Задача A. Битломания

Имя входного файла:	`beatlemania.in`
Имя выходного файла:	`beatlemania.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

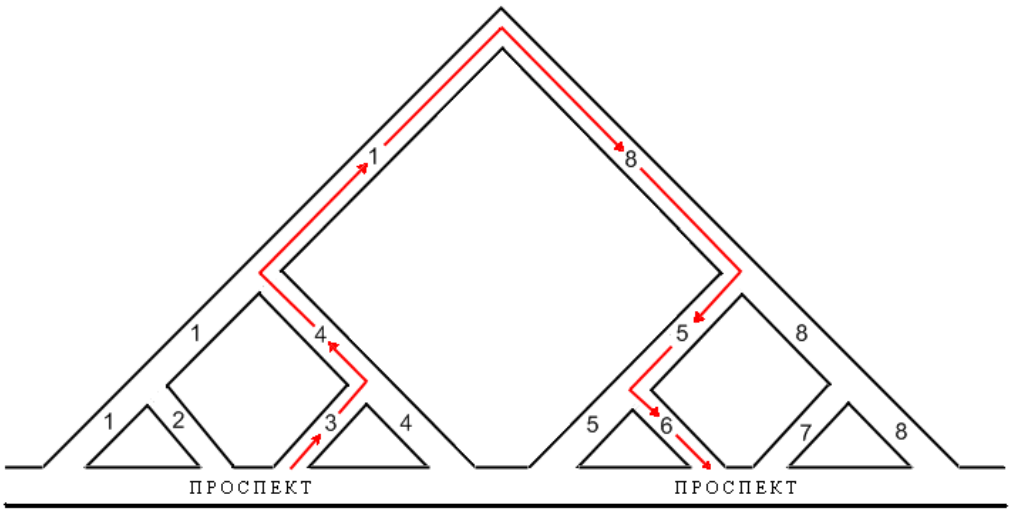
Популярность The Beatles в 60-е годы назвать иначе как манией нельзя: месяцами песни группы находились на вершинах хит-парадов, концерты собирали немыслимые по тем временам аудитории, а альбомы раз за разом побивали рекорды продаж. Но была и обратная сторона медали — музыканты не могли спокойно передвигаться по городу...

На улицах, в отелях, в аэропортах — "ливерпульскую четвёрку" всюду преследовали толпы поклонников и особенно поклонниц. Чтобы выбраться из здания, окруженного фанатами, музыкантам нередко приходилось прибегать к помощи полицейского кордона. Но в незнакомом городе, среди лабиринта улиц, где единственным, да и то временным, укрытием от фанатов служит телефонная будка, самый верный способ уйти от погони — позвонить менеджеру группы!

"Битлы" укрылись от погони в телефонной будке, расположенной на одном из оживлённых проспектов города у подножья холма. Продолжать бегство по проспекту нельзя — там слишком много людей. Скрыться от погони можно в переулках, берущих начало у проспекта и уходящих вверх по склону холма, разделённого на H ярусов. Всего переулков 2^H, все они пронумерованы по порядку, начиная с 1.

На каждом из ярусов одна половина переулков "врезается" в другую половину по следующим правилам. Рассмотрим список A_i, содержащий упорядоченные по возрастанию номера переулков, входящих в i-й ярус. Все соседние переулки объединим в пары, начиная с 1-го: $(A_{i,1}, A_{i,2})$, $(A_{i,3}, A_{i,4})$ и т. д. В парах под нечётными индексами второй переулок будет "врезаться" в первый, в парах под чётными индексами — наоборот. На последнем ярусе два переулка пересекаются, не образуя продолжения.

Рисунок ниже демонстрирует схему расположения переулков для холма с тремя ярусами.



Телефонная будка расположена возле начала переулка с номером S. Чтобы запутать толпу и не дать музыкантам потеряться в сети переулков, менеджер группы предлагает следующий план:

1. Совершить подъём до одного из ярусов.
2. Спуститься к подножью холма по траектории, зеркальной к траектории подъёма.

Повторять такие подъёмы и спуски необходимо до тех пор, пока участники группы не окажутся у подножья холма в начале переулка T, где их будет ждать машина. Вам необходимо определить

минимальное число подъёмов по холму, которое должны сделать музыканты для спасения от фанатов.

Формат входных данных

В единственной строке через пробел задаются число ярусов H ($1 \leqslant H \leqslant 25$) и номера начального и конечного переулков S, T ($1 \leqslant S, T \leqslant 2^H$).

Формат выходных данных

Выведите искомое число подъёмов по холму.

Пример

beatlemania.in	beatlemania.out
3 3 6	1
2 1 3	2

Задача B. Вторая попытка

Имя входного файла:	`chance.in`
Имя выходного файла:	`chance.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 Мебибайта

В 60-е годы были разработаны первые многозадачные операционные системы, такие, как Multics, OS/360, UNIX и т. д. Наряду с многозадачностью появился новый механизм управления памятью — свопинг.

Если оперативная память заполнена, то при выделении в ней места для новой страницы необходимо предварительно удалить одну из находящихся там страниц. Правила замещения страниц позволяют принять решение о том, какую страницу следует удалить из памяти. Идеальным кандидатом является страница, обращение к которой произойдёт позже всех остальных. Однако операционная система не может в точности предсказать обращения к страницам в будущем, поэтому для выбора удаляемой страницы применяются эвристические алгоритмы, например, алгоритм «Вторая попытка». Буфер представляет собой связный список не более, чем из N элементов, содержащих страницы и различные флаги.

- Флаг обращения устанавливается всякий раз, когда происходит обращение к странице для чтения или записи.
- Флаг изменения устанавливается только при выполнении операции записи.

Замещение страниц выполняется по следующим правилам.

1. Если искомая страница уже находится в списке — структура списка остаётся неизменной.
2. Если число элементов в буфере меньше N и необходимой страницы в нём не содержится, то она загружается в конец списка.
3. Если число элементов в буфере равно N и необходимой страницы в нём не содержится, выполняется последовательный просмотр списка от начала до тех пор, пока не встретится страница со снятым флагом обращения. Эта страница либо выгружается на диск (если установлен флаг изменения), либо удаляется. Если во время просмотра встречается страница с установленным флагом обращения, то он снимается, а страница перемещается в конец списка. После успешной выгрузки или удаления страницы из списка, страница, к которой изначально производилось обращение, загружается в конец списка.

Вам необходимо промоделировать работу алгоритма “Вторая попытка” для заданной последовательности операций и вывести информацию о выгружаемых и загружаемых страницах.

Формат входных данных

В первой строке через пробел задаются числа N и M ($1 \leqslant N \leqslant 1000$, $1 \leqslant M \leqslant 10^4$) — максимальное число элементов в списке страниц и число операций соответственно.

В следующих M строках задаются операции. Каждая строка начинается либо с символа ‘`R`’ (если это операция чтения), либо с символа ‘`W`’ (если это операция записи). Далее через пробел указывается номер страницы A_i ($1 \leqslant A_i \leqslant 10^6$), к которой происходит обращение для чтения или записи.

Формат выходных данных

Выведите M блоков, разделённых пустой строкой — по одному на каждую операцию.

Если в ходе выполнения операции произошла выгрузка страницы на диск, выведите в первой строке блока слово “`UNLOAD`” и номер страницы.

Если в ходе выполнения операции произошло удаление страницы, выведите в первой строке блока слово “`DELETE`” и номер страницы. Если в ходе выполнения операции в список была загружена новая страница, выведите слово “`LOAD`” и номер страницы. Если при выполнении операции структура списка не изменилась, выведите “`NONE`”.

Пример

chance.in	chance.out
3 6 R 1 W 2 R 3 W 7 W 3 W 9	LOAD 1 LOAD 2 LOAD 3 DELETE 1 LOAD 7 NONE UNLOAD 2 LOAD 9

Задача C. Конвейер

Имя входного файла:	`conveyor.in`
Имя выходного файла:	`conveyor.out`
Ограничение по времени:	8 секунд
Ограничение по памяти:	64 Мебибайта

Феноменальный рост экономики Японии в 60-е годы называют "японским экономическим чудом". Одним из основных факторов, обеспечивших столь стремительные темпы роста, было смещение приоритетов экономики к наукоёмким отраслям и освоение новых технологий. Японские производители, ориентированные в первую очередь на экспорт, одними из первых стали автоматизировать процесс контроля качества продукции на всех этапах производства.

Конвейер состоит из N блоков по M ячеек в каждом. Ячейка может быть либо пустой, либо заполненной одной единицей продукции. Изначально все ячейки пусты и конвейер остановлен.

- **Операция загрузки** заключается в том, что X копий некоторой последовательности ячеек непрерывно загружаются в начало конвейера. При этом все содержимое конвейера сдвигается вперёд. Заполненные ячейки, проходящие крайнюю границу последнего блока, содержат готовую продукцию, которая отправляется на склад.
- **Операция отката конвейера** заключается в том, что содержимое конвейера сдвигается в обратную сторону на Y ячеек. Заполненные ячейки, проходящие крайнюю границу первого блока, при этом очищаются, а содержащаяся в них продукция считается бракованной.

После полного выполнения каждой операции загрузки или отката конвейер останавливается. Продукция в ячейках, не успевшая пройти всю цепочку блоков, остается неизменной вплоть до выполнения следующей операции. Важнейшим свойством конвейера является поэтапный контроль производства, осуществляемый с помощью специальных устройств, расположенных на стыке каждой пары соседних блоков. Когда конвейер начинает движение (независимо от направления), то заполненная ячейка, которая первой проходит через устройство, очищается, а содержащаяся в ней продукция отправляется на контроль. Все пустые ячейки, а также заполненные ячейки, проходящие через устройство после этого, остаются неизменными.

Ваша задача — посчитать количество готовой, бракованной, отправленной на контроль и оставшейся на конвейере продукции после выполнения всех операций.

Формат входных данных

В первой строке через пробел заданы числа N ($1 \leqslant N \leqslant 5000$), M ($1 \leqslant M \leqslant 60$) и K ($1 \leqslant K \leqslant 5000$) — число блоков, ячеек в одном блоке и операций, соответственно. В следующих K строках задаются операции. Если первым в строке записан символ '`>`', то далее через пробел задаются число X_i ($1 \leqslant X_i \leqslant 10^6$) и строка S_i длиной не более M, состоящая только из символов '`0`' и '`1`'. Таким образом задаётся операция загрузки X_i копий последовательности ячеек S_i ('0' означает пустую ячейку, '1' — заполненную).

Если первым в строке записан символ '`<`', то далее указывается число Y_i ($1 \leqslant Y_i \leqslant 10^6$). Так задаётся операция отката конвейера на Y_i ячеек.

Формат выходных данных

Выведите через пробел четыре числа: количество готовой, бракованной, отправленной на контроль и оставшейся на конвейере продукции.

Пример

conveyor.in	conveyor.out
4 5 5 > 2 111 > 5 001 < 3 > 4 00100 > 1 00	1 1 10 3

Замечание

11111 00000 00000 00000 — после 1-й операции (1 на контроль);
00100 10010 01001 11000 — после 2-й операции (3 на контроль);
00000 10000 01010 00000 — после 3-й операции (1 в брак, 3 на контроль);
00100 00100 00100 00000 — после 4-й операции (1 готовая, 3 на контроль);
00001 00001 00001 00000 — после 5-й операции (без изменений).

Задача D. Точки

Имя входного файла:	`dots.in`
Имя выходного файла:	`dots.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 Мебибайта

Никто не знает точно, когда игра “Точки” проникла в школы и высшие учебные заведения СССР, но вполне вероятно, что случилось это именно в шестидесятые. Эта настольная вариация японской игры “Го” быстро завоевала популярность у пионеров и комсомольцев, ведь для игры требовались только письменные принадлежности и бумага. Важным преимуществом перед другими играми было также то, что игроки не создавали никакого шума и выглядели со стороны, как прилежные ученики, которые старательно что-то обдумывают и делают пометки в тетради.

Игра “Точки” ведётся на прямоугольном поле размером $N \times M$ пунктов. Пунктом считается пересечение линий на поле.

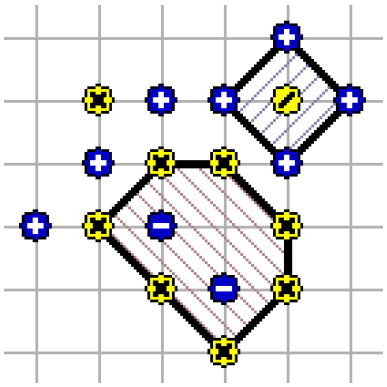
В игре принимают участие два игрока, которые ходят по очереди. За один ход игрок должен поставить одну точку в свободный пункт.

Свободным пунктом является пункт, не содержащий точки и не входящий ни в одну область окружения.

Область окружения — это область, окружённая непрерывной замкнутой ломаной, состоящей из активных точек одного игрока, содержащая строго внутри себя хотя бы одну точку соперника. Каждый отрезок ломаной проводится между двумя соседними по горизонтали, вертикали или диагонали точками игрока, которому принадлежит область. Область окружения всегда создаётся так, чтобы число активных точек игрока, которому она принадлежит, на её границе и внутри области было минимально возможным.

Активная точка — это точка игрока, не принадлежащая ни одной области окружения соперника.

Пассивная точка — это точка игрока, которая входит в область окружения соперника.



Необходимо разработать программу, которая по заданному списку ходов выведет иллюстрацию конечного состояния игры. Если ход является некорректным (пункт не является свободным), его следует игнорировать; при этом смены игрока не происходит. После совершения каждого хода программа должна создать все возможные области окружения, границы которых состоят из активных точек текущего игрока. При этом внутри области окружения все точки игрока соперника становятся пассивными, а точки текущего игрока — активными, вне зависимости от того, были ли они окружены соперником до этого хода.

Формат входных данных

В первой строке указываются размеры поля N и M ($1 \leqslant N, M \leqslant 50$) и число ходов K ($0 \leqslant K \leqslant 2500$).

Далее в K строках задаются числа R_i и C_i ($1 \leqslant R_i \leqslant N, 1 \leqslant C_i \leqslant M$) — номера строк и столбцов, в которые пытается походить игрок.

Формат выходных данных

Выведите иллюстрацию конечного состояния игры в виде поля $(N+2) \times (M+2)$, где каждому пункту соответствует один символ, согласно следующим правилам: символ '#' соответствует границам поля, символ '+' — активной точке первого игрока, символ '-' — пассивной точке первого игрока, символ '*' — активной точке второго игрока, символ '/' — пассивной точке второго игрока, пробел — пункту, не содержащему точки.

Примеры

dots.in	dots.out
6 6 20 2 3 2 2 2 4 3 3 3 5 2 5 2 6 3 4 1 5 4 2 4 1 4 5 3 2 6 4 5 4 5 5 4 3 5 3 4 4 1 5	######## # + # # *++/+# # +**+ # #+*- * # # *-* # # * # ########
4 5 11 3 3 4 5 2 2 1 2 1 3 3 2 2 4 2 1 3 1 2 3 4 2	####### # *+ # #*-*+ # #+*+ # # + *# #######
5 5 17 3 3 3 2 4 2 4 3 4 4 3 4 3 5 2 3 5 3 5 4 2 4 1 4 1 3 1 2 2 2 5 2 3 1	####### # *+* # # +/+ # #+/+/+# # +/+ # # *+* # #######

Задача E. HAL 9000

Имя входного файла:	`hal9000.in`
Имя выходного файла:	`hal9000.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 Мебибайта

Научно-фантастический фильм "Космическая Одиссея 2001 года" американского режиссера Стэнли Кубрика, вышедший в 1968-м году, по праву считается одним из величайших фильмов в истории кинематографа. Фильм знаменит не только революционными спецэффектами, с помощью которых его создателям впервые удалось достичь сверхреалистичного изображения космоса, но и своими прогнозами относительно ближайшего будущего человечества.

По сюжету фильма HAL 9000 — мыслящий суперкомпьютер, способный к самообучению. Он полностью контролирует ход полёта, общается с людьми на естественном языке, выражает эмоции и, безусловно, ему под силу обыграть любого человека в шахматы. Чтобы дать членам экипажа хоть какой-то шанс на победу, придётся существенно изменить правила игры.

Игра будет проводиться на доске $N \times M$ клеток. У каждого игрока есть всего одна фигура — ладья, координаты которой выбираются случайным образом перед началом игры. Игроки ходят по очереди, перемещая свою фигуру по горизонтали или вертикали, при этом не пересекая горизонтальную и вертикальную линии, на которых расположена фигура соперника, и не становясь на эти линии. Игрок, который не может сделать ход, терпит поражение. Вам необходимо определить исход игры при условии, что человек ходит первым и оба игрока действуют оптимально, то есть стараются выиграть за минимальное число ходов или проиграть за максимальное число ходов, если поражение неизбежно.

Формат входных данных

В единственной строке через пробел задаются размеры поля N, M ($1 \leqslant N, M \leqslant 100$) и координаты фигур человека X_1, Y_1 и компьютера X_2, Y_2 ($1 \leqslant X_1, X_2 \leqslant N$, $1 \leqslant Y_1, Y_2 \leqslant M$, $X_1 \neq X_2$, $Y_1 \neq Y_2$).

Формат выходных данных

В первой строке в случае победы человека выведите "`MAN`", в случае победы компьютера — "`HAL`". Во второй строке выведите число ходов, сделанных обоими игроками к моменту завершения игры.

Примеры

hal9000.in	hal9000.out
2 2 1 1 2 2	HAL 0
3 9 1 8 3 9	MAN 1

Задача F. Хиппимобиль

Имя входного файла:	`hippiemobile.in`
Имя выходного файла:	`hippiemobile.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 Мебибайта

Volkswagen T1 — именно так назывался один из первых гражданских микроавтобусов. Ему было суждено не только совершить переворот в автомобилестроении, но и стать легендарным "хиппимобилем" — символом эпохи конца шестидесятых... В среде хиппи были очень популярны путешествия, и такой микроавтобус как нельзя лучше подходил для этих целей. Он был вместительным, недорогим и вполне годился на роль временного жилища при длительных стоянках. Хиппи раскрашивали его яркими цветами, покрывали лозунгами и даже меняли оригинальный значок Volkswagen на символ мира — "пацифик". Одним словом, не заметить такое "произведение искусства" на дороге было просто невозможно!

В одной из стран, ставшей объектом паломничества для хиппи, есть N поселений. Между поселениями проложены M дорог, по которым можно ездить только в одном заданном направлении. Среди хиппи давно ходят легенды о таинственных дорогах. Если проехать по такой дороге один раз, то вернуться в поселение, в котором эта дорога начинается, уже не получится. Необходимо определить количество таких дорог в стране.

Формат входных данных

В первой строке через пробел заданы числа N ($1 \leqslant N \leqslant 10^5$) и M ($1 \leqslant M \leqslant 10^6$) — количество поселений и дорог, соответственно.

Далее следуют M строк, содержащих разделённые пробелом числа U_i и V_i ($1 \leqslant U_i \neq V_i \leqslant N$) — номера поселений, соединяемых i-й дорогой. Движение по дороге разрешено только в направлении из U_i в V_i. Каждую пару поселений может соединять несколько дорог.

Формат выходных данных

Выведите одно число — количество таинственных дорог в стране.

Пример

hippiemobile.in	hippiemobile.out
5 7 1 2 1 3 1 5 3 2 3 4 4 1 5 4	2

Задача G. Море дырок

Имя входного файла:	`holes.in`
Имя выходного файла:	`holes.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

“Давным давно однажды, а может быть дважды, существовал неземной рай и назывался он Пепперленд. И лежал он под водой, на глубине 80 000 лиг... Или лежит, — я в этом не уверен”, — так начинается мультипликационный фильм “Жёлтая подводная лодка” — один шедевров психоделической анимации, выпущенный в 1968-м году.

Жизнь текла своим прекрасным чередом, пока на Пепперленд не напали Синие Злюки... Они принялись замораживать всех направо и налево, а хуже всего было то, что они заморозили Ансамбль Клуба Одиноких Сердец Сержанта Пеппера — главную музыкальную силу Пепперленда. Юный Фред забрался в Жёлтую Подводную Лодку и уплыл на поиски Музыки, которая прогонит Синих Злюк. Он встретил группу The Beatles, и они согласились плыть с Юным Фредом в Пепперленд, чтобы помочь жителям вновь обрести волшебство Музыки и победить Синих Злюк. Теперь им предстоит полное загадок и приключений путешествие по различным морям — Морю Времени, Морю Науки, Морю Монстров и Морю Дырок.

Море Дырок на самом деле представляет собой прямоугольное поле размерами $N \times M$, в ячейках которого расположены дырки. Только K из них ведут в Пепперленд, а остальные дырки не ведут никуда, и, даже если на них наступить, ничего не происходит. Чтобы не заблудиться в Море Дырок, путнику нужно двигаться следующим образом:

1. Идти до упора в горизонтальном направлении (изменяется столбец).
2. Перейти на одну ячейку в вертикальном направлении (изменяется строка).
3. Сменить горизонтальное направление на противоположное.
4. Перейти к пункту 1.

Если при попытке выполнения пункта 2 оказывается, что ячейки в вертикальном направлении не существует, нужно, оставаясь на месте, сменить вертикальное и горизонтальное направления на противоположные и перейти к выполнению пункта 1.

Вам необходимо для каждой ячейки в Море Дырок посчитать, сколько минут потребуется путнику, чтобы попасть в чудесную страну Пепперленд, начав движение из этой ячейки, если на один переход между ячейками он тратит ровно одну минуту, а попав в ячейку, ведущую в Пепперленд, — сразу же перемещается туда. Известно также, что путник в начале пути в качестве горизонтального направления всегда выбирает движение вправо, а в качестве вертикального — движение вниз.

Формат входных данных

В первой строке через пробел задаются размеры поля N, M ($1 \leqslant N, M \leqslant 300$) и число K ($1 \leqslant K \leqslant N \times M$) — количество дырок, ведущих в Пепперленд. В следующих K строках через пробел задаются числа R_i и C_i ($1 \leqslant R_i \leqslant N$, $1 \leqslant C_i \leqslant M$) — номера строк и столбцов, в которых находятся ячейки с дырками, ведущими в Пепперленд. Все числа во входных данных целые. Строки занумерованы сверху вниз, а столбцы — слева направо.

Формат выходных данных

Выведите N строк по M чисел A_{ij} (соседние числа в списке разделены одним пробелом) — время в минутах, необходимое путнику, чтобы попасть в Пепперленд, начав движение из ячейки с координатами (i, j).

Пример

holes.in	holes.out
4 5 3 1 3 2 1 3 5	2 1 0 6 5 0 4 3 2 1 4 3 2 1 0 13 12 11 10 9
1 5 1 1 3	2 1 0 3 2

Задача N. Монополии

Имя входного файла:	`monopolies.in`
Имя выходного файла:	`monopolies.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мебибайта

В развитых капиталистических странах 60-е годы прошли под знаком борьбы с монополиями. Особенно остро проблема монополий проявлялась в гражданской авиации — в ряде регионов корпорации устанавливали полный контроль над воздушным пространством, необоснованно завышая цены и не обеспечивая при этом должного уровня обслуживания.

Комитет по антимонопольной политике выпустил постановление, согласно которому авиакомпания, сотрудником которой Вы являетесь, должна передать ряд прав на осуществление перелётов другим авиакомпаниям. Руководство компании заинтересовано в том, чтобы выполнение данного постановления не отразилось на позициях компании в регионе, и потребовало посчитать число пар городов, воздушное пространство между которыми является, по его мнению, стратегически незначимым.

Авиакомпания осуществляет перелеты между N городами, координаты которых известны. Воздушное пространство между парой городов представляет собой отрезок, соединяющий эти два города. Оно считается стратегически незначимым, если между любой парой городов, которые не входят в это воздушное пространство, существует хотя бы один маршрут (возможно, через другие города), не имеющий общих точек с этим воздушным пространством.

Формат входных данных

В первой строке указывается число N ($1 \leqslant N \leqslant 10^5$) — количество городов. В следующих N строках через пробел задаются координаты городов X_i, Y_i — целые числа, не превышающие по абсолютному значению 10^9 ($(X_i - X_j)^2 + (Y_i - Y_j)^2 > 0$ для любых $i \neq j$).

Формат выходных данных

Число пар городов, воздушное пространство между которыми является стратегически незначимым.

Пример

monopolies.in	monopolies.out
4 -1 0 0 0 0 1 1 2	5

Замечание

Стратегически незначимыми являются следующие пары городов: (1,2), (1,3), (1,4), (2,4), (3,4).

Задача I. В поле ягода навсегда

Имя входного файла:	`strawberry.in`
Имя выходного файла:	`strawberry.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Столь забавно в одном из советских журналов перевели название знаменитой песни The Beatles “Strawberry fields forever”. Эта фраза, ставшая нарицательной, прочно вошла в обиход русскоязычных поклонников группы, и даже вдохновила Бориса Гребенщикова на создание песни “В поле ягода навсегда”.

Никто не знает истинных причин появления такого перевода, и может быть, что на журналиста, писавшего статью, просто нахлынули детские воспоминания о том, как его заставляли собирать землянику с младшим братом. Поля казались бесконечными, сам процесс вечным, а младший брат и вовсе не помогал, просто съедая всю землянику, до которой мог добраться.

Земляничное поле состоит из бесконечного числа одинаковых прямоугольных полян размером $N \times M$ кустов, соприкасающихся друг с другом по краям и образующих прямоугольную сетку. Каждый куст содержит некоторое число ягод A_{ij}. На рисунке показана часть поля с полянами размером 2×2.

A_{11}	A_{12}	A_{11}	A_{12}	A_{11}	A_{12}
A_{21}	A_{22}	A_{21}	**A_{22}**	A_{21}	A_{22}
A_{11}	A_{12}	**A_{11}**	**A_{12}**	**A_{11}**	A_{12}
A_{21}	**A_{22}**	**A_{21}**	**A_{22}**	**A_{21}**	**A_{22}**
A_{11}	A_{12}	**A_{11}**	**A_{12}**	**A_{11}**	A_{12}
A_{21}	A_{22}	A_{21}	**A_{22}**	A_{21}	A_{22}

Старший брат выбирает один из кустов в качестве начального и собирает землянику с кустов, которые расположены на расстоянии не более K от начального. Расстояние между двумя кустами вычисляется как сумма модулей разности их координат. На рисунке жирным шрифтом выделена одна из возможных областей сбора ягод для старшего брата при $K = 2$.

Младший брат также выбирает один из кустов в качестве начального и ест землянику с кустов, которые расположены на расстоянии не более L от него. При этом начальный куст младшего брата должен быть максимально удалён от начального куста старшего, и область сбора ягод старшего брата должна полностью включать область младшего. На рисунке одна из возможных областей, доступных младшему брату при $L = 1$, выделена подчёркиваниями.

Ваша задача — посчитать максимальное число ягод, которые может собрать старший брат, при условии, что младший брат всегда успевает съесть все ягоды в своей области до того, как их начнёт собирать старший.

Формат входных данных

В первой строке через пробел задаются числа N, M ($1 \leqslant N, M \leqslant 100$) — размеры полян. Далее в N строках через пробелы задаются по M чисел A_{ij} ($0 \leqslant A_{ij} \leqslant 100$) — количество ягод на соответствующих земляничных кустах.

В последней строке через пробел задаются числа K и L ($1 \leqslant K \leqslant 10^6$, $1 \leqslant L < K$) — максимально допустимые расстояния от начальных кустов для старшего и младшего брата, соответственно. Все числа во входных данных целые.

Формат выходных данных

Искомое число ягод.

Пример

strawberry.in	strawberry.out
2 2 2 1 2 4 2 1	21

Задача J. Студия звукозаписи

Имя входного файла:	`studio.in`
Имя выходного файла:	`studio.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мебибайта

В наше время многие профессиональные музыканты предпочитают записывать музыку в домашних условиях, используя специальные программы, и прибегают к помощи студий звукозаписи лишь на этапе сведения. В 60-е годы подобное представить было трудно — оборудование было громоздким и дорогостоящим, и даже его аренда была многим не по карману. Музыканты должны были приходить в студию с полностью отрепетированным материалом и работать строго в отведённое им время. Может показаться невероятным, но на запись своего первого альбома, разошедшегося тиражом более 10 млн. экземпляров, легендарные Led Zeppelin потратили всего 36 часов студийной работы...

Для достижения такой эффективности одного профессионализма музыкантов недостаточно — весь персонал студии должен работать слаженно. Это касается даже тех, кто убирает помещения. В студии звукозаписи N помещений. Для помещений известно время T_i, которое необходимо затратить на их уборку в течение дня, а также график посещений. В штате один мастер чистоты, и Вы должны составить план его работы так, чтобы все помещения были полностью убраны, а суммарное время по всем посетителям, в течение которого они находились в помещениях при уборке, было минимально.

Для помещений задаётся также график ограничений, содержащий интервалы времени, в которые уборка либо запрещена полностью, либо должна длиться в сумме не более определённого числа минут. Мастер чистоты в течение рабочего дня может несколько раз убирать одно помещение, но время начала и окончания уборки всегда должно быть кратно минуте. В любой момент времени мастер чистоты может убирать не более одного помещения. Временем перехода между помещениями можно пренебречь.

Формат входных данных

В первой строке указывается число помещений N ($1 \leqslant N \leqslant 30$). Во второй строке через пробел задаются время прихода и ухода мастера чистоты в формате HH:MM. Продолжительность его рабочего дня находится в интервале от 1 до 500 минут.

Далее задаются N блоков описания помещений, перед каждым из которых следует пустая строка. В первой строке блока указывается число T_i ($0 \leqslant T_i \leqslant 500$) — время, необходимое для уборки i-го помещения в день.

Во второй строке блока задаётся число K_i ($0 \leqslant K_i \leqslant 100$) записей в графике посещений i-го помещения.

Далее следуют K_i строк, в которых через пробел указывается время прихода и ухода посетителей в формате HH:MM. Время ухода позже времени прихода как минимум на одну минуту.

В следующей строке задаётся число L_i ($0 \leqslant L_i \leqslant 10$) записей в графике ограничений.

Далее следуют L_i строк, в которых через пробел указывается время начала и конца интервала в формате HH:MM, а также число W_{ij} ($0 \leqslant W_{ij} \leqslant 500$) минут, разрешённых для уборки в i-м помещении в j-й интервал времени. Интервалы имеют длительность как минимум в одну минуту и не перекрываются внутри блока. Все числа во входных данных целые.

Начало суток в формате HH:MM задаётся значением 00:00, окончание суток — значением 24:00.

Формат выходных данных

Если убрать все помещения согласно описанным правилам нельзя, выведите "`NO`". Если решение существует, в первой строке выведите минимальное суммарное время по всем посетителям, в течение которого они будут находиться в помещениях при уборке. Далее выведите N блоков с описанием плана уборки каждого помещения в том же порядке, что и на вводе. Перед каждым блоком выведите пустую строку. В первой строке блока должно быть указано число P_i интервалов уборки i-го помещения. В следующих P_i строках через пробел выведите время начала и конца каждого интервала уборки в формате HH:MM, отсортировав их по времени начала. Если время начала одного

интервала совпадает со временем окончания другого интервала, то они должны быть объединены в один непрерывный интервал. Если существует несколько оптимальных решений — выведите любое из них.

Примеры

studio.in	studio.out
2 16:50 19:00 80 3 08:00 18:50 17:00 17:30 17:50 18:09 1 17:00 18:00 10 7 2 12:00 17:35 17:45 24:00 1 17:45 24:00 0	79 4 16:50 17:00 17:30 17:35 17:45 17:50 18:00 19:00 1 17:35 17:42
1 00:00 02:01 120 2 00:00 24:00 01:00 02:00 0	179 2 00:00 01:59 02:00 02:01
1 08:00 12:00 300 0 0	NO

Задача K. Турнир

Имя входного файла:	`tournament.in`
Имя выходного файла:	`tournament.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

В шестидесятые годы футбольные трансляции только начинали завоёвывать свою аудиторию, но телекомпании уже диктовали футбольным лигам свои условия...

В большинстве турниров при составлении календаря не учитывается географическое положение команд. Это создаёт ряд проблем в крупных странах, где перемещение между городами может занимать длительное время. Игроки после продолжительных перелётов находятся не в оптимальных кондициях, болельщикам приходится тратить немало времени и средств, чтобы посмотреть выездные матчи вживую, а телекомпании отказываются транслировать игры в прямом эфире из-за разности в часовых поясах.

В качестве эксперимента было решено опробовать схему с учётом географического фактора на небольшом кубковом турнире. В соревновании принимают участие N команд. Каждой команде присвоено уникальное значение L_i — показатель уровня, определяемый в соответствии с предыдущими результатами. На первой стадии турнира все команды разбиваются на пары. Для любого такого разбиения можно посчитать коэффициент равномерности W, равный сумме модулей разности показателей уровня команд во всех парах. Оптимальным считается такое разбиение команд, коэффициент равномерности которого является максимально возможным.

Для учета географического фактора в правила предлагается добавить пункт, согласно которому каждая пара разбиения должна состоять из команд одного региона. При этом в исходном списке команд чётные номера принадлежат командам первого региона, а нечётные — второго. Будем называть оптимальные разбиения, которые удовлетворяют этому пункту, оптимальными разбиениями с учётом географического фактора.

В силу того, что оптимальных разбиений с учётом географического фактора может быть много, было введено понятие "номер разбиения". Если записать разбиение, как массив из N чисел, в котором в каждой позиции находится индекс команды-соперника в паре, а затем отсортировать все разбиения лексикографически (по возрастанию числа в позиции 1; если эти числа равны, по возрастанию числа в позиции 2 и т. д.), то номером разбиения будет его индекс в отсортированном списке оптимальных разбиений с учётом географического фактора.

В задаче требуется по заданному номеру K вывести K-е оптимальное разбиение с учётом географического фактора, если оно существует.

Формат входных данных

В первой строке задано целое число команд N ($4 \leqslant N \leqslant 48$), всегда кратное четырём. Во второй строке через пробел задаются N положительных чисел L_i, не превышающих 10^9 — показатели уровня команд. В третьей строке записано число K ($1 \leqslant K \leqslant 10^{18}$) — номер искомого разбиения.

Формат выходных данных

Если не существует оптимального разбиения с учетом географического фактора, то есть во всех оптимальных разбиениях можно найти команду с чётным номером, соответствующую команде с нечётным, выведите "`NO`". Если номер искомого разбиения превышает общее число оптимальных разбиений с учётом географического фактора, выведите "`OVER`", иначе в первой строке выведите "`YES`", а во второй строке само разбиение, разделяя числа пробелами.

Примеры

tournament.in	tournament.out
8 21 72 16 83 51 32 64 45 2	YES 5 8 7 6 1 4 3 2
8 22 71 82 14 55 38 69 40 3	NO
4 11 22 33 44 2	OVER

Задача L. Вудсток

Имя входного файла:	`woodstock.in`
Имя выходного файла:	`woodstock.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Вудстокская ярмарка музыки и искусств — это один из самых знаменитых рок-фестивалей в истории, проходивший летом 1969-го года и собравший более полумиллиона зрителей. Считается, что на тот момент это была самая большая группа людей, когда-либо собиравшихся в одном месте, “чтобы иметь три дня забавы и музыки, и не иметь ничего, кроме забавы и музыки”, — так это описал владелец фермы, на территории которой проходил фестиваль.

Организаторам пришлось решать множество проблем при проведении фестиваля. Например, чтобы избежать давки и бесконтрольного перемещения людей, билеты на фестиваль продавались в так называемые периметры, которые формировались вокруг сцены. В течение дня разрешалось произвольно перемещаться только внутри своего периметра. Периметры состоят из секторов и нумеруются по порядку, в зависимости от расстояния до сцены, которая в свою очередь является нулевым периметром. Периметр с номером K состоит из $4K+1$ секторов, огибающих предыдущий периметр, как показано на рисунке:

3	2	1	0	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

Громкость и качество звука в каждом секторе могут существенно отличаться и выражаются целым числом A_{ij}. Вам необходимо для заданной конфигурации поля определить периметр, сумма значений A_{ij} в секторах которого максимальна.

Формат входных данных

В первой строке указывается максимальный номер периметра N ($1 \leqslant N \leqslant 100$).

В следующих $N+1$ строках, каждая из которых состоит из $2N+1$ чисел, задаются значения A_{ij} ($|A_{ij}| \leqslant 10^5$) для соответствующих секторов. Сцена находится в первой строке в позиции $N+1$.

Формат выходных данных

Выведите два числа — номер искомого периметра и сумму значений A_{ij} в его секторах. В случае равенства сумм выбирается периметр с меньшим номером.

Примеры

woodstock.in	woodstock.out
2 1 2 3 2 -2 2 3 5 4 -1 3 5 5 3 0	1 16
3 -1 1 1 10 1 1 -1 -2 1 1 12 1 1 -2 -3 1 1 15 1 1 -3 -7 10 20 30 20 10 -6	3 65