

## Problem A. Anti-Lines (Division 1 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 3 секунды  
Memory limit: 256 мегабайт

В этой задаче речь пойдет об игре “Lines”. В классической версии есть квадратное поле размером  $9 \times 9$  клеток. Каждая клетка может быть либо пустой, либо содержать шарик одного из 7 цветов. Каждый ход компьютер выставляет три шарика случайных цветов в случайные пустые клетки. После чего Игрок может переместить один любой шарик на поле в какую-то допустимую пустую клетку. Если 5 или более шариков одного цвета расположить в виде вертикальной, горизонтальной или диагональной линии, то эти шарики уничтожаются, а игрок получает очки и дополнительный ход.

В этой задаче все происходит немного иначе: мы взломали компьютер и на каждом ходу можем выбирать пустые клетки, в которые выставляются новые шарики и цвета этих шариков. Каждый ход выставляется ровно по два новых шарика. Игрок ничего не передвигает (поскольку мы уже руководим действиями компьютера). Цель — совершать такие ходы, чтобы в момент первого уничтожения шариков количество одновременно уничтожившихся шариков было максимальным. То есть, можно совершить несколько ходов, собрать какую-нибудь удачную конструкцию, а потом одним ходом ее уничтожить. Делать дополнительные уничтожения, очищая при этом карту, нельзя.

Два шарика выставляются на поле одновременно. После этого любой шарик, который принадлежит вертикальной, горизонтальной или диагональной линии хотя бы из 5 шариков одного цвета, мгновенно уничтожается. Если на поле меньше двух пустых клеток, то ход совершить нельзя.

### Input

Ровно 9 строк по 9 символов каждая — текущая позиция в игре. Каждый символ — это либо цифра от 1 до 7, означающая цвет шарика в текущей клетке, либо символ ‘.’, если клетка пустая. Гарантируется, что на поле нет ни одной линии из 5 или более шариков одного цвета.

### Output

В первой строке два числа  $D$  и  $M$  — максимальное количество шариков, которые можно уничтожить одним ходом и количество ходов, которые приводят к этому уничтожению. Далее  $2M$  строк по две на каждый ход. Каждая строка должна содержать три числа  $r_i$ ,  $c_i$  и  $color_i$  ( $1 \leq r_i, c_i \leq 9$ ,  $1 \leq color_i \leq 7$ ) — номер строки и столбца пустой клетки, на которую надо выставить шарик цвета  $color_i$ . Оба шарика одного хода выставляются одновременно. Ровно  $D$  шариков должны уничтожиться в процессе первого уничтожения.

Если ни одного шарика уничтожить нельзя, то  $D$  должно быть равно 0.

## Examples

standard input	standard output
723134552	25 9
2421.2456	6 7 2
353...442	3 4 2
14...2..4	5 8 2
23.2....4	5 7 2
4....1..1	5 3 2
1222..112	6 3 2
3621..124	3 6 2
631211777	8 5 2
	7 5 2
	6 5 2
	5 5 2
	2 5 2
	3 5 2
	4 8 2
	4 7 2
	4 4 2
	5 6 2
	4 5 2

## Problem B. Big Fellow (Division 1 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 секунды  
Memory limit: 256 мегабайт

На день рождения Вам подарили игру  $nm$ -нашки. Она представляет собой таблицу, состоящую из  $n$  строк и  $m$  столбцов. В каждой её ячейке лежит карточка с натуральным числом. Когда все карточки лежат на своём месте, таблица выглядит так:

1	2	3	...	$m$
$m+1$	$m+2$	$m+3$	...	$2m$
$2m+1$	$2m+2$	$2m+3$	...	$3m$
...	...	...	...	...
$(n-1)m+1$	$(n-1)m+2$	$(n-1)m+3$	...	$nm$

Как-то Вы пришли домой и увидели, что с таблицей кто-то поигрался. Она стала выглядеть так:

1	$n+1$	$2n+1$	...	$(m-1)n+1$
2	$n+2$	$2n+2$	...	$(m-1)n+2$
3	$n+3$	$2n+3$	...	$(m-1)n+3$
...	...	...	...	...
$n$	$2n$	$3n$	...	$mn$

Вам ужасно не понравился новый вид таблицы, и Вы решили уложить все карточки на место. Для этого Вы решили выполнить последовательность операций. Каждая операция состоит из следующих действий:

- 1) Поднять какую-либо карточку в воздух
- 2) Найти её правильное место в таблице
- 3) Если это место свободно, положить туда карточку, находящуюся в воздухе
- 4) Иначе, поменять местами карточку, лежащую на этом месте, с карточкой, находящейся в воздухе, и вернуться к шагу 2.

При этом Вы решили поставить себе условие, что в процессе укладывания каждая из  $nm$  карточек должна побывать в воздухе хотя бы один раз. Это означает, что даже если карточка уже лежит на своём месте, Вам придётся потратить операцию на её поднятие и укладывание обратно.

Какое минимальное количество таких операций Вам необходимо выполнить для приведения таблицы к исходному виду?

### Input

В единственной строке файла находятся два натуральных числа  $n$  и  $m$  такие, что  $n \cdot m \leq 10^{14}$

### Output

Вывести единственное число - минимальное количество необходимых операций.

### Examples

standard input	standard output
4 4	10
2 3	3

## Problem C. Concave Polygon (Division 1 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 5 секунд  
Memory limit: 256 мегабайт

Дан простой многоугольник без самопересечений и самокасаний. Требуется разрезать его диагоналями на минимальное количество выпуклых многоугольников. В качестве вершин выпуклых многоугольников можно использовать только вершины исходного многоугольника. Каждая вершина исходного многоугольника должна встретиться в качестве вершины хотя бы одного выпуклого многоугольника. Никакие три вершины любого выпуклого многоугольника не должны лежать на одной прямой. В качестве сторон выпуклых многоугольников можно использовать стороны исходного многоугольника и его диагонали. Каждый из выпуклых многоугольников должен лежать внутри исходного многоугольника. Объединение всех выпуклых многоугольников должно составлять исходный многоугольник. Никакие два выпуклых многоугольника не должны иметь положительную площадь пересечения.

### Input

Первая строка содержит число  $N$  ( $3 \leq N \leq 150$ ) — количество вершин исходного многоугольника. Каждая из последующих  $N$  строк содержит по два целых числа  $x_i$  и  $y_i$  ( $-10\,000 \leq x_i, y_i \leq 10\,000$ ) — координаты  $i$ -ой вершины многоугольника в порядке обхода против часовой стрелки. Гарантируется, что многоугольник невырожденный.

### Output

Одно число — минимальное количество выпуклых многоугольников, на которые можно разрезать исходный многоугольник.

### Examples

standard input	standard output
7 3 -1 3 1 2 1 3 3 0 0 3 -3 2 -1	3

## Problem D. Dead Points (Division 1 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 секунды  
Memory limit: 256 мегабайт

На плоскости есть  $N$  точек. Точки бывают двух цветов: черные и белые. С точками происходит такой процесс: каждую итерацию любая точка, которая видит хотя бы одну точку противоположного цвета, умирает. Точка  $A$  видит точку  $B$ , если отрезок  $AB$  не содержит других живых точек множества. В течение одной итерации все необходимые точки умирают одновременно. Если не осталось живых точек какого-нибудь цвета, процесс заканчивается.

Определить, сколько итераций будет продолжаться процесс, и точки какого цвета выживут в результате.

### Input

В первой строке число  $N$  ( $1 \leq N \leq 100\,000$ ) — количество точек на плоскости. Далее  $N$  строк по три целых числа в каждой:  $x_i, y_i$  и  $color_i$  ( $0 \leq x_i, y_i \leq 1\,000\,000, color_i \in \{0, 1\}$ ) — координаты и цвет  $i$ -ой точки. 0 означает белый, 1 — черный. Никакие две точки не имеют одинаковые координаты.

### Output

Слово “Draw”, если в результате умерли все точки, “Black”, если черные точки выжили, или “White”, если выжили белые. В той же строке через пробел вывести количество итераций процесса.

### Examples

standard input	standard output
3 0 0 0 1 0 1 0 1 1	Draw 1
3 0 0 0 1 0 1 2 0 1	Black 1

## Problem E. Equilateral Polygon

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мебибайт

Требуется построить выпуклый многоугольник, длины сторон которого равны 1, а все углы при вершинах различны.

### Input

Единственное число  $N$  ( $5 \leq N \leq 100$ ) — количество вершин многоугольника.

### Output

Ровно  $N$  строк.  $i$ -ая строка должна содержать два вещественных числа  $x_i$  и  $y_i$  (рекомендуется с 16 знаками после запятой), которые являются координатами  $i$ -ой вершины многоугольника.  $x_i$  и  $y_i$  по модулю не должны превышать  $10^3$ . Вершины можно выводить в порядке обхода по или против часовой стрелки. Длина каждой стороны не должна отличаться от 1 более чем на  $10^{-8}$ . Любые два угла должны отличаться друг от друга как минимум на  $10^{-4}$  радиан. Все углы должны быть в пределах  $[10^{-4}, \pi - 10^{-4}]$ .

Гарантируется, что хотя бы один многоугольник, удовлетворяющий условию, существует.

### Examples

standard input	standard output
5	1.0000000000000000 1.0000000000000000 2.0000000000000000 1.0000000000000000 2.3093247548365476 1.9509564638012140 1.5003804226047008 2.5388417139563542 0.6912907818637410 1.9511564638048255

## Problem F. Flawless Numbers

Input file: *standard input*  
Output file: *standard output*  
Time limit: 5 секунд  
Memory limit: 256 мегабайт

Число  $N$  называется безупречным, если  $\frac{\sigma(N)}{N} = \frac{A}{B}$ , где  $\sigma(N)$  — сумма всех делителей числа  $N$ .

Для заданных  $A$  и  $B$  найдите все безупречные числа от 1 до  $10^{14}$  включительно.

### Input

Два натуральных числа  $A$  и  $B$  ( $1 \leq A, B \leq 100$ ,  $A$  — нечетное (внезапно),  $2 \leq \frac{A}{B} \leq 5$ ).

### Output

Первая строка должна содержать число  $K$  — количество безупречных чисел. Далее  $K$  строк, по одному безупречному числу в каждой. Каждое безупречное число должно встречаться ровно один раз. Числа можно выводить в любом порядке. Гарантируется, что  $K$  не превосходит 1000.

### Examples

standard input	standard output
5 2	3 24 91963648 10200236032

## Problem G. Grep

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мебибайт

Дана строка  $S$  длины  $N$  из символов 'a', 'b' и 'c' и строка  $T$  длины  $N$  из символов '.' и '\*'. За одну операцию можно циклически сдвинуть строку  $S$  на один символ вправо или влево. Требуется за минимальное количество операций добиться того, чтобы на всех позициях, которые в строке  $T$  отмечены символом '\*', каждая из букв 'a', 'b' и 'c' побывала хотя бы один раз.

### Input

Первая строка содержит строку  $S$ , которая состоит только из символов 'a', 'b' и 'c'. Каждый из символов 'a', 'b' и 'c' встречается в строке  $S$  хотя бы один раз. Вторая строка содержит строку  $T$ , состоящую только из символов '.' и '\*'.  $3 \leq |S| = |T| \leq 1\,000\,000$ .

### Output

Одно число — минимальное количество операций сдвига, необходимых, чтобы на каждой позиции, отмеченной символом '\*', каждая из букв 'a', 'b' и 'c' побывала хотя бы раз.

### Examples

standard input	standard output
abacaba ...**..	3



## Problem H. Hash It!

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мегабайт

Изначально есть пустая строка  $S$ . Над строкой требуется производить операции двух типов:

- Добавить к строке  $S$  справа символ  $c$ .
- Удалить из строки  $S$  один символ справа.

После каждой операции требуется определить количество различных непустых подстрок строки  $S$ .

### Input

Одна строка  $Q$  ( $1 \leq |Q| \leq 100\,000$ ), означающая последовательность операций со строкой  $S$ . Если  $i$ -ый символ строки  $Q$  является строчной буквой английского алфавита, тогда в качестве  $i$ -ой операции нужно добавить эту букву в конец строки  $S$ , если  $i$ -ый символ строки  $Q$  является символом '-', тогда в качестве  $i$ -ой операции нужно удалить из строки  $S$  один символ справа. Никакие другие символы в строке  $Q$  не встречаются. Гарантируется, что перед любой операцией удаления строка  $S$  не является пустой.

### Output

Вывести ровно  $|Q|$  строк.  $i$ -ая строка должна содержать одно число — количество различных подстрок строки  $S$  после применения первых  $i$  операций.

### Examples

standard input	standard output
aba-caba	1
	3
	5
	3
	6
	9
	12
	17

## Problem I. Interactive Problem 2

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 секунды  
Memory limit: 8 мегабайт

*Обратите внимание! Эта задача имеет специальное ограничение по памяти.*

Дано корневое бинарное дерево. Все вершины дерева пронумерованы числами от 1 до  $N$ . У каждой вершины может быть либо 2 сына, либо 0. Требуется обойти дерево поиском в ширину или поиском в глубину — по вашему усмотрению, и вывести номера вершин в порядке выбранного вами обхода, а также сумму расстояний от корня до всех вершин. Корень всегда имеет номер 1.

Если вы выводите вершины в порядке обхода поиска в ширину, то выводить номера вершин в пределах одного слоя следует слева направо. Если вы выводите вершины в порядке обхода поиска в глубину, то номер вершины следует выводить в момент первого посещения, а при обходе сначала идти в левого сына, потом в правого.

Более формально: поставим каждой вершине  $v_i$  в соответствие строку  $S_i$ ,  $j$ -ым символом которой является 'l', если  $j$ -ое ребро на пути из корня в вершину  $v_i$  ведёт в левого сына и 'r', если в правого. Количество символов в строке  $S_i$  равно расстоянию от вершины до корня.

При обходе поиском в ширину вершина  $v_i$  должна быть посещена раньше вершины  $v_j$  тогда и только тогда, когда:

- $S_i$  короче  $S_j$ , если  $S_i$  и  $S_j$  имеют разную длину
- $S_i$  лексикографически меньше  $S_j$ , если  $S_i$  и  $S_j$  имеют одинаковую длину

При обходе поиском в глубину вершина  $v_i$  должна быть посещена раньше вершины  $v_j$  тогда и только тогда, когда  $S_i$  лексикографически меньше  $S_j$ .

### Input

Первая строка входного файла содержит число  $N$  ( $1 \leq N \leq 1\,000\,000$ ) — количество вершин в дереве. Каждая из следующих  $N$  строк содержит пару чисел  $l_i, r_i$  ( $1 \leq l_i, r_i \leq N$ ) — номера левого и правого сыновей вершины  $i$ , или два нуля, если вершина  $i$  является листом.

### Output

В первой строке выведите “DFS”, если вы обходите дерево поиском в глубину, или “BFS”, если поиском в ширину. В последующих  $N$  строках выведите номера вершин в порядке выбранного вами обхода. В последней строке выведите сумму расстояний от корня до всех вершин.

### Examples

standard input	standard output
5	DFS
3 4	1
0 0	3
0 0	4
2 5	2
0 0	5
	6

## Problem J. Jolly Dolls

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мебибайт

Матрешки — деревянные куклы, вложенные одна в другую. Каждая кукла имеет собственный внешний объем  $out_i$  и объем  $in_i$  пустого места внутри нее. Одну куклу можно вложить в другую, если внешний объем первой меньше (в этот момент вы спрашиваете себя “меньше???”). Да, вам не показалось, именно меньше) объема пустого места внутри второй. В каждую куклу можно непосредственно вложить не более одной куклы. Но эта вложенная кукла может содержать в себе другую куклу и так далее. То есть, если две куклы фактически находятся внутри третьей, то одна из них находится внутри другой.

Матрешки вложены друг в друга оптимально, если суммарный объем пустого пространства внутри всех матрешек минимален. Требуется найти количество способов оптимально вложить матрешки друг в друга.

### Input

Первая строка содержит число  $N$  ( $1 \leq N \leq 100\,000$ ) — количество матрешек в наборе.  $i$ -ая из следующих  $N$  строк содержит два целых числа  $out_i$  и  $in_i$  ( $1 \leq in_i < out_i \leq 1\,000\,000\,000$ ) — внешний объем и объем пустого места внутри  $i$ -ой матрешки.

### Output

Количество способов оптимального расположения матрешек по модулю  $1\,000\,000\,007$ .

### Examples

standard input	standard output
3	1
5 4	
4 2	
3 2	

## Problem K. Block Shuffling (Division 2 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мегабайт

Есть тождественная перестановка из  $2^p$  элементов  $(1, 2, 3, \dots, 2^p)$ . Требуется произвести с ней последовательность операций такого вида:

Разбить перестановку на непрерывные непересекающиеся блоки длиной  $2^k$ . То есть, элементы на позициях  $(1, \dots, 2^k)$  формируют первый блок, элементы на позициях  $(2^k + 1, \dots, 2 \cdot 2^k)$  — второй блок, элементы  $(2 \cdot 2^k + 1, \dots, 3 \cdot 2^k)$  — третий блок, и так далее. После этого соседние блоки поменять местами. То есть, первый блок меняется со вторым, третий с четвертым и так далее.

Дано множество позиций, требуется сказать, какие элементы перестановки находятся на этих позициях после выполнения всех операций.

### Input

Первая строка содержит три числа  $N$ ,  $K$  и  $Q$  ( $N = 2^p$ ,  $0 \leq p \leq 21$ ,  $1 \leq K, Q \leq 100\,000$ ) — количество элементов перестановки, количество операций над перестановкой и количество интересующих позиций.

Далее следуют  $K$  строк по одному числу  $k_i$  в каждой ( $0 \leq k_i < p$ ), означающих, что нужно менять соседние блоки по  $2^{k_i}$  элементов.

Далее следуют  $Q$  строк по одному числу  $q_i$  в каждой ( $1 \leq q_i \leq N$ ) — номера интересующих позиций.

### Output

Вывести  $Q$  строк, в  $i$ -ой из которых номер элемента перестановки на позиции  $q_i$ .

### Examples

standard input	standard output
8 3 3	8
0	6
1	2
2	
1	
3	
7	

## Problem L. Digit Permutation (Division 2 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мегабайт

Найти число в системе счисления по основанию  $B$ , удовлетворяющее следующим ограничениям:

- Каждая цифра от 1 до  $B - 1$  встречается ровно один раз (цифра 0 не встречается).
- Для любого  $i$ :  $1 \leq i \leq B - 1$ , если взять первые  $i$  цифр этого числа, не меняя порядок, то полученное число делится на  $i$ .

### Input

Число  $B$  ( $2 \leq B \leq 36$ ) — основание системы счисления.

### Output

Если требуемого числа не существует, вывести "No", иначе вывести одно число в системе счисления по основанию  $B$ . Цифры от 0 до 9 задаются символами '0' - '9', Цифры от 10 до 35 задаются символами 'A' - 'Z'. Если решений несколько выведите любое из них.

### Examples

standard input	standard output
10	381654729
18	No

## Problem M. Mortal Points (Division 2 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мегабайт

На прямой есть  $N$  точек. Точки бывают двух цветов: черные и белые. С точками происходит такой процесс: каждую итерацию любая точка, которая видит хотя бы одну точку противоположного цвета, умирает. Точка  $A$  видит точку  $B$ , если отрезок  $AB$  не содержит других живых точек множества. В течение одной итерации все необходимые точки умирают одновременно. Если не осталось живых точек какого-нибудь цвета, процесс заканчивается.

Определить, сколько итераций будет продолжаться процесс, и точки какого цвета выживут в результате.

### Input

В первой строке число  $N$  ( $1 \leq N \leq 100\,000$ ) — количество точек на прямой. Далее  $N$  строк по два целых числа в каждой:  $x_i$  и  $color_i$  ( $0 \leq x_i \leq 1\,000\,000$ ,  $color_i \in \{0, 1\}$ ) — координата и цвет  $i$ -ой точки. 0 означает белый, 1 — черный. Никакие две точки не имеют одинаковые координаты.

### Output

Слово “Draw”, если в результате умерли все точки, “Black”, если черные точки выжили, или “White”, если выжили белые. В той же строке через пробел вывести количество итераций процесса.

### Examples

standard input	standard output
3 0 0 1 1 2 1	Black 1

## Problem N. Non-convex Polygon (Division 2 Only!)

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 секунда  
Memory limit: 256 мегабайт

Дан простой многоугольник без самопересечений и самокасаний. Требуется разрезать его диагоналями на минимальное количество выпуклых многоугольников. В качестве вершин выпуклых многоугольников можно использовать только вершины исходного многоугольника. Каждая вершина исходного многоугольника должна встретиться в качестве вершины хотя бы одного выпуклого многоугольника. Никакие три вершины любого выпуклого многоугольника не должны лежать на одной прямой. В качестве сторон выпуклых многоугольников можно использовать стороны исходного многоугольника и его диагонали. Каждый из выпуклых многоугольников должен лежать внутри исходного многоугольника. Объединение всех выпуклых многоугольников должно составлять исходный многоугольник. Никакие два выпуклых многоугольника не должны иметь положительную площадь пересечения.

### Input

Первая строка содержит число  $N$  ( $3 \leq N \leq 15$ ) — количество вершин исходного многоугольника. Каждая из последующих  $N$  строк содержит по два целых числа  $x_i$  и  $y_i$  ( $-10\,000 \leq x_i, y_i \leq 10\,000$ ) — координаты  $i$ -ой вершины многоугольника в порядке обхода против часовой стрелки. Гарантируется, что многоугольник невырожденный.

### Output

Одно число — минимальное количество выпуклых многоугольников, на которые можно разрезать исходный многоугольник.

### Examples

standard input	standard output
7 3 -1 3 1 2 1 3 3 0 0 3 -3 2 -1	3