:	4 seconds
Ограничение по памяти:	256 Mebibytes

Как-то профессор Налейпиво заметил, что один из студентов на его лекции уделяет слишком много внимания мобильному телефону. Подкравшись сзади (а несмотря на большие размеры, профессор Налейпиво умеет незаметно подкрадываться), профессор обнаружил смягчающее вину студента обстоятельство — тот не отправлял SMS-ки, а увлечённо играл в следующую игру.

Есть поле размером $N \times M$ ячеек и несколько (возможно ноль) стен между ячейками. Одна из ячеек является *стоком*, в то время как одна из оставшихся занята *уткой*. Ваша задача привести *утку* в *сток*. Единственный способ передвижения *утки*, доступный вам — это *скольжение*, что подразумевает, что вы можете толкнуть *утку* в одном из четырёх направлений, и она будет *скользить* в нем, пока не упрется в стену. Вы не можете толкнуть *утку* снова, пока она не остановится. Уровень считается пройденным, если *утка* останавливается в *стоке*. Если *утка* просто *проскользнула* через *сток*, не остановившись, уровень не считается пройденным.

Профессор остановил лекцию и попросил студента создать несколько своих уровней. Он расчертил на доске уровень, нанёс стены, поместил *сток*, и теперь

студенту надо разместить где-то *утку*. Профессор выбрал несколько мест, куда бы он хотел поместить её, и предложил студенту описать алгоритм прохождения уровня. Студент быстро понял, что это ловушка — среди них есть такие, начав игру из которых, довести *утку* до *стока* невозможно. А сумеете ли это сделать Вы?

Ваша задача — имея размеры поля, позицию стен и *стока*, а также выбранные позиции для *утки*, найти среди выбранных позиций такие, начав игру из которых пройти уровень возможно.

Формат входного файла

Первая строка содержит два числа N и M — размеры поля.

Далее следует $2N+1$ строк, каждая по $2M+1$ символов, где $2k$-ый символ $2i$-ой стоки либо пробел, если ячейка пуста, либо `S` если ячейка содержит сток либо `D` если ячейка входит в список выбранных для размещения утки.

$(2k+1)$-ый символ $2i$-ой строки либо пробел, если $k > 0$ и $k < M$ и нет стены между ячейками (k,i) и $(k+1,i)$; либо `|` в противном случае.

$2k$-ый символ $(2i+1)$-ой строки либо пробел, если $i > 0$ и $i < N$ и нет стены между ячейками (k,i) и $(k,i+1)$; либо `-` в противном случае.

$(2k+1)$-ый символ $(2i+1)$-ой строки всегда `+`.

$1 \leqslant N \leqslant 1000$

$1 \leqslant M \leqslant 1000$

Формат выходного файла

Выведите поле из входного файла, заменив буквы `D` на пробел в ячейках, начиная с которых, нельзя пройти уровень.

Пример

getduck.in	getduck.out
5 5 +-+-+-+-+-+ | | + +-+ + + + | S D D| + + + + + + | D | | + + + + + + | | | | + + + +-+ + | | +-+-+-+-+-+	+-+-+-+-+-+ | | + +-+ + + + | S D | + + + + + + | D | | + + + + + + | | | | + + + +-+ + | | +-+-+-+-+-+

Задача C. Треугольник

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

Треугольник с положительной площадью расположили на плоскости так, что все его вершины оказались в точках с целочисленными координатами. Длины двух его сторон равны $\sqrt{A}$ и $\sqrt{B}$, соответственно. Вычислите максимальную площадь, которую может иметь такой треугольник.

Формат входного файла

На единственной строке входного файла записаны целые числа A и B $(1 \leqslant A, B \leqslant 2 \cdot 10^9)$.

Формат выходного файла

На единственной строке выходного файла запишите ответ (вещественное число с ровно одним знаком после точки), либо -1.0, если такого треугольника не существует.

Примеры

triangle.in	triangle.out
1 1	0.5
3 7	-1.0