Задача A. Лазурит

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В мире Minecraft Стив построил автоматическое хранилище лазурита из N сундуков, расположенных по кругу. Сундуки пронумерованы числами от 1 до N по часовой стрелке.

Вместимость i-го сундука составляет a_i единиц лазурита. Изначально каждый сундук заполнен полностью, то есть в i-м сундуке находится ровно a_i единиц лазурита.

Соседние сундуки соединены специальными воронками. Каждую минуту все воронки срабатывают одновременно:

- весь лазурит из сундука i перемещается в сундук $i + 1$ для каждого $1 \leqslant i < N$;
- весь лазурит из сундука N перемещается в сундук 1.

Все перемещения происходят одновременно. Иными словами, лазурит, полученный сундуком в течение текущей минуты, будет перемещён дальше только на следующей минуте.

Если после перемещения в i-м сундуке оказывается больше a_i единиц лазурита, то весь лишний лазурит выпадает в лаву и безвозвратно исчезает.

Стив хочет узнать, сколько всего единиц лазурита останется в системе после каждой из первых N минут.

Формат входных данных

В первой строке находится одно целое число N — количество сундуков ($1 \leqslant N \leqslant 5 \cdot 10^5$).

Во второй строке находятся N целых чисел $a_1, a_2, \ldots, a_N$ — вместимости сундуков ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите N строк.

В i-й строке выведите суммарное количество молока, оставшееся во всех вёдрах после i минут.

Примеры

стандартный ввод	стандартный вывод
6 2 2 2 1 2 1	8 7 6 6 6 6
8 3 8 6 4 8 3 8 1	25 20 17 14 12 10 8 8
10 9 9 10 10 6 8 2 1000000000 1000000000 100	2000000053 1000000054 56 49 42 35 28 24 20 20

Замечание

Изменение количества лазурита в сундуках в первом примере:

Момент времени	Лазурит в сундуках	Всего лазурита
Изначально	$[2, 2, 2, 1, 2, 1]$	10
После 1-й минуты	$[1, 2, 2, 1, 1, 1]$	8
После 2-й минуты	$[1, 1, 2, 1, 1, 1]$	7
После 3-й минуты	$[1, 1, 1, 1, 1, 1]$	6
После 4-й минуты	$[1, 1, 1, 1, 1, 1]$	6

Начиная с третьей минуты в каждом сундуке находится ровно одна единица лазурита, поэтому его общее количество больше не изменяется.

Задача B. Лосяш и шоколадка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Лосяш очень любит шоколад. У него есть плитка размером $n \times m$ разбитая на клетки. Каждый день Лосяш съедает одну клетку, потому что в шоколадке нет пары клеток с одинаковым вкусом, и Лосяш хочет попробовать их все. Каждая клека съедается ровно 1 раз.

Лосяш очень любит делиться шоколадом со своими друзьями, поэтому после каждого дня он хочет знать, сколько существует прямоугольных кусочков, которые можно отрезать от оставшейся части плитки, не разрезая ни одной клетки. Кусочек считается прямоугольным, если его границы проходят по линиям сетки (то есть он состоит из целых клеток, а его стороны параллельны сторонам плитки). Только такими кусочками Лосяш может делиться со своими друзьями.

Помогите Лосяшу — для каждого дня выведите количество таких кусочков, полностью состоящих из ещё не съеденных клеток.

Формат входных данных

Первая строка содержит два целых числа n и m ($1 \leqslant n, m \leqslant 500$).

Далее следуют $n \cdot m$ строк, каждая из которых содержит два числа r и c ($1 \leqslant r \leqslant n, 1 \leqslant c \leqslant m$) — координаты клетки, которую Лосяш съедает в этот день.

Гарантируется, что каждая клетка встречается ровно один раз.

Формат выходных данных

Выведите $n \cdot m$ строк. В i-й строке выведите число прямоугольных кусочков, полностью состоящих из клеток, которые ещё не съедены после первых i дней.

Пример

стандартный ввод	стандартный вывод
2 2 1 1 2 1 1 2 2 2	5 3 1 0

Замечание

- После первого дня съедена клетка $(1, 1)$. Целы остаются 3 клетки и 2 прямоугольника 1×2 (во второй строке и во втором столбце) $\rightarrow$ всего 5.
- После второго дня (съедены $(1, 1)$ и $(2, 1)$) цел только второй столбец – в нём 3 прямоугольника.
- После третьего дня остаётся одна клетка $\rightarrow$ 1 прямоугольник.
- После четвёртого – ни одной $\rightarrow$ 0.

Задача C. Маленькие машинки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	128 мегабайт

У Егора есть большой прямоугольный коврик-автодром для маленьких машинок. Коврик разбит на $h \times w$ квадратных участков размером 1×1. По исправному участку машинки могут свободно проехать, а повреждённый участок использовать нельзя.

Егор хочет выделить на коврике прямоугольную площадку для гонки. Он умеет установить один небольшой мостик, поэтому внутри выбранной площадки может находиться не более одного повреждённого участка. Все остальные участки площадки должны быть исправны.

Найдите максимальную площадь прямоугольной площадки, которую может выбрать Егор. Стороны площадки должны проходить по границам клеток коврика.

Формат входных данных

В первой строке записаны два целых числа h и w ($1 \leqslant h, w \leqslant 2000$) — высота и ширина коврика.

В следующих h строках записано по w символов:

- символ `.` обозначает исправный участок;
- символ `#` обозначает повреждённый участок.

Формат выходных данных

Выведите одно целое число — максимальную площадь прямоугольной площадки, содержащей не более одного повреждённого участка.

Пример

стандартный ввод	стандартный вывод
4 5 #.#.. ....# ..#.. ....#	12

Замечание

В примере можно выбрать прямоугольник из строк со второй по четвёртую и столбцов с первого по четвёртый. Его площадь равна $3 \cdot 4 = 12$, и внутри находится ровно один повреждённый участок.

Задача D. Сложная задача

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	768 мегабайт

У Олега сломалась мышка и теперь он вынужден играть в Geometry Dash по памяти. Из реквизита у Олега есть только ручка и рулон клетчатой туалетной бумаги шириной m. Длину рулона можно считать неограниченной.

Теперь Олег хочет нарисовать на рулоне уровень для любимой игры. Он планирует разместить на уровне n партов, пронумерованных от 1 до n. Парты должны следовать на рулоне слева направо. Если в какой-то момент нарисовать следующий парт невозможно — его обязательно нужно перенести на следующую строку. Также переносы разрешено делать в произвольном месте рулона — на усмотрение Олега, однако каждый парт должен быть написан целиком на одной строке.

В силу специфичности вкусов Олега, каждый парт имеет форму прямоугольника. Разные парты могут иметь разную высоту и ширину. А именно, i-й парт имеет ширину w_i и высоту h_i клеток.

Чтобы уровень выглядел красиво, Олег выравнивает все парты, записанные в одной строке, по нижней границе строки. Таким образом, высота строки равняется высоте самого высокого парта в строке. Суммарной высотой уровня назовем суммарную высоту всех строк рулона.

Теперь, из соображений экономии, Олега интересует, какую наименьшую высоту может иметь его уровень? Поскольку Олегу нужно делать контест, он просит о помощи Вас.

Формат входных данных

В данной задаче ввод устроен необычным образом — для некоторых тестов часть входного файла вам придется генерировать самостоятельно.

В первой строке вводятся три числа n, m, k ($1 \leqslant n \leqslant 10^7$, $1 \leqslant m \leqslant 10^{11}$, $1 \leqslant k \leqslant \min(500\,000, n)$) — общее количество партов, ширина рулона; количество партов, высота и ширина для которых будут введены далее.

В следующих k строках вводится по три целых числа i_j w_{i_j}, h_{i_j} ($1 \leqslant i_j \leqslant n$, $1 \leqslant w_{i_j}, h_{i_j} \leqslant m$). Первое число — номер одного из партов Олега. Следующие два числа — ширина и высота этого парта соответственно.

Гарантируется, что индексы i_j возрастают с каждой следующей строкой.

В последней строке вводятся 4 числа a, b, c, d ($1 \leqslant a, c \leqslant \min(10\,000, m)$, $0 \leqslant b, d \leqslant m$) — вспомогательные параметры, которые отвечают за генерацию остальных входных данных.

Для тех партов, номера которых не были перечислены во входных данных, высоту и ширину придётся генерировать самостоятельно. А именно, для любого i, не перечисленного во входных данных, $w_i = ((w_{i-1} \cdot a + b) \ mod \ m) + 1$, $h_i = ((h_{i-1} \cdot c + d) \ mod \ m) + 1$. Гарантируется, что $i = 1$ присутствует во входных данных (соответственно, размеры всех партов определяются однозначно).

Формат выходных данных

Выведите одно число — минимальную высоту уровня шириной m, в который получится уместить все парты.

Примеры

стандартный ввод	стандартный вывод
3 3 3 1 1 2 2 2 3 3 1 3 2 2 2 2	5
4 3 1 1 2 3 1 1 1 1	7

Замечание

В первом примере из условия оптимальным вариантом размещения будет тот, при котором первый парт находится на отдельной строке, а второй и третий — на следующей. Таким образом, суммарная высота получается равной $2 + 3 = 5$ клеткам.

Во втором примере из условия прямоугольники имеют вид [(2, 3), (1, 2), (3, 1), (2, 3)]. Третий прямоугольник обязательно займет целую строку. Четвертый парт, таким образом, займет свою отдельную строчку, а первый и второй парты сгруппируются. Получившиеся строки будут вместе занимать $3 + 1 + 3 = 7$ клеток в высоту.

Задача E. Тренировка по футболу

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Сергей проводит необычную футбольную тренировку. Все игроки его команды стоят в один ряд вдоль боковой линии поля. Позиции в ряду пронумерованы от 1 до количества игроков, начиная от ворот.

Во время тренировки происходят события трёх типов:

1. Перед игроком, занимающим первую позицию, в ряд встаёт несколько новых футболистов;
2. После последнего игрока в ряд встаёт несколько новых футболистов;
3. Сергей проводит очередной этап тренировки и изменяет показатели всех игроков.

В каждый момент времени позиции футболистов нумеруются от ворот, начиная с 1. После добавления новых игроков в начало ряда номера всех ранее стоявших футболистов увеличиваются.

Для каждого футболиста Сергей хранит показатель усталости. Обозначим через A_i показатель футболиста, который в данный момент занимает позицию i.

Показатели усталости изменяются по следующим правилам:

- В начале тренировки у всех футболистов $A_i = 0$. У каждого нового футболиста в момент его появления показатель также равен нулю.
- Во время очередного этапа тренировки Сергей выбирает целые **положительные** числа b и s. После этого к показателю футболиста на позиции i прибавляется величина $b + (i - 1) \cdot s$.

После каждого события Сергей выбирает футболиста с минимальным показателем усталости. Если минимальное значение имеют несколько игроков, Сергей выбирает среди них того, кто находится ближе всего к воротам.

Помогите Сергею после каждого события определить позицию выбранного футболиста и его показатель усталости.

Формат входных данных

Первая строка содержит два целых числа n и m ($1 \leqslant n \leqslant 10^9$, $1 \leqslant m \leqslant 300,000$) — начальное количество футболистов в ряду и количество событий соответственно.

Следующие m строк содержат описания событий. Каждое событие имеет один из трёх следующих видов:

- «1 k» ($1 \leqslant k \leqslant 10^9$) — перед текущим первым футболистом в ряд встают k новых игроков;
- «2 k» ($1 \leqslant k \leqslant 10^9$) — после текущего последнего футболиста в ряд встают k новых игроков;
- «3 b s» ($1 \leqslant b, s \leqslant 10^9$) — Сергей проводит этап тренировки с параметрами b и s.

Гарантируется, что в любой момент времени количество футболистов в ряду не превышает 10^9, а все значения A_i не превышают 10^{18}.

Формат выходных данных

После каждого из m событий выведите два целых числа j и A_j — позицию ближайшего к воротам футболиста среди имеющих минимальный показатель усталости и значение этого показателя.

Пример

стандартный ввод	стандартный вывод
1 8 1 1 3 1 1 3 1 1 2 1 2 1 3 1 1 2 1 3 1 5	1 0 1 1 1 2 3 0 3 0 1 3 5 0 1 4

Замечание

Рассмотрим пример, в котором изначально в ряду находится один футболист.

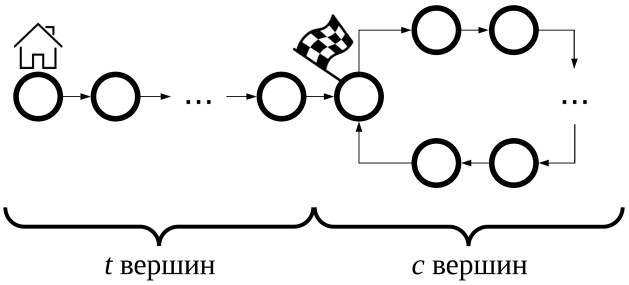
- Изначально его показатель усталости равен нулю, поэтому ряд имеет вид $[0]$.
- После того как перед ним встаёт ещё один футболист, ряд принимает вид $[0, 0]$.
- После этапа тренировки с параметрами $b = 1$, $s = 1$ показатели становятся равны $[1, 2]$.
- После ещё одного этапа с такими же параметрами получается $[2, 4]$.
- После добавления одного футболиста в конец ряда получается $[2, 4, 0]$.
- После добавления ещё одного футболиста в конец ряда получается $[2, 4, 0, 0]$.
- После этапа тренировки с параметрами $b = 1$, $s = 1$ получается $[3, 6, 3, 4]$.
- После добавления одного футболиста в конец ряда получается $[3, 6, 3, 4, 0]$.
- После этапа тренировки с параметрами $b = 1$, $s = 5$ получается $[4, 12, 14, 20, 21]$.

Задача F. Двое в лесу

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Это интерактивная задача.

Игровое поле для одной малоизвестной кабо-вердианской игры представляет из себя **ориентированный** граф, состоящий из двух частей. Одна из этих частей — цикл из c вершин. Вторая часть — цепь из t вершин, причём из последней вершины этой цепи проведено ребро в одну из вершин цикла — назовём её интересной. Игровое поле держат от вас в секрете, так что никто кроме ведущего не знает ни t, ни c.



В начале игры игровые фишки всех десяти игроков, для удобства пронумерованных цифрами от 0 до 9, располагаются в начале цепи, в вершине у домика (см. рис.). Затем не более q раз ведущий будет одновременно передвигать в следующие вершины фишки игроков, высказавших такое желание на этом ходу, а после этого объявлять, фишки каких игроков находятся в одной вершине, а каких — в разных. Обратите внимание, что у каждой вершины игрового поля есть только одна следующая, при этом каждая кроме стартовой и интересной является следующей только для одной вершины. Стартовая вершина не является следующей ни для какой вершины, а интересная вершина является следующей для последней вершины цепочки и для одной из вершин цикла. Ведущему так понравилась идея не сообщать игрокам t и c, что и q он решил тоже не объявлять.

Цель игры — перевести игровые фишки всех игроков в интересную вершину, то есть в ту, которая отмечена финишным флагом (см. рис.). Игроки не представляют, как можно выиграть в такой игре не зная ни c, ни t, ни q, но, к счастью для них, они ещё и ваши друзья. Помогите им: скоординируйте их действия так, чтобы победить.

Протокол взаимодействия

Для удобства пронумеруем игроков целыми числами от 0 до 9 включительно.

Для того, чтобы отдать команды перемещения игрокам, выведите «`next`» а затем через пробел номера игроков, которым необходимо отдать команды. Например, чтобы отдать команду перемещения игрокам с номерами 0, 2, 5 и 9 выведите «`next 0 2 5 9`». На каждом ходу обязательно перемещать хотя бы одного из игроков.

В качестве ответа проверяющая программа выдаст сначала число k, а потом 10 цифр, разбитых на k групп, разделённых пробелами. Игроки, соответствующие цифрам, находящимся в одной группе, находятся в одной вершине. Игроки, соответствующие цифрам, находящимся в разных группах, находятся в разных вершинах. Цифры в каждой группе следуют в порядке возрастания.

Например ответ проверяющей программы «`2 05 12346789`» означает, что игроки с номерами 0 и 5 находятся в одной вершине, а все остальные игроки в одной и той же, но другой вершине. Ответ проверяющей программы «`4 01 567 234 89`» означает, что игроки находятся в четырёх различных вершинах: в первой находятся игроки с номерами 0 и 1, во второй — игроки с номерами 5, 6 и 7, в третьей — игроки с номерами 2, 3 и 4, а в четвёртой — игроки с номерами 8 и 9.

Если вы превысите лимит на количество ходов или нарушите протокол взаимодействия, то вместо информации о местоположении игроков вашей программе на вход будет передана строка «`stop`».

Считав такую строку, ваша программа должна немедленно завершить работу, иначе она может получить вердикт тестирования, отличный от настоящего.

После того, как все игроки соберутся в интересной вершине необходимо вывести «`done`» и завершить работу программы.

После вывода каждой строки сбрасывайте буфер вывода — для этого используйте `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Тесты	Дополнительные ограничения	
	$(t + c)$	q
1	$(t + c) \leqslant 65$	$6600 \leqslant q \leqslant 10\,000$
2–10	$(t + c) \leqslant 65$	$6600 \leqslant q \leqslant 10\,000$
11–20	$(t + c) \leqslant 100$	$(t + c)^2 \leqslant q \leqslant 10\,000$
21–61	$(t + c) \leqslant 1000$	$4 \cdot (t + c) \leqslant q \leqslant 10\,000$
62–100	$(t + c) \leqslant 1000$	$3 \cdot (t + c) \leqslant q \leqslant 10\,000$

Пример

стандартный ввод	стандартный вывод
2 05 12346789 3 246789 135 0 3 246789 0 135 3 246789 0 135 2 135 0246789 1 0123456789	next 0 5 next 0 1 3 next 2 3 0 1 4 5 6 7 8 9 next 9 8 7 6 5 4 3 2 1 0 next 0 1 3 5 next 1 3 5 done

Замечание

В условии в примере взаимодействия вводимые и выводимые данные расположены для удобства восприятия в хронологическом порядке, при реальном взаимодействии никакие «лишние» переводы строк возникать не должны.

Задача G. Галактические гусеницы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Копатыч выращивает вдоль длинного прямого забора N грядок. Грядки пронумерованы от 1 до N в порядке их расположения вдоль забора. Расстояния между соседними грядками могут быть различными.

На i-й грядке растут растения общей ценностью v_i. Чем больше v_i, тем важнее для Копатыча спасти эту грядку.

Однажды Копатыч заметил, что к огороду приближается огромное количество гусениц. Чтобы защитить растения, он подготовил S электронных отпугивателей.

Каждый отпугиватель можно установить на любой грядке или не устанавливать вовсе. Установленный на некоторой грядке отпугиватель защищает от гусениц все грядки, расположенные на расстоянии не более K метров от него. В частности, грядка, на которой установлен отпугиватель, также защищена.

Копатыч хочет разместить отпугиватели так, чтобы суммарная ценность защищённых грядок была максимальной.

Кроме того, Копатыч хочет, чтобы все защищённые грядки образовывали один непрерывный участок огорода. Иными словами, если защищены грядки с номерами a и b, где $a < b$, то каждая грядка с номером от a до b также должна быть защищена.

Помогите Копатычу определить, на каких грядках следует установить отпугиватели.

Формат входных данных

В первой строке даны три целых числа N, S и K — количество грядок, количество имеющихся отпугивателей и радиус действия каждого отпугивателя в метрах соответственно.

Во второй строке даны $N - 1$ целых чисел $d_1, d_2, \ldots, d_{N-1}$. Число d_i обозначает расстояние в метрах между грядками i и $i + 1$.

В третьей строке даны N целых чисел $v_1, v_2, \ldots, v_N$. Число v_i обозначает ценность растений на грядке i.

Гарантируется, что $1 \leqslant S \leqslant N \leqslant 10^6; 1 \leqslant K \leqslant 10^{12}; 1 \leqslant d_i \leqslant 10^6$ для всех $1 \leqslant i < N$;
$1 \leqslant v_i \leqslant 10^6$ для всех $1 \leqslant i \leqslant N$.

Формат выходных данных

В первой строке выведите целое число T — количество отпугивателей, которые следует установить. Должно выполняться условие

$$0 \leqslant T \leqslant S.$$

Во второй строке выведите T целых чисел — номера грядок, на которых следует установить отпугиватели. Номера можно выводить в любом порядке.

Разрешается устанавливать несколько отпугивателей на одной и той же грядке.

Суммарная ценность защищённых грядок должна быть максимально возможной. Кроме того, между любыми двумя защищёнными грядками не должно быть незащищённой грядки.

Если существует несколько оптимальных вариантов размещения отпугивателей, выведите любой из них.

Пример

стандартный ввод	стандартный вывод
6 2 7 10 4 7 18 11 5 8 2 4 8 12	2 3 5

Замечание

В первом примере выгодно установить отпугиватели на грядках 3 и 5.

Отпугиватель на грядке 3 защищает грядки 2, 3 и 4, а отпугиватель на грядке 5 защищает грядку 5. Таким образом, все защищённые грядки образуют непрерывный участок с номерами от 2 до 5.

Установить отпугиватели на грядках 3 и 6 нельзя: в этом случае грядка 5 останется незащищённой и окажется между защищёнными грядками.

Задача H.

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Стив решил построить огромную пирамиду для маяка. Для строительства он выбрал прямоугольный участок мира Minecraft, разбитый на квадратные блоки.

Участок состоит из n строк и m столбцов. Для каждой клетки известна высота находящегося в ней столба блоков.

Основание пирамиды должно иметь прямоугольную форму размером a столбцов на b строк. Внутри основания Стив хочет оставить прямоугольное подземное хранилище размером c столбцов на d строк.

Основание пирамиды и хранилище должны занимать целые клетки, а их стороны должны быть параллельны сторонам участка.

Хранилище должно полностью находиться внутри основания. Кроме того, между хранилищем и каждой из четырёх сторон основания должна оставаться полоса шириной хотя бы в одну клетку.

Высоты столбов блоков в клетках, занятых хранилищем, изменять нельзя.

Во всех остальных клетках основания Стив может свободно перемещать блоки между столбами. Он хочет выровнять эти столбы так, чтобы после строительства они имели одинаковую высоту.

Таким образом, итоговая высота основания будет равна средней высоте всех столбов основания, не принадлежащих хранилищу.

Помогите Стиву выбрать положение основания пирамиды и положение хранилища внутри него так, чтобы итоговая высота основания была максимально возможной.

Формат входных данных

В первой строке находятся шесть целых чисел m, n, a, b, c и d:

- m — количество столбцов участка;
- n — количество строк участка;
- a — ширина основания пирамиды;
- b — высота основания пирамиды;
- c — ширина хранилища;
- d — высота хранилища.

Следующие n строк содержат по m целых чисел.

Число $h_{i,j}$ задаёт высоту столба блоков, расположенного в строке i и столбце j.

Первая строка описывает верхнюю строку участка, а последняя — нижнюю.

Гарантируется, что $3 \leqslant m, n \leqslant 1000$, $3 \leqslant a \leqslant m$, $3 \leqslant b \leqslant n$, $1 \leqslant c \leqslant a - 2$ и $1 \leqslant d \leqslant b - 2$.

Высота каждого столба является целым числом от 1 до 100 включительно.

Формат выходных данных

В первой строке выведите два целых числа — номер столбца и номер строки, в которых находится левый верхний угол основания пирамиды.

Во второй строке выведите два целых числа — номер столбца и номер строки, в которых находится левый верхний угол хранилища.

Строки и столбцы нумеруются начиная с единицы.

Если существует несколько оптимальных вариантов размещения, разрешается вывести любой из них.

Пример

стандартный ввод	стандартный вывод
8 5 5 3 2 1 1 5 10 3 7 1 2 5 6 12 4 4 3 3 1 5 2 4 3 1 6 6 19 8 1 1 1 3 4 2 4 5 6 6 3 3 3 2 2 2	4 1 6 2

Замечание

В первом примере левый верхний угол основания пирамиды находится в столбце 4 и строке 1. Левый верхний угол хранилища находится в столбце 6 и строке 2.

Основание занимает прямоугольник размером 5×3, а хранилище — прямоугольник размером 2×1.

Между хранилищем и каждой из четырёх сторон основания остаётся хотя бы одна клетка.

Задача I. Набор ЛКШ 2027

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дирекция ЛКШ объявила конкурс на должность преподавателя!

В качестве вступительного испытания каждому кандидату выдали длинную строку s, состоящую только из символов `(`, `)` и `?`.

Кандидат должен заменить каждый символ `?` либо на открывающую скобку `(`, либо на закрывающую скобку `)`. Результат кандидата определяется длиной самой длинной подпоследовательности получившейся строки, являющейся правильной скобочной последовательностью. Победителем конкурса станет кандидат, которому удастся сделать эту длину максимально возможной.

Подпоследовательностью строки называется строка, получающаяся из исходной удалением некоторого, возможно нулевого, количества символов. Например, строки `abc`, `ac`, `bcc` и `abbcc` являются подпоследовательностями строки `abbcc`, а строки `cb` и `ba` не являются. Пустая строка является подпоследовательностью любой строки.

Последовательность круглых скобок называется правильной в следующих случаях:

1. Пустая последовательность является правильной.
2. Если последовательность является правильной, то последовательность, полученная добавлением открывающей скобки в её начало и закрывающей скобки в её конец, также является правильной.
3. Если две последовательности являются правильными, то их конкатенация также является правильной.

Например, последовательности `()()` и `((()))()` являются правильными, а последовательности `)(()`, `(((((` и `())` правильными не являются.

Один из участников конкурса очень хочет стать преподавателем ЛКШ, но не может самостоятельно справиться со вступительным испытанием. Помогите ему заменить все символы `?` так, чтобы его результат был максимально возможным.

Формат входных данных

В единственной строке содержится строка s, состоящая только из символов `(`, `)` и `?`. Длина строки удовлетворяет ограничению $1 \leqslant |s| \leqslant 10\,000\,000$.

Формат выходных данных

Выведите строку, полученную из s заменой каждого символа `?` на `(` или `)`, такую, что длина самой длинной правильной скобочной подпоследовательности этой строки максимальна.

Примеры

стандартный ввод	стандартный вывод
`?)))?)`	`()))()`
`)(???)(`	`)((())(`

Замечание

В первом примере из полученной строки можно выделить правильную скобочную подпоследовательность длины 4: `()()`.

Во втором примере из полученной строки можно выделить правильную скобочную подпоследовательность длины 4: `(())`. Возможны и другие оптимальные ответы.