A. Легкоатлетический манеж НГУ

Имя входного файла:	`manege.in`
Имя выходного файла:	`manege.out`
Ограничение по времени:	2 *секунды*
Ограничение по памяти:	256 *мегабайт*
Максимальная оценка:	100 *баллов*

Рядом со спортивным комплексом НГУ решили построить легкоатлетический манеж с M одинаковыми прямолинейными беговыми дорожками. Они будут покрыты полосами из синтетического материала тартана. На складе имеются N полос тартана, длины которых равны 1, 2, …, N метров соответственно (i-я полоса имеет длину i метров).

Было решено использовать все полосы со склада, что определило длину дорожек манежа. Полосы тартана должны быть уложены без перекрытий и промежутков. Разрезать полосы на части нельзя. Полосы укладываются вдоль дорожек, ширина полосы тартана совпадает с шириной беговой дорожки.

Требуется написать программу, которая определяет, можно ли покрыть всем имеющимся материалом M дорожек, и если это возможно, то распределяет полосы тартана по дорожкам.

Формат входных данных

Во входном файле содержатся два целых числа, разделенных пробелом: M — количество дорожек и N — количество полос тартана ($1 \le M \le 1000$, $1 \le N \le 30000$).

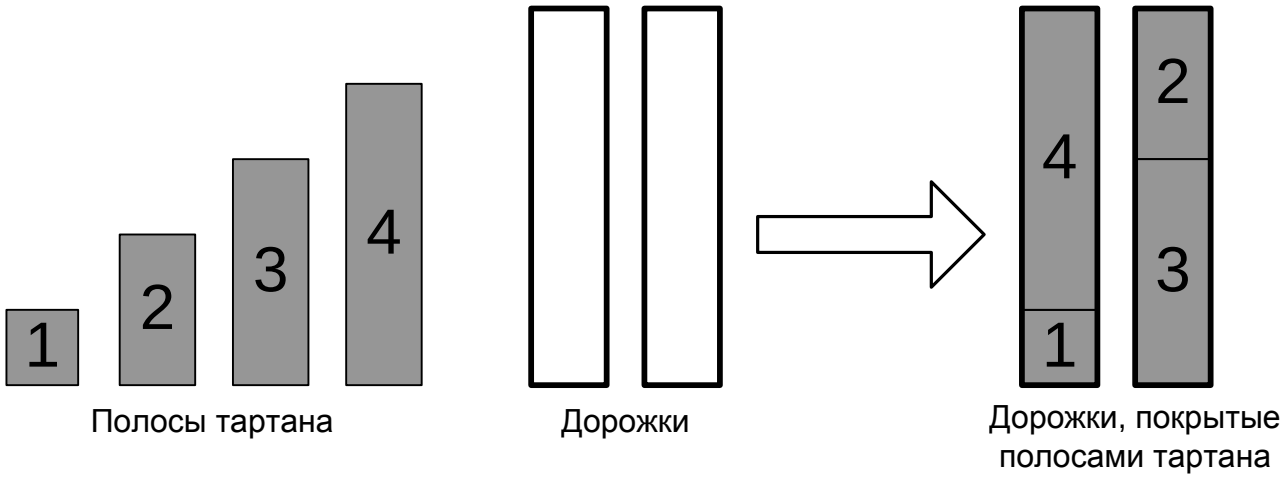
Формат выходных данных

В случае, если распределить имеющиеся полосы тартана на M дорожек одинаковой длины невозможно, то в выходной файл выведите слово «`NO`».

В противном случае, в первую строку выведите слово «`YES`». В последующих M строках дайте описание использованных полос для каждой дорожки в следующем формате: сначала целое число t — количество полос на дорожке, затем t целых чисел — длины полос, которые составят эту дорожку. Если решений несколько, можно вывести любое из них.

Примеры входных и выходных данных

manege.in	**manege.out**
2 4	YES 2 1 4 2 3 2
3 4	NO



B. Граффити на заборе

Имя входного файла:	`fence.in`
Имя выходного файла:	`fence.out`
Ограничение по времени:	2 *секунды*
Ограничение по памяти:	256 *мегабайт*
Максимальная оценка:	100 *баллов*

Около прямолинейного забора, состоящего из N одинаковых бетонных плит, проводится конкурс граффити, в котором участвуют M граффити-художников. Художники должны разрисовать все плиты своими произведениями за наименьшее возможное время.

Плиты пронумерованы числами от 1 до N, граффити-художники имеют номера от 1 до M. Первоначально i-й граффити-художник находится около плиты с заданным номером p_i. Каждому художнику требуется b минут на разрисовывание любой плиты. Каждую плиту должен разрисовать ровно один граффити-художник.

В начале работы, а также после разрисовывания любой плиты граффити-художник может перейти к любой неразрисованной плите. Время перемещения граффити-художника от любой плиты к соседней с ней одинаково и равно a минут. Таким образом, чтобы перейти от плиты с номером i к плите с номером j художнику требуется $a \cdot |i - j|$ минут.

Требуется написать программу, которая поможет участникам конкурса разрисовать все плиты за минимальное возможное время.

Формат входных данных

В первой строке входного файла указаны числа N — количество плит в заборе и M — количество граффити-художников ($1 \le N, M \le 100000$). Во второй строке заданы два целых числа: a — количество минут, которое требуется для перехода от любой плиты к соседней, и b — количество минут, которое требуется граффити-художнику на разрисовывание одной плиты ($1 \le a, b \le 10^6$). В третьей строке заданы M чисел $p_1, p_2, \ldots, p_M$ — начальные положения граффити-художников ($1 \le p_i \le N$).

Формат выходных данных

В первую строку выходного файла выведите минимальное количество минут, требуемых художникам для выполнения работы.

В последующих M строках выведите описание действий художников. В i-й из этих строк должно содержаться описание действий i-го художника: количество плит, которые должен разрисовать этот художник, и номера этих плит в очередности их разрисовывания. Если оптимальных решений несколько, можно вывести любое из них.

Примечание

Решения, корректно работающие при $M \le 2$, будут оцениваться из 40 баллов.

Пример входных и выходных данных

fence.in	fence.out
3 4 2 3 3 1 3 3	5 1 2 1 1 1 3 0
2 1 1 1 1	3 2 1 2

C. Стековый калькулятор

Имя входного файла:	`stack.in`
Имя выходного файла:	`stack.out`
Ограничение по времени:	2 *секунды*
Ограничение по памяти:	256 *мегабайт*
Максимальная оценка:	100 *баллов*

В кабинете информатики есть старый *стековый калькулятор*. Он содержит K ячеек памяти, организованных в виде *стека*. Первая ячейка называется *вершиной* стека. На индикаторе калькулятора отображается содержимое вершины стека, если стек непуст.

Над стеком может выполняться операция *проталкивания*. Применение этой операции приводит к записи числа на вершину стека. Перед этим, если в стеке уже были числа, то каждое из них перемещается в ячейку с номером на единицу больше. Если в стеке уже находится K чисел, то выполнение операции проталкивания невозможно.

Калькулятор позволяет выполнять арифметические операции. Они применяются к числам, хранящимся во второй и первой ячейках стека. Результат операции записывается в первую ячейку стека, а число из второй ячейки удаляется. После этого, если третья ячейка непуста, то число из неё переписывается во вторую, если есть число в четвертой ячейке — оно переписывается в третью и так далее до последней занятой ячейки, которая становится пустой. Для выполнения арифметической операции в стеке должно быть хотя бы два числа. Например, при выполнении операций сложения или умножения, значения соответственно суммы или произведения чисел из первой и второй ячеек помещаются на вершину стека. Операция вычитания выполняется так: из содержимого второй ячейки стека вычитается содержимое первой ячейки.

Известно, что калькулятор неисправен. Из цифровых клавиш работает только клавиша «`1`». Нажатие этой клавиши приводит к проталкиванию в стек числа 1. Например, попытка набрать число 11, два раза нажав клавишу «`1`», приводит к тому, что в стек два раза проталкивается число 1. Из операций работают только сложение (клавиша «+»), умножение (клавиша «`*`») и вычитание (клавиша «−»). Если в результате вычитания получено отрицательное число, то калькулятор зависает.

Легко заметить, что на индикаторе возможно получить произвольное натуральное число. Например, чтобы получить число 3, необходимо дважды нажать клавишу «`1`», затем клавишу «+» (на индикаторе после этого появится число 2), затем один раз нажать клавишу «`1`» и один раз — клавишу «+». При $K > 2$ того же результата можно достичь иначе, трижды нажав клавишу «`1`», а затем два раза клавишу «+». Дополнительно используя операции умножения и вычитания, в некоторых случаях число N можно получить быстрее, чем сложив N единиц.

Требуется написать программу, которая определяет, каким образом можно получить на индикаторе калькулятора заданное число N, выполнив минимальное количество нажатий клавиш. Стек изначально пуст.

Формат входных данных

В единственной строке входного файла записаны два натуральных числа — N и K ($1 \le N \le 10^9$, $2 \le K \le 100$).

Формат выходных данных

В первой строке выходного файла выведите минимальное количество нажатий клавиш, необходимых для получения числа N. Если число нажатий не превосходит 200, то дополнительно выведите во второй строке оптимальную последовательность нажатий, приводящих к появлению данного числа на индикаторе.

Последовательность необходимо выводить без пробелов. Клавиши обозначаются символами «`1`» — протолкнуть число 1 в стек, «+» — выполнить операцию сложения, «`*`» —

выполнить операцию умножения, «−» — вычесть из значения второй ячейки стека значение первой ячейки.

В результате выполнения выведенной последовательности нажатий на индикаторе должно отображаться число N. Если оптимальных последовательностей нажатий несколько, разрешается выводить любую из них.

Примечания

Решения, корректно работающие при $N \le 100$ и $K \le 10$, будут оцениваться из 40 баллов.

Решения, корректно работающие при $N \le 10^6$ и $K \le 100$, будут оцениваться из 80 баллов.

Примеры входных и выходных данных

stack.in	**stack.out**
6 3	9 111++11+*
11 4	15 11+111++*11+*1-