Задача A. Assembler

Имя входного файла:	`asm.in`
Имя выходного файла:	`asm.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Если Вы спросите, кто является главной достопримечательностью Удмуртии, то в первую очередь вам назовут М. Т. Калашникова, изобретателя знаменитого автомата. Вторым же с большой вероятностью окажется профессор логики Н. Н. Налейпиво, знаменитый не только своими научными работами, но и нестандартными взглядами на образование.

Одной из последних разработок профессора Налейпиво является система тестирования навыков выбора средства программирования. Проверяемый может решить задачу на одном из упрощённых языков программирования (на данный момент — PL-1, Рефал, Prolog, Brainf**k и ассемблер), после чего система определяет адекватность сделанного выбора. Вам поступил заказ на разработку модуля этой системы, отвечающей за работу с ассемблером.

Более точно, Ваша задача — вычислить результат выполнения программы, написанной на простом языке, похожем на ассемблер. Программа на этом языке оперирует с четырьмя 32-битными регистрами `AX`, `BX`, `CX` и `DX`. Каждая строка программы содержит оператор и, опционально, несколько операндов. Все возможные операторы и соответствующие им операнды перечислены в таблице:

Оператор	**Ограничения**	**Описание**
`MOV` a b	a — регистр b — регистр или константа	Помещает значение b в регистр a
`ADD` a b	a — регистр b — регистр или константа	Прибавляет значение b к значению a и помещает результат в a
`SUB` a b	a — регистр b — регистр или константа	Вычитает значение b из значения a и помещает результат в a
`MUL` a b	a — регистр b — константа	Умножает значение a на значение b и помещает результат в a
`LOOP` a	a — положительная константа	Повторяет все строки кода начиная со следующей и заканчивая соответствующим оператором ENDL a раз
`ENDL`	*нет*	Используется для обозначения конца тела цикла для соответствующего оператора LOOP.

Все величины в программе являются 32-битными целыми со знаком. Все операции выполняются также над 32-битными целыми числами со знаком.

Соответственно, после выполнения следующей программы:

```
MOV AX 2147483647
ADD AX AX
```

`AX` примет значение -2.

Количество вложенных циклов неограничено.

Ваша задача — по данной программе и начальным значениям регистров вывести значения регистров после выполнения программы.

Формат входного файла

Входной файл содержит программу, которую необходимо выполнить. Каждая строка входного файла содержит оператор, записанный заглавными латинскими буквами, и соответствующие

операнды, разделённые одним пробелом. Если операнд — регистр, он будет задан ровно двумя заглавными буквами. В программе нет пустых строк, равно как пробелов в начале или конце строк. Первые четыре строки программы всегда являются операторами `MOV`, которые задают некоторые начальные значения всех четырёх регистров.

Программа содержит не более 10000 строк кода.

Формат выходного файла

Выведите четыре целых числа: значения регистров `AX`, `BX`, `CX` и `DX` соответственно после выполнения программы.

Пример

asm.in	asm.out
MOV AX 15 MOV BX 20 MOV CX 25 MOV DX 30 LOOP 3 ADD AX BX ENDL MUL DX 3	75 20 25 90

Задача B. Blue-White Tree

Имя входного файла:	`bluewhite.in`
Имя выходного файла:	`bluewhite.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

В методике профессора Налейпиво значительное место занимают разнообразные игры. Профессор считает, что это удобный способ формулировки задачи, к тому же позволяющий создать новые формы взаимодействия обучаемых (отличные от банального списывания). Так что на его лекциях вполне возможно появление, например, подобных задач.

Вы с другом играете в следующую игру: перед вами подвешенное дерево с некоторым неотрицательным количеством черники в каждой вершине. Также каждая вершина в каждый момент времени является либо синей, либо белой; при этом изначально все вершины белые. Поочерёдно каждый из игроков выбирает некоторую вершину и выполняет с ней одну из следующих операций:

- *Операция поедания*. Можно применять только к вершинам с положительным количеством черники. Игрок съедает любое положительное количество черники из вершины и красит вершину в синий цвет.
- *Операция сбора*. Можно применять только к белым вершинам, у которых есть хотя бы один ребенок, и все потомки (не только непосредственные дети) белые. Игрок собирает всю чернику со всех потомков текущей вершины и кладет их в текущую вершину. Текущая вершина и все потомки красятся в синий цвет.
- *Операция разделения*. Если в текущей вершине M ягод и $M > 1$, игрок выбирает число K ($K < M$, $M \bmod K > 0$), добавляет K синих детей к текущей вершине и кладет в каждую $\text{floor}(M/K)$ ягод. Затем игрок съедает оставшиеся $M - \text{floor}(M/K) \times K$ ягод. Если выбрать число K, удовлетворяющее указанным условиям, нельзя, операцию нельзя применять к вершине.

Игрок, который не может совершить ход, проигрывает. Ваша задача определить, кто выиграет, если известно, что и вы и ваш друг играете оптимально, и вы ходите первым.

Формат входного файла

Первая строка входного файла содержит число N, затем $N - 1$ целых чисел P_i, где P_i номер родителя i-ой вершины ($0 \le i \le N$).

Вторая строка содержит N целых чисел B_i, где B_i — количество черники в i-ой вершине.

$1 \le N \le 10^4$

$0 \le P_i < i$

$0 \le B_i \le 1024$

Формат выходного файла

Выведите в выходной файл одну букву `W`, если при оптимальной игре выиграете вы, и букву `L` в противном случае.

Пример

bluewhite.in	bluewhite.out
5 0 1 1 1 8 0 1 2 4	L
5 0 1 1 1 7 0 1 2 4	W
3 0 0 0 0 0	W

Примечание: в третьем примере первый игрок может применить операцию сбора для 0-ой вершины, несмотря на то, что на вершинах дерева нет ни одной ягоды.

Задача C. Cruisers

Имя входного файла:	`cruisers.in`
Имя выходного файла:	`cruisers.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Среди всех игр профессор Налейпиво особенно выделяет стратегические игры. Построения профессора, подкреплённые многолетним опытом, позволили ему выдвинуть гипотезу, согласно которой наилучшей проверкой способностей абитуриента при поступлении в вуз является не экзамен, и уж тем более не тест ЕГЭ — а запись сыгранной участником партии в одну из стратегических игр. Для проверки этой гипотезы компания «New Paradigms Of Computer» по заказу ряда крупнейших российских вузов разрабатывает специальную стратегическую игру космической тематики.

Ваша задача — реализовать модуль, отвечающий за передвижение космических кораблей. Каждый корабль имеет следующие характеристики: *максимальную скорость*, *ускоряемость* и *поворотливость*. Изначально корабль расположен в координатах sx, sy, с направляющим вектором dx, dy и имеет скорость, равную *начальной скорости*. Корабль должен прилететь в координаты ex, ey. Он выполняет следующие действия для достижения цели:

- Если точка назначения лежит на пути движения корабля, он летит прямо.
- Иначе строятся две окружности с радиусом, равным *поворотливости* корабля, где точка (sx, sy) является точкой касания, а прямая, проходящая через точки (sx, sy) и (sx+dx, sy+dy) является касательной к обеим окружностям. Гарантируется, что точка (ex, ey) не лежит ни на одной из этих окружностей.
- Корабль движется вдоль одной из окружностей, пока не окажется направленным строго на конечную точку, после чего двигается прямо. Если точка назначения не лежит внутри одной из окружностей, выбирается окружность, для которой расстояние, которое необходимо пролететь вдоль окружности, меньше (гарантируется, что конечная точка выбрана так, чтобы расстояние, которое необходимо пролететь вдоль одной из окружностей, было строго меньше, чем вдоль другой). Иначе он движется вдоль окружности, внутри которой точка назначения не находится.

Если корабль имеет скорость 0, он потратит время, равное его *ускоряемости*, чтобы набрать *максимальную скорость*, равно как если сейчас его скорость равна *максимальной скорости*, он затратит время, равное его *ускоряемости*, чтобы полностью остановиться. Корабль разгоняется и тормозит с постоянным ускорением. Он не может разгоняться или тормозить медленнее. Во время полета корабль сначала разгоняется до *максимальной скорости*, затем некоторое количество единиц времени (возможно нулевое, не обязательно целое) летит с *максимальной скоростью*, и, наконец, тормозит, пока не остановится в точке назначения.

Есть два исключения:

- Если после достижения *максимальной скорости* невозможно остановиться в точке назначения (то есть даже начав тормозить сразу после достижения *максимальной скорости* корабль будет иметь положительную скорость в точке назначения), корабль сначала разгоняется до некоторой *особой скорости*, после чего сразу начинает тормозить. *Особая скорость* выбирается таким образом, чтобы корабль, начав тормозить в момент ее достижения, остановился ровно в точке назначения.
- Если даже, начав в первый момент времени тормозить, корабль не может остановиться в точке назначения, то по достижении этой точки он продолжает движение по прямой до полной остановки.

Ваша задача по заданным характеристикам корабля определить, где он будет в момент времени t.

Формат входного файла

Первая строка содержит четыре вещественных числа: ms, s, a и r — *максимальную скорость*, *начальную скорость*, *ускоряемость* и *поворотливость* соответственно.

Вторая строка содержит шесть чисел: sx, sy, ex, ey, dx, dy.

Третья строка содержит единственное вещественное число t.

$1 \le ms \le 1000.0$

$0 \le s \le ms$

$1 \le a \le 1000.0$

$-1000.0 \le sx, sy, ex, ey \le 1000.0$

$-100.0 \le dx, dy \le 100.0$

$0 \le t \le 10^{12}$

Формат выходного файла

Единственная строка выходного файла должна содержать два вещественных числа x и y — координаты корабля в момент времени t с точностью до четырех знаков после запятой.

Пример

cruisers.in	cruisers.out
10.0 0.0 10.0 5.0 0.0 0.0 100.0 0.0 1.0 0.0 10.0	50.0 0.0
1.0 1.0 1.0 2.0 0.0 0.0 2.0 4.0 1.0 0.0 3.14159265358979323846264338327 95	2.0 2.0
1.0 1.0 1.0 2.0 0.0 0.0 1.0 2.0 1.0 0.0 3.14159265358979323846264338327 95	2.0 -2.0

Задача D. Develop a Water-Pipe

Имя входного файла:	`develop.in`
Имя выходного файла:	`develop.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Профессор Налейпиво на одной из лекций рассказал, что в своё время он много играл в следующую игру: задана схема водопровода, расположенная на двумерной плоскости и разбитая на квадратные ячейки. Каждая ячейка является либо `I`-ячейкой, которая соединяет трубой свою левую границу с правой или верхнюю с нижней, либо `L`-ячейкой, которая соединяет правую границу с нижней, левую с нижней, левую с верхней или правую с верхней, либо `T`-ячейкой, которая соединяет любые три границы, или, наконец, `A`-ячейкой, которая соединяет все четыре границы.

Также есть ячейки *исток* и *сток*. *Исток* соединяет свою правую границу с некоторым внешним источником воды, в то время как *сток* соединяет свою левую границу с ванной. Может быть больше чем одна ячейка с *истоком* или *стоком*.

Водопровод называется *правильным*, если все его ячейки расположены таким образом, что если одна из них соединяет, например, свою левую границу с чем-либо, то ячейка, расположенная слева от данной, обязательно соединяет свою правую границу с чем-либо.

В *правильном* водопроводе исток не обязательно должен быть связан со стоком.

Перед началом игры все ячейки были повернуты несколько раз на 90 градусов. Задача игрока: поворачивая квадраты, получить *правильный* водопровод.

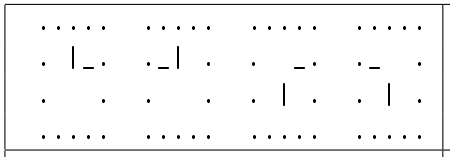
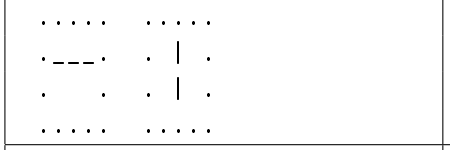
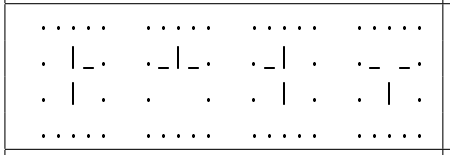
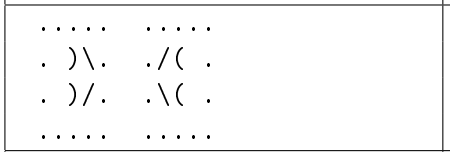
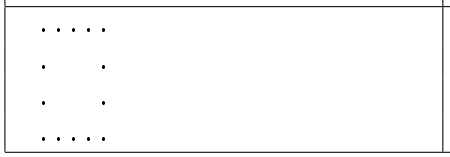
После нескольких уровней правила игры усложнились: водопровод нельзя сделать *правильным*, не заменив часть `T`-ячеек на `A`-ячейки.

Профессор заявил, что тот, кто обыграет его в эту игру, получит экзамен ижевским автоматом.

Учитывая, что после зависания сервера на первом этапе Кубка Вам захотелось достать автомат (и пару рожков к нему), Вам срочно нужна программа, которая решит все оставшиеся уровни раньше, чем профессор решит их сам.

Формат входного файла

Первая строка входного файла содержит два числа N и M — размеры схемы водопровода. Затем идет $3N + 1$ строка, каждая содержащая $4M + 1$ символов. Каждая ячейка задается 4 строками по 5 символов каждая согласно следующей таблицы (в описании ячеек используются только символы точки, подчеркивания, пробела и вертикального слэша):

	L-ячейки
	I-ячейки
	T-ячейки
	A-ячейки
	*Исток* и *Сток* соответственно *Примечание: используются символы точки, слэша, обратного слэша, пробела и скобок.*
	Пустая ячейка

$1 \le N \le 25$
$1 \le M \le 25$
Для лучшего понимания смотрите пример входных и выходных данных.

Формат выходного файла

Выведите схему правильного водопровода, полученную из входной вращением некоторых ячеек и заменой нескольких (возможно нуля) `T`-ячеек на `A`-ячейки в том же формате, как во входном файле. Если правильный водопровод получить нельзя, выведите единственное слово `IMPOSSIBLE`.

Пример

develop.in	develop.out
3 6 ..... .)\.___. | ._| . . . .)/. . | . . . . .___. . . . . ._| .___./(. . . . . . .\(.	..... .)\.___.___._ . . . .)/. . . | . . . . | . . . . | . . . . |_.___./(. . . . . . .\(. .....
3 6 ..... .)\._ _. | ._| . |_./(. .)/. | . | . | . | .\(. ..... . |_._|_._| .___. | . . . . | . . .. |_._|_._|_._|_._ _./(. . . . . . | .\(.	..... .)\._ _.____._ _._ _./(. .)/. | . . | . | .\(. ..... . _._|_._ . | . | . . . | . | . | . | . | . . .. |_._|_._|_._|_._|_._|_./(. . . . . . .\(. .....
3 6 ..... .)\.___. | ._| . |_./(. .)/. . | . | . | .\(. ..... .)\._|_. .___. | . . .)/. . . . | . . .. ._| ._ _./(. . . . . . | .\(.	IMPOSSIBLE

Задача E. Expected Number of Crystals

Имя входного файла:	`expected.in`
Имя выходного файла:	`expected.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Развивая свою идею относительно взаимодействия обучаемых, а также по поводу игры как одного из основных способов формулировки задачи, профессор Налейпиво пришёл к выводу о том, что студенты часть заданий должны выполнять внутри какой-нибудь MMORPG.

В одной очень популярной онлайн-игре кристаллы являются очень важным ресурсом. Один из способов получить кристаллы — посетить волшебную поляну. Процесс сбора кристаллов на поляне заключается в следующем: есть поле, состоящее из $N \times N$ ячеек, M из которых содержат кристаллы, в то время как все остальные не содержат. Вы можете открыть K любых ячеек и забрать все кристаллы, лежащие в открытых ячейках.

Есть два режима игры - *слепой* и *базовый*. В *слепом* вы сначала выбираете K ячеек, и только после выбора всех они открываются и вы узнаете сколько кристаллов вы собрали. В *базовом* режиме после открытия каждой ячейки вы сразу узнаете, содержала она кристалл или нет.

Ваша задача написать программу, определяющую ожидаемое количество кристаллов, которое вы найдете, если известны N, M, K и режим игры.

Формат входного файла

Первая и единственная строка содержит четыре числа: N, M, K и V, где V равно единице, если игра идет в *слепом* режиме, и двум, если в *базовом*.

$1 \le N \le 10^{30}$
$1 \le M \le 10^{30}$
$1 \le K \le 10^{30}$
$M \le N^2$
$K \le N^2$

Формат выходного файла

Если ожидаемое количество кристаллов целое, выведите в выходной файл одно число, равное ожидаемому количеству кристаллов. Иначе выведите ожидаемое количество в виде несократимой дроби.

Примеры

expected.in	expected.out
3 3 3 1	1
6 10 10 1	25/9

Задача F. Forum Search

Имя входного файла:	`forum.in`
Имя выходного файла:	`forum.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

По мнению профессора Налейпиво, умение решать задачи на поиск в Интернете является в наше время необходимым навыком. Так что многим своим студентам профессор даёт нестандартные задачи на поиск. Недавно Налейпиво был приглашён прочитать курс лекций студентам одного столичного спецвуза, и после первой лекции выяснил, что искать никто из студентов не умеет.

Было решено начать с более простой модели — англоязычных форумов, посвящённых MMORPG. Одна проблема — форумы делались наспех в те моменты, когда игровой сервер висел, так что поиск по форумам реализован не был...

Ваша задача — реализовать поиск для такого форума. Форум состоит из топиков, топики из постов, каждый пост представляет собой разделенную пробелами последовательность английских слов, написанных в нижнем регистре.

Каждый поисковый запрос состоит из нескольких ключевых слов. Есть три режима поиска:

1. Найти все топики, содержащие все ключевые слова внутри одного поста, независимо от порядка. Например, если запрос «`A B C`», и текст поста «`D B C A`», топик, содержащий этот пост, должен быть возвращен поисковым алгоритмом.

2. Найти все топики, содержащие все ключевые слова внутри одного поста как подпоследовательность. Например, если запрос «`A B C`», и текст поста «`D B C A`», топик, содержащий пост, не должен быть возвращен, в то время как если пост «`A D B C`», топик, его содержащий, должен быть возвращен.

3. Найти все топики, содержащие все ключевые слова внутри одного поста как подстроку. Например, если запрос «`A B C`», и текст поста «`D B C A`» или «`A D B C`», топик, содержащий пост, не должен быть возвращен, в то время как если пост «`D A B C`», топик, его содержащий, должен быть возвращен.

Формат входного файла

Первая строка содержит одно число N — количество топиков.

Каждый топик задан следующим образом: первая строка содержит одно целое число M — количество постов в топике, затем M строк, каждая состоит из разделенных одним пробелом английских слов в нижнем регистре — содержимое соответствующего поста.

После описания содержимого форума идет строка, содержащая число K — количество запросов. Каждый запрос задан на отдельной строке, содержащей тип запроса (1, 2 или 3) и произвольное количество разделенных одним пробелом ключевых слов.

Размер входного файла не превышает 400 KB. Общее количество ключевых слов по всем запросам не превышает 12000. Все ключевые слова не короче четырех символов. Все ключевые слова внутри одного запроса различны.

Вы можете рассчитывать на то, что все топики во входном файле (кроме примера) взяты с некоторого реального форума, посвященного фэнтэзи, без постов, содержащих флуд.

Несмотря на то, что запросы содержат популярные слова, такие как «that» и «have», их общее количество по всем запросам так мало, что вы можете исходить из предположения, что запросы почти не содержат таких слов.

Формат выходного файла

Для каждого запроса выведите одну строку. Строка должна начинаться с количества топиков, найденных поисковым алгоритмом. Для каждого из найденных топиков выведите его номер во входном файле начиная с 1, количество постов в топике, удовлетворяющих запросу, и номера постов

внутри топика (начиная с 1). Номера найденных топиков и постов внутри топика должны идти в порядке возрастания.

Пример

forum.in	forum.out
2 2 d b c a a d b c 1 d a b c 3 1 a b c 2 a b c 3 a b c	2 1 2 1 2 2 1 1 2 1 1 2 2 1 1 1 2 1 1

Задача G. 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 `+`.

$1 \le N \le 1000$

$1 \le M \le 1000$

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 H. How Many Chipmunks are There?

Имя входного файла:	howmany.in
Имя выходного файла:	howmany.out
Ограничение по времени:	3 seconds
Ограничение по памяти:	256 Mebibytes

Профессор Налейпиво часто поясняет вводимые им теоретические понятия различного рода занимательными историями. А чтобы студенты учились мыслить критически, он часто намеренно делает ошибочные выводы — и наблюдает за реакцией зала. Вот история, которую профессор стал рассказывать после того, как пообщался с участниками летних сборов по программированию.

В секретной лаборатории КоДаПет (название является аббревеатурой, составленной из фамилий сотрудников), был выведен новый вид боевых бурундучков. Они намного умнее любых других бурундучков, даже Чипа и Дэйла. Но работа еще не завершена, и ученые в лаборатории активно трудятся над выведением новых типов бурундучков. Каждый бурундучок характеризуется силой и умом. В то же время, каждый бурундучок в КоДаПет уникален, и даже если два бурундучка имеют равные силу и ум, они все равно остаются абсолютно разными.

Вы наверное хотите спросить, а как КоДаПет получает новые виды бурундучков? Очень просто. Всё, что вам надо знать, это что происходит, когда два бурундучка встречаются.

Если разница в силе у двух бурундучков при встрече 7 или более, то более сильный съедает более слабого. Иначе бурундучки играют в разные весёлые игры и новый бурундучок появляется на свет. Этот новый бурундучок имеет силу и ум, равные сумме силы и сумме ума его родителей соответственно. Но если те же самые бурундучки встретятся снова, и снова сыграют в свои весёлые игры, новый бурундучок, появившийся на свет, будет точно таким же как предыдущий, созданный теми же родителями, и, так как он нарушает правило «каждый бурундучок индивидуален», он будет немедленно продан в Индию.

На самом деле, учёные не хотят, чтобы бурундучки захватили мир, поэтому если ум бурундучка выше 4000 или сила выше 20, он будет немедленно продан инопланетянам.

Изначально в распоряжении учёных есть несколько бурундучков с умом от 1 до 4000 включительно и силой 1. Они хотят получить бурундучков с умом k и силой s.

В конце этой истории профессор утверждает, что максимальное количество бурундучков с умом k и силой s, которых в конечном итоге никуда не продали, по модулю 40961 оказалось равно константе эффективности обучения m, вычисленной им совершенно из других соображений.

Проверьте утверждение профессора.

Формат входного файла

Первая строка содержит три числа m, k и s.

Вторая строка содержит от 1 до 4000 целых чисел a_i - количество бурундучков с умом i и силой 1, которые есть у ученых изначально. Если в строке меньше 4000 чисел, у ученых нет бурундучков с большими умом.

$1 \le k \le 4000$
$1 \le s \le 20$
$0 \le m < 40961$
$0 \le a_i < 40961$

Формат выходного файла

Если максимальное количество бурундучков, которое может быть получено, равно m по модулю 40961, выведите «YES». Иначе выведите «NO».

Пример

howmany.in	howmany.out
30 4 4 3	YES

Задача I. Inhabitants (or Yet Another Mincost Maxflow Problem)

Имя входного файла:	`inhabitants.in`
Имя выходного файла:	`inhabitants.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Очередной эксперимент, проведённый профессором Налейпиво, состоял в создании MMORPG с резко отличающейся от привычной системой построения реальности и нестандартными базовыми ресурсами. В результате получился довольно мрачный — и при этом необычный мир. Вот история одного из континентов этого мира.

Страна оказалась под контролем Темного Лорда. Каждому человеку в стране строжайше запрещено покидать город, в котором он находится, хотя в стране есть несколько односторонних дорог, соединяющих города.

На самом деле, Темный Лорд не контролирует людей в Стране непрерывно. Большую часть времени он плетёт с тринадцатью Избранными интриги против юнца по имени Ранд. Единственное, что Темный Лорд делает для контроля Страны — он проверяет, сохраняется ли количество людей в каждом из городов одинаковым между двумя проверками. Время между двумя последовательными проверками постоянное и равно одному месяцу. Если во время проверки Темный Лорд заметит, что в одном из городов количество жителей изменилось, он немедленно уничтожит Страну.

Невероятное совпадение, но дойти от одного города до другого, если есть соответствующая дорога, можно ровно за месяц. То есть если житель покидает город сразу после проверки, он придет в другой город непосредственно перед очередной проверкой. Единственная проблема — это действие не должно повлиять ни на количество жителей в городе, который он покинул, ни на количество жителей в городе, в который он пришел. То есть кто-то должен прийти в исходный город и кто-то уйти из конечного.

Будучи под властью Тёмного Лорда, люди становятся несчастными. И средний уровень счастья жителей страны становится все хуже и хуже. В то же время дороги имеют магический эффект: они изменяют уровень счастья человека, идущего по ним. Более конкретно, каждая дорога имеет некоторое *волшебное значение*, и когда кто-то идет по этой дороге, его счастье уменьшается на *волшебное значение* этой дороги.

Сторонники Тёмного Лорда за пределами страны хотят попасть в неё, в то время как многие внутри страны хотят ее покинуть.

В Стране N городов, пронумерованных от 1 до N. Есть только одна дорога, по которой люди могут попасть в Страну. Эта дорога ведёт из-за пределов страны в город 1. Также есть дорога, по которой люди могут покинуть страну, и она ведёт из города N за пределы страны.

Эти дороги очень широкие, соответственно любое количество людей может войти в страну или покинуть её в течение одного месяца, в то время как дороги внутри страны имеют *пропускную способность* — максимальное количество людей, которое может пройти по этой дороге в течение месяца.

Вы один из последователей Творца, и ваша миссия — удовлетворить максимальное количество запросов на вход или выход из страны, если известно, что общее количество этих запросов намного больше, чем Страна может выполнить без риска быть уничтоженной. Также вы хотите поддержать средний уровень счастья как можно выше. К сожалению, неизвестно, сколько человек проживает в каждом конкретном городе (хотя вы можете рассчитывать на то, что это количество строго больше, чем сумма всех *пропускных способностей* дорог в Стране), так что вы не можете посчитать средний уровень счастья людей ни до того, как они покинули города, ни после того, как они дойдут до своих пунктов назначения.

Но, разумеется, вы можете посчитать среднее изменение уровня счастья у людей, которые воспользовались возможностью перейти из города в город. Соответственно, ваша задача для одного конкретного месяца:

1. Определить максимальное количество людей, которые могут войти в страну, и покинуть ее (эти числа, очевидно, равны);

2. Для максимального количества людей, которые могут войти в страну, минимизировать среднюю величину, на которую уменьшится счастье людей, которые перейдут из одного города в другой (не включая людей, вошедших в страну или покинувших ее).

Формат входного файла

Первая строка входного файла содержит два числа: N и M — количество городов и дорог соответственно. Затем следует M строк, каждая содержит четыре числа: u_i, v_i, c_i, m_i, где i-ая дорога ведет из города u_i в город v_i, имеет пропускную способность c_i и волшебное значение m_i.

$2 \le N \le 20$
$1 \le M \le 20$
$1 \le u_i \le N$
$1 \le v_i \le N$
$1 \le c_i \le 6$
$-10 \le m_i \le 10$

Есть хотя бы один направленный путь от 1-ого города в N-ый.

Два города могут быть соединены более чем одной дорогой, равно как дорога может вести из города в самого себя.

Формат выходного файла

Единственная строка выходного файла должна содержать одно вещественное число — минимальное среднее число, на которое уменьшится уровень счастья людей в течение одного месяца. Выведите ответ с точностью не меньше чем до четвертого знака.

Примеры

inhabitants.in	inhabitants.out
2 1 1 2 1 4	4.0
3 4 1 2 2 1 2 3 1 2 2 3 1 3 2 3 1 4	1.75
7 6 1 2 2 5 2 3 2 5 3 7 2 5 4 5 2 3 5 6 2 3 6 4 2 3	4.0
7 6 1 2 2 5 2 3 2 5 3 7 2 5 4 5 2 6 5 6 2 6 6 4 2 6	5.0

Задача J. JavaScript Interpreter

Имя входного файла:	`js.in`
Имя выходного файла:	`js.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Особое внимание профессор Налейпиво уделяет выбору правильного инструмента для решения той или иной задачи. Поэтому упоминавшаяся в первой задаче система тестирования навыков выбора средства программирования за то время, пока Вы читали предыдущие задачи, успела получить своё развитие — было принято решение о добавлении языка JavaScript.

Так что сейчас вам надо реализовать интерпретатор упрощенной версии языка JavaScript, который оперирует с некоторой простой объектной моделью HTML страницы.

Страница будет отображена в текстовом браузере с разрешением 20 строк и 40 столбцов.

Изначально, до выполнения JavaScript кода, страница абсолютно пуста и объектная модель содержит единственный элемент - *document*.

Для создания нового элемента используется метод *createElement* элемента *document*. Он принимает единственный параметр, который для данной задачи всегда равен `"span"` в двойных кавычках, и возвращает указатель на созданный элемент.

Каждый элемент имеет следующие атрибуты:

Имя аттрибута	**Ограничения**	**Описание**	**По умолчанию**
left	целое $-100 \le left \le 100$	Отступ от левой границы родителя	0
top	целое $-100 \le top \le 100$	Отступ от правой границы родителя	0
width	целое $0 \le width \le 100$	Ширина элемента	10
height	целое $0 \le height \le 100$	Высота элемента	10
border	булево *true* or *false*	Есть у элемента обрамление или нет	*false*
zIndex	целое $0 \le zIndex \le 100$	Элементы с более высоким zIndex отображаются поверх элементов с более низким значением zIndex	0
innerText	строка От 0 до 100 символов	Текст внутри элемента	*Пустая строка*
align	строка «center» or «left» or «right»	Расположение текста внутри элемента	«left»

Программа может содержать произвольное количество переменных, где название переменной — набор английских символов, цифр и знаков подчеркивания. Первый символ названия переменной всегда буква, названия переменных чувствительны к регистру.

Каждая переменная должна быть инициализирована перед первым использованием. Изначально единственная заданная переменная — это *document*. Есть четыре типа переменных в данной задаче: элемент, целое число, строка и булева переменная.

Для определения новой переменной используется следующая конструкция:

`var` *variable_name* = *value*;

или

`var` *variable_name* = *other_variable_name*;

Где *value* либо целое число, либо строка в двойных кавычках, либо *true* или *false*, либо вызов метода *createElement*. Это определяет однозначно тип переменной. Обратите внимание, что после выполнения следующей команды:

```
var moo = "5";
```

moo содержит строку, а не целое число.

Для изменения значения переменной используется следующая конструкция:

variable_name `=` *value*`;`

или

variable_name `=` *other_variable_name*`;`

Все элементы хранятся по ссылке, то есть выполнение следующего кода:

```
var moo = document.createElement( "span" );
var q = moo;
q.width=100;
```

bзменит ширину как *q* так и *moo* (так как *q* и *moo* указывают на один и тот же элемент).

В тоже время, следующий код:

```
var moo = document.createElement( "span" );
var q = moo;
q = document.createElement( "span" );
q.width=100;
```

уе изменит ширину *moo*, так как *q* указывает на другой элемент.

Все строки, целые числа и булевы значения хранятся по значению а не по ссылке.

Для изменения любого атрибута некоторого элемента, кроме *innerText*, используется следующая конструкция:

element`.style.`*attribute_name* `=` *value*`;`

или

element`.style.`*attribute_name* `=` *variable_name*`;`

Для изменения *innerText* используется аналогичная конструкция, но без *style*.

Все члены *style* и *innerText* также являются переменными, соответственно следующий код допустим:

```
moo.style.left = q.style.top;
```

Чтобы добавить дочерний элемент некоторому элементу, используется конструкция:

element1`.appendChild(` *element2* `);`

Где *element2* — либо переменная, содержащая элемент, либо вызов метода *createElement*, а *element1* — либо уже созданный элемент, либо *document*. *element1* не указывает на одного из потомков *element2* или на *element2* непосредственно.

Вы можете рассчитывать на то, что код содержит только перечисленные конструкции, равно как на то, что любые операции изменения атрибутов используют переменные и значения допустимых типов (например никогда не будет конструкции, пытающейся присвоить строку атрибуту *left*).

document имеет тип элемент (немного расширенный добавлением метода *createElement*), но единственными свойствами, которые у него могут быть изменены, являются *innerText* и *align*.

Команды разделяются точкой с запятой. Любое количество пробельных символов может встречаться везде, кроме названий переменных, атрибутов, методов, а также значений переменных. К пробельным символам относятся символы с ASCII кодами 9, 10, 13 and 32 (‘`\t`’, ‘`\n`’, ‘`\r`’ и ‘ ’ соответственно).

Ваша задача — выполнить программу и вывести страницу после выполнения всех операций.

Следующие правила следует выполнять, прорисовывая страницу:

1. Если два элемента (с общим родителем) пересекаются, элемент с большим значением *zIndex* рисуется на области пересечения. Если *zIndex*-ы равны, элемент, добавленный к родителю позже, рисуется на области пересечения.

2. При прорисовке элемента заменяются на пробел все символы родителя с координатами x, y, для которых $left \le x < left + width$ и $top \le y < top + height$, где $left$, top, $width$ и $height$ - соответствующие атрибуты элемента.

3. Если у элемента установлен атрибут *border*, он дополнительно обрамляется символами +, - and | (см. пример для дополнительной информации). Граница рисуется за пределами области, занятой элементом.

4. Текст пишется внутри элемента. Если длина текста превышает значение атрибута *width* элемента, он должен быть разбит на несколько строк. Для разбиения текста следуйте следующему алгоритму: если после вывода нового слова длина строки не превысит *width* символов, выведите его. Иначе начните новую строку. Если длина слова длиннее чем *width*, выведите *width* символов на одной строке и продолжите процесс для оставшейся части слова со следующей строки. Если значение атрибута *align* равно «center» и некоторая строка не может быть расположена ровно по центру (то есть ее длина не равна атрибуту *width* по модулю 2), один пробел должен быть добавлен в конец строки. Если количество строк текста больше, чем значение атрибута *height*, все лишние строки должны быть опущены. Обратите внимание, что в данной задаче значение *innerText* всегда содержит только разделенные одним пробелом последовательности английских букв и цифр.

5. Все, что не помещается в пределы родительского элемента (или экрана, если родитель элемента *document*), включая обрамление, не должно выводиться на экран.

Формат входного файла

Входной файл содержит код, который необходимо выполнить.
Код содержит не более 1000 точек с запятой.

Формат выходного файла

Выведите 22 строки, каждая по 42 символа - финальное состояние экрана с обрамлением. Смотрите пример для подробностей.

Пример

js.in

```
var el = document.createElement( "span" );
var el2 = document.createElement( "span" );
var el3 = document.createElement( "span" );
var q = true; el.style.border = q; el2.style.border = q;
el.style.width = 20; el.style.height = 15; el.style.top = 2;
el2.style.left = 5; el2.style.top = el2.style.left;
el2.style.width = 5;
el2.innerText = "abra abra abra abra abra ab ra kadabra z obama";
el3.style.left = 21; el3.style.top = 3; el3.style.height = 15;
el3.innerText = "Moo cow likes fresh grass very much";
el3.style.align = "center";
el3.style.border = true;
el.appendChild( el2 ); el.appendChild( document.createElement( "span" ) );
document.appendChild( el );
document.appendChild( el3 );
document.innerText = "This problem is not supposed to be solved";
document.style.align = "right";
el.style.zIndex = 1;
```

js.out

```
+----------------------------------------+
|      This problem is not supposed to be|
|--------------------+             solved|
|                    |----------+        |
|                    | Moo cow  |        |
|                    |  likes   |        |
|                    |  fresh   |        |
|          +         |grass very|        |
|          |         |   much   |        |
|          |         |          |        |
|          |         |          |        |
|          |         |          |        |
|          |         |          |        |
|    |ab ra|         |          |        |
|    |kadab|         |          |        |
|    |ra z |         |          |        |
|    |obama|         |          |        |
|    |     |         |          |        |
|--------------------+          |        |
|                   +----------+        |
|                                       |
+---------------------------------------+
```

Задача K. Keep Them Separately! (Division 2 Only!)

Имя входного файла:	`keepsep.in`
Имя выходного файла:	`keepsep.out`
Ограничение по времени:	3 seconds
Ограничение по памяти:	256 Mebibytes

Во время международной школы-семинара по программированию профессор Налейпиво собирается прочесть серию лекций. Всего есть k мест в аудитории — ровно по количеству участников. Для экономии времени каждый приглашённый участник имеет билет с указанием номера места. Так как профессор уделяет большое внимание конструктивному взаимодействию участников в процессе обучения, он хочет провести лекции так, чтобы для любых двух лекций было хотя бы одно место, занятое представителями разных вузов в течение этих двух лекций.

Но тут возникла проблема: существуют пары «дружественных» вузов, обладающих следующим свойством — если участники, представляющие два этих вуза, сидят рядом, то они вместо лекции начинают обсуждать, например, свежесыгранную партию в Quake, или же тонкости расположения звёзд в рейтинг-листе TopCoder, что ни в коей мере не является конструктивным взаимодействием обучаемых. Профессор хочет узнать, сколько лекций он сможет провести, прежде чем либо будет вынужден рассадить учеников «по вузам» в точности так же, как на одной из прошлых лекций, либо посадить двух учеников из «дружественных» вузов рядом.

Два места считаются соседними, если разница в их номерах равна единице.

Формат входного файла

Первая строка содержит три числа: n, m и k — количество вузов, количество пар «дружественных» вузов, и количество мест в аудитории, равное количеству участников. Далее следует m строк, каждая содержит два целых числа a_i, b_i, где a_i и b_i — номера «дружественных» вузов. Все пары различны.

$1 \le k \le 100$
$1 \le n \le 100$
$1 \le ai < bi \le n$

Формат выходного файла

Единственная строка выходного файла должна содержать единственное число — количество лекций, которое сможет провести профессор.

Примеры

keepsep.in	keepsep.out
2 1 3 1 2	2
3 2 3 1 2 1 3	9

Note: В первом примере возможные рассадки — 1-1-1 и 2-2-2. Во втором примере possible возможные рассадки — 1-1-1, 2-2-2, 2-2-3, 2-3-2, 2-3-3, 3-2-2, 3-2-3, 3-3-2 и 3-3-3.

Задача L. Numbers on the Field (Division 1 Only!)

Имя входного файла:	`numbers.in`
Имя выходного файла:	`numbers.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Профессор Налейпиво для популярного ресурса IT Happens написал интересную заметку. В качестве курсовой работы он поставил своему ученику, ни разу не посещавшему занятий, следующую задачу:

Есть поле 5×5. Вы можете размещать в ячейках поля числа от 1 до 4, следуя следующим правилам: 1 может быть помещена в любую ячейку. 2 может быть помещена в любую ячейку, у которой среди соседей есть хотя бы одна ячейка, содержащая 1. 3 может быть помещена в любую ячейку, у которой среди соседей есть хотя бы одна ячейка, содержащая 1, и хотя бы одна ячейка, содержащая 2. Наконец, 4 может быть помещена в любую ячейку, у которой среди соседей есть хотя бы одна ячейка, содержащая 1, хотя бы одна, содержащая 2 и хотя бы одна, содержащая 3. Задача разместить числа так, чтобы их сумма была максимально возможной.

Две ячейки являются соседними, если у них общее ребро.

Студент решил задачу, сдал курсовую, а в следующем году посещал все лекции профессора.

Профессор спрашивал в заметке, сможет ли кто-то из читателей решить задачу со следующим усложнением: поле в данной задаче не обязательно пустое, оно может уже содержать некоторые числа внутри. Задача — поместить некоторые числа в свободные ячейки таким образом, чтобы как новые, так и старые числа удовлетворяли условию задачи, и среди всех таких расстановок сумма чисел была максимально большой.

Попробуйте и Вы решить эту задачу. Если решений больше, чем одно, выведите любое.

Формат входного файла

Входной файл содержит пять строк по пять чисел от 0 до 4 в каждой, где 0 обозначает что ячейка пуста.

Формат выходного файла

Если расположить числа так, чтобы они удовлетворяли условию задачи, нельзя, выведите в единственной строке -1.

Иначе в первой строке выведите максимально возможную сумму. Затем выведите пять строк по пять чисел в каждой — результирующую расстановку чисел.

Примеры

numbers.in	numbers.out
1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 0 1 1 0 0 1	43 1 2 3 1 1 4 3 1 1 1 2 1 1 1 3 1 1 1 4 2 1 1 2 3 1
1 2 0 2 1 2 3 0 3 2 3 4 0 4 3 1 2 0 2 1 2 3 0 3 2	55 1 2 3 2 1 2 3 1 3 2 3 4 1 4 3 1 2 3 2 1 2 3 1 3 2