Задача 1. Переправа

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Крестьянину нужно перевезти через реку волка, козу и капусту. Но лодка такова, что в ней может поместиться только крестьянин, а с ним или один волк, или одна коза, или одна капуста. Но если оставить волка с козой, то волк съест козу, а если оставить козу с капустой, то коза съест капусту. Как перевёз свой груз крестьянин?

(Старинная головоломка)

Афанасий и Анатолий работают смотрителями в знаменитом Новосибирском Зоопарке, и им приходится решать задачи похлеще, чем спасение капусты с козой. Например, однажды им пришлось вдвоём перевезти через Обь на двух лодках всех животных зоопарка! Вам предлагается почувствовать себя в их шкуре.

Всего в зоопарке имеется N животных. Гастрономические пристрастия питомцев весьма разнообразны, и не нужно объяснять, что почти каждый из них с удовольствием слопает кого-нибудь другого, если оставить их вместе без присмотра. Про каждое животное известно, кого оно может съесть. Каждый смотритель — ас своего дела. Он может справиться даже с тигром! Когда рядом с животным находится хотя бы один смотритель, оно точно никого не может съесть.

У смотрителей есть две лодки. В связи с условиями договора о страховке, посадить двух или больше животных в одну лодку одновременно нельзя. В любой лодке в любой момент времени может находиться не более одного смотрителя и не более одного животного. Изначально все животные, смотрители и лодки находятся на левом берегу реки, и нужно перевезти их на правый берег реки. Естественно, других берегов, кроме левого и правого, у реки нет.

Смотрители могут плавать на лодках между берегами, как им вздумается. Обе лодки вёсельные, и без смотрителя двигаться не могут. Если в какой-то момент на одном берегу находится пара животных, одно из которых может съесть другого, и нет ни одного смотрителя, то происходит непоправимое. Разумеется, этого требуется избежать: все животные должны оказаться на правом берегу живыми и здоровыми.

Нужно определить, возможно ли осуществить переправу.

Формат входного файла

В первой строке входного файла записано одно целое число N — количество животных в зоопарке ($2 \le N \le 200$). Каждое животное имеет инвентарный номер в диапазоне от 1 до N включительно.

Следующие N строк содержат информацию о гастрономических пристрастиях животных. i-ая из них описывает, каких животных может съесть животное с инвентарным номером i ($1 \le i \le N$). Строка начинается с целого неотрицательного числа k_i — количества животных, которых может съесть i-ое животное, а далее записано k_i целых положительных чисел — инвентарные номера этих несчастных животных. Эти числа различны и находятся в диапазоне от 1 до N, никакое из этих чисел не может быть равно i, поскольку никакое животное не

может съесть само себя.

Общее количество пар «хищник-жертва», то есть сумма всех чисел k_i, не превосходит 1500.

Формат выходного файла

Если переправить всех животных на правый берег целыми и невредимыми можно, в выходной файл нужно вывести “`Hurrah!`”. В противном случае выведите “`Fired.`”.

Примеры

input.txt	output.txt
5 1 2 1 3 1 4 0 0	Hurrah!
5 4 2 3 4 5 4 3 4 5 1 4 4 5 1 2 4 5 1 2 3 4 1 2 3 4	Fired.

Задача 2. Список файлов

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Задан набор файлов, необходимо вывести количество файлов для каждого из расширений. Расширением файла называется последовательность символов в имени файла после символа точки.

Файловая система чувствительна к регистру. Даже если имена файлов отличаются только регистром символов, они всё равно считаются различными.

Формат входного файла

В первой строке входного файла дано целое число N — количество имён файлов ($1 \le N \le 10^3$).

Далее в каждой из следующих N строк находится имя файла длиной не больше 200 символов. Имя файла состоит только из латинских символов верхнего и нижнего регистров, цифр и символа точки ‘.’. Гарантируется, что символ точки встречается в имени файла ровно один раз. Также до и после символа точки в имени файла присутствует не менее одного символа. Гарантируется, что во входном файле каждое имя встречается ровно один раз.

Формат выходного файла

Для каждого из расширений, присутствующих во входном файле, вывести в выходной файл количество файлов с таким расширением в виде: `<расширение>:␣<количество>`. Расширения требуется выводить в порядке первого упоминания во входном файле.

Пример

input.txt	output.txt
6 218052.pdf Movie00.mkv Invoice.xls book.pdf book.epub Movie01.mkv	pdf: 2 mkv: 2 xls: 1 epub: 1

Пояснение к примеру

Два файла с расширением `pdf`: `218052.pdf`, `book.pdf`.

Два файла с расширением `mkv`: `Movie00.mkv`, `Movie01.mkv`.

Один файл с расширением `xls`: `Invoice.xls`.

Один файл с расширением `epub`: `book.epub`.

Задача 3. Оптимизация графа

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
	5 секунд (для Java)
Ограничение по памяти:	256 мегабайт

Ответ на Самый Главный Вопрос Жизни, Вселенной и Вообще: ***32***

Кондратий обожает оптимизировать код. Такое это прекрасное занятие, что хоть деньги с него бери за возможность пооптимизировать! К сожалению, он совершенно не замечает, что 90% его кода могло бы работать в сотни раз быстрее, если бы он использовал более эффективные алгоритмы вместо сочинения ассемблерных вставок.

Недавно Кондратий познакомился с девушкой по имени Всемила. Всемила обожает решать олимпиадные задачки: просто хлебом её не корми, дай написать что-нибудь «высокоалгоритмическое». Конечно же, код её решений Кондратию в основном не нравится, но он не жалуется, ведь в своей жизни он ещё не встречал девушки, которая так хорошо программирует!

Однажды между друзьями произошёл холивар о прекрасном. В результате они поспорили, кто сможет эффективнее решить первую попавшуюся в интернете задачу: Всемила с её суффиксными автоматами и потоками минимальной стоимости, или Кондратий с его ненавистью к ветвлениям и любовью к AVX-512. К сожалению для Кондратия, им попалась задача на построение графа: здесь вычислительные мощности вряд ли помогут ему, надо думать. Кондратий просит вас помочь ему решить задачу, ведь он никак не может уступить первенство в таком важном вопросе, как программирование, девушке!

Вот задача, которую решают Кондратий с Всемилой. Требуется построить ориентированный граф с n вершинами, удовлетворяющий заданному набору условий. Условия бывают двух типов:

1. Из вершины a можно добраться до вершины b.
2. Из вершины a нельзя добраться до вершины b.

Если таких графов не существует, необходимо определить это.

Формат входного файла

В первой строке входного файла дано целое число n — количество вершин в искомом графе ($1 \le n \le 10^5$).

Во второй строке записано целое число k — количество условий достижимости ($0 \le k \le 10^5$). В каждой из следующих k строк приведено по два целых числа a_i и b_i, означающих, что из вершины a_i должно быть возможно добраться до вершины b_i по рёбрам графа ($1 \le a_i, b_i \le n$).

В следующей строке дано целое число p — количество условий недостижимости ($0 \le p \le 10^5$). В каждой из следующих p строк приведено по два целых числа c_j и d_j , означающих, что из вершины c_j должно быть невозможно добраться до вершины d_j ($1 \le c_j, d_j \le n$).

Формат выходного файла

Если искомого графа не существует, то в выходной файл нужно вывести слово `NO`.

В противном случае, в первую строку требуется вывести слово `YES`, а во вторую строку — количество рёбер m ($0 \le m \le 3 \cdot 10^5$). В следующие m строк нужно выдать описания рёбер, по одному в строке, в формате "x_i y_i", где x_i — начало ребра, а y_i — его конец.

Кратные рёбра и петли разрешены. Гарантируется, что если граф, удовлетворяющий условиям из входных данных, существует, то среди таких графов найдётся граф не более чем с $3 \cdot 10^5$ рёбрами. Если решений несколько, можно вывести любое.

Примеры

input.txt	output.txt
2 1 1 2 1 1 2	NO
2 1 1 2 0	YES 1 1 2

Задача 4. ЖКХ

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Пришел как-то Ипполит Матвеевич за квартиру платить. Увидел он квитанцию, и все что смог вымолвить: «Однако...». Ну да делать нечего, достал он эту сумму, а ему из окошечка: «Маловато будет!». Бедняга — тык, мык, что к чему, а ему говорят: «А у нас теперь новый начальник будет, Остап Ибрагимович. Он там из кого-то в управдомы переквалифицировался и придумал новый, четыреста какой-то там способ сравнительно честного отнимания денег».

И выяснилось, что помимо означенной в квитанции суммы надо еще оплатить комиссионные банку за перевод средств — X_i рублей, если платишь в i-ый месяц, за каждый платеж, независимо от его суммы. Казалось бы, плати пореже, экономь... Так не тут-то было... Каждый месяц сумма всего долга увеличивается на p_i процентов. Так что, выбирай: или плати реже, тогда меньше будут платежи за комиссию, но больше за пени, или плати чаще — тогда, наоборот, за пени платить будешь меньше (а если платить каждый месяц, то и вообще ничего), а за комиссию, наоборот, больше.

Надо бы предводителю дворянства помочь с расчетами, минимизировать общую сумму уплаченных денег.

Формат входного файла

В первой строке входного файла задано одно целое число N — количество месяцев, за которые надо вносить платежи ($1 \le N \le 10^5$).

В следующих N строках записано по три целых числа S_i, x_i, p_i — соответственно, сумма основного платежа, сумма платежа за перевод средств, и процент, на который увеличится сумма долга к следующему месяцу ($10^3 \le S_i \le 10^4, 10 \le x_i \le 500, 1 \le p_i \le 10$).

Формат выходного файла

В выходной файл необходимо вывести одно вещественное число — минимально возможную сумму, которую придется уплатить за весь период, с погрешностью, абсолютной или относительной не более 10^{-8}.

Примеры

input.txt	output.txt
3 5000 20 1 5000 20 1 5000 20 1	15060
3 5000 500 1 5000 200 1 5000 100 3	15250.5

Пояснение к примеру

В первом случае выгоднее платить каждый месяц, а во втором — в самом конце за все три месяца.

Задача 5. Арифметические выражения

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Школьник Петя составляет арифметические выражения из символов `'0'-'9'`, скобок `'('` и `')'`, а также знаков `'+'` и `'-'` (плюс и минус). Каждое составленное выражение может быть либо строкой, содержащей десятичное представление целого числа без ведущих нулей, лежащего в интервале от 0 до $M-1$ включительно, либо строкой вида (E), где E тоже является выражением, либо строкой вида $E_1 \sigma E_2$, где E_1 и E_2 — это тоже выражения, а σ — это символ `'+'` или `'-'`.

Петя хочет подсчитать количество различных выражений фиксированной длины N, для которых остаток от деления на M равен P. Значение арифметического выражения вычисляется по школьным правилам арифметики. Обратите внимание: остаток от деления всегда лежит в диапазоне от 0 до $M-1$, даже если значение выражения отрицательное.

Поскольку ответ может быть большим, выведите его остаток от деления на 10^9+7.

Формат входного файла

В первой строке входного файла задано три целых числа N, M и P: N — длина выражений, M — модуль по которому вычисляются выражения, P — требуемый остаток от деления ($1 \le N \le 50$, $0 \le P < M \le 200$).

Формат выходного файла

В выходной файл необходимо вывести одно целое число — количество всех выражений, которые имеют заданную длину и значение которых имеет заданный остаток от деления. Ответ нужно вывести по модулю 10^9+7.

Примеры

input.txt	output.txt
3 100 2	12
3 100 98	8
2 100 98	1

Пояснение к примеру

Все искомые выражения для случая $N=3$, $M=100$, $P=2$:

```
(2)   0+2   1+1   2+0   2-0   3-1   4-2   5-3   6-4   7-5   8-6   9-7
```

Все искомые выражения для случая $N=3$, $M=100$, $P=98$:

```
0-2   1-3   2-4   3-5   4-6   5-7   6-8   7-9
```

Все искомые выражения для случая $N=2$, $M=100$, $P=98$:

```
98
```

Задача 6. Спутник

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Зонд «Новый Зенит» находится на круговой орбите возле вновь открытой карликовой планеты в поясе Койпера. Обзорная камера зонда зарегистрировала аномальное явление на поверхности планеты, предположительно активный криовулкан, в точке с широтой ϕ и долготой λ. Ученые требуют как можно скорее сделать повторный снимок этой области камерой высокого разрешения.

Запас топлива на борту спутника ограничен, поэтому, прежде чем рассматривать вопрос о коррекции орбиты, центр управления полетом хочет определить, когда (и если) зонд вновь пройдет над нужной точкой, двигаясь по существующей орбите. К счастью, для получения снимков приемлемого качества, не нужно точно проходить над самой точкой. Достаточно оказаться над любой точкой круга с центром в точке (ϕ, λ) и угловым радиусом δ градусов.

Зонд движется по орбите с постоянной угловой скоростью и периодом обращения t секунд.

Наклонение орбиты — это угол между плоскостью орбиты и плоскостью экватора планеты. Оно равно α градусам.

Точка восходящего узла — это точка, где орбита зонда пересекает экватор при движении с юга на север. В начальный момент времени долгота точки восходящего узла равна нулю.

Плоскость орбиты зонда проходит через центр планеты и неподвижна относительно него. Сама же планета вращается вокруг оси, перпендикулярной плоскости экватора, с постоянной угловой скоростью и с периодом обращения T.

Угловая скорость спутника постоянна в плоскости его орбиты. Угловая скорость планеты постоянна в плоскости экватора. Вращение планеты и обращение спутника прогрaдное. Это означает, что и спутник, и точки на поверхности планеты движутся с запада на восток.

В начальный момент спутник находится над точкой с широтой ϕ и долготой λ. Зонд может исполнить команду на фотографирование с высоким разрешением не ранее, чем через время $2t$ от начального момента.

Если ранее T_{max} секунд от начального момента фотографию сделать невозможно, то есть зонд не попадает в требуемую окрестность требуемой точки, то необходима коррекция орбиты.

Формат входного файла

В единственной строке входного файла задано семь вещественных чисел: t, T, T_{max}, α, ϕ, λ и δ ($60 \le t \le 1\,000$, $3\,600 \le T \le 100\,000$, $10\,000 \le T_{max} \le 1\,000\,000$, $0 \le \alpha \le 60$, $-60 \le \phi \le 60$, $0 \le \lambda < 360$, $0.1 \le \delta \le 1$). Временны́е параметры t, T, T_{max} заданы в секундах, угловые параметры $\alpha, \phi, \lambda, \delta$ заданы в градусах.

Числа разделены пробелами, у каждого числа указано не более 15 знаков после десятичной точки.

Формат выходного файла

Если в интервале времени от $2t$ до T_{max} от начального момента зонд находится на угловом расстоянии δ или меньше от точки (ϕ, λ), то в выходной файл необходимо вывести самый ранний момент времени τ, когда это происходит, с абсолютной или относительной погрешностью не более 10^{-7}. В противном случае необходимо вывести число `-1`.

Гарантируется, что при изменении δ на 10^{-3} градуса в любую сторону ответ изменяется меньше, чем на секунду.

Пример

input.txt	output.txt
120 4500 100000 60 30 19.471220634490699 0.5	8999.83111211

Комментарий

Во входном файле все числа записаны в одной строке через пробел. В тексте данного условия в примере числа приведены в двух строках, так как в одну они не вошли.

Задача 7. Голосование

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В Берляндии выборы президента происходят по сложной многоуровневой иерархической системе. Оппозиция уверена, что такая система выборов была изобретена с целью запутать население и скрыть несправедливость голосования. В этом году от оппозиции баллотируется в президенты один кандидат. Если он победит, то появится реальная возможность изменить политическую ситуацию.

Проще всего представить систему голосования в виде корневого дерева. Листья дерева — это отдельные избиратели. Каждый внутренний узел дерева — некоторая официальная социальная группа (семья, дом, район, округ и т.п.). Множество сыновей узла соответствует сущностям, которые непосредственно составляют эту группу — это отдельные избиратели и/или другие группы. Корень дерева — группа, включающая всю Берляндию.

Каждый из избирателей может либо отдать свой голос за одного конкретного кандидата, либо воздержаться, например, не придя на выборы. После того, как все избиратели так или иначе проголосовали, в каждой группе определяются локальные результаты выборов, в порядке от более мелких групп к более крупным. Для этого рассматриваются все голоса составляющих эту группу сущностей. В локальных выборах побеждает кандидат, набравший строго больше голосов, чем любой другой кандидат. Если такого кандидата нет, то локальные выборы признаются несостоявшимися. Далее результаты голосования в группе учитываются в локальном голосовании на уровень выше, как один голос от всей группы. Если локальные выборы не состоялись, то группа считается «воздержавшейся», и её мнение никак не учитывается.

Президентом Берляндии становится тот, кто победил в локальных выборах в самой верхней группе, то есть в корне дерева. Если выборы в этой группе не состоялись, то должность президента в Берляндии упраздняется до следующих выборов.

Для каждого отдельного избирателя известно, за кого он планирует голосовать. Оппозиция может изменить предпочтение любого избирателя любым образом, но это требует определённых усилий. Необходимо определить минимальное количество избирателей, мнение которых нужно изменить, чтобы президентом Берляндии стал кандидат от оппозиции.

Формат входного файла

В первой строке входного файла даны целые числа N — количество отдельных избирателей, M — количество социальных групп и K — количество кандидатов ($2 \le N \le 5\,000$, $1 \le M \le 5\,000$, $2 \le K \le 3$). Избиратели, группы и кандидаты пронумерованы по отдельности, начиная с единицы. Кандидат от оппозиции идёт под номером 1.

Во второй строке записано N целых чисел — исходные мнения избирателей. Мнение либо равно номеру кандидата, за которого избиратель планирует голосовать, либо равно нулю, если он воздерживается.

В следующих M строках описаны социальные группы. В t-ой из них описаны сущности, непосредственно составляющие t-ую социальную группу. Сначала записано целое положительное число S_t — количество этих сущностей, а затем через пробел S_t целых чисел, определяющих эти сущности ($2 \le S_t \le 20$). Если сущность является отдельным избирателем, то записан его номер, а если группой — то её номер со знаком минус. Корневая группа, соответствующая всей Берляндии, всегда имеет номер 1.

Формат выходного файла

В первую строку выходного файла необходимо вывести одно целое число R — минимальное количество избирателей, планы которых нужно изменить для победы.

В каждую из следующих R строк нужно записать по два целых числа a и v, где a — номер избирателя, планы которого нужно изменить, а v — либо номер кандидата, за которого этот избиратель должен голосовать в результате, либо 0, если избирателю не нужно приходить на выборы.

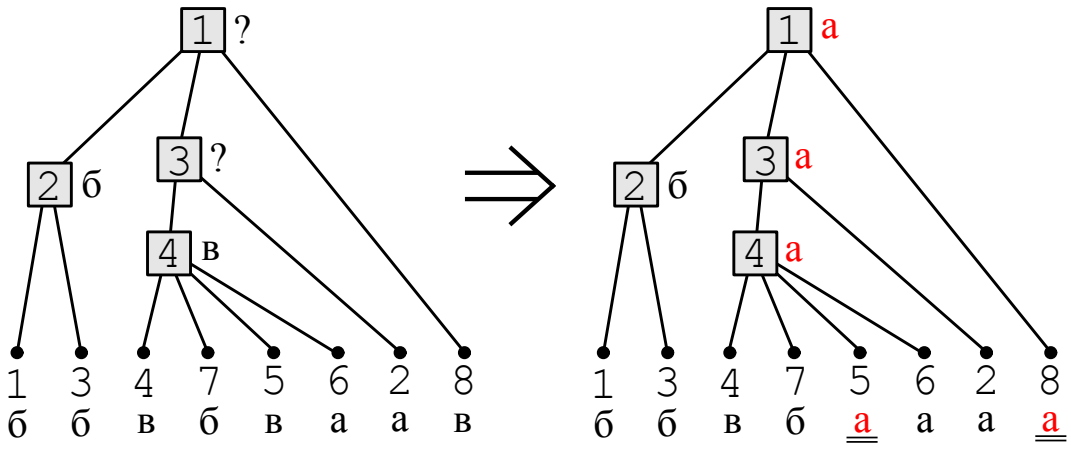
Если кандидат от оппозиции и так побеждает безо всяких изменений, в выходной файл нужно вывести одно число 0.

Пример

input.txt	output.txt
8 4 3 2 0 2 1 1 3 2 1 3 -2 -3 8 2 1 3 2 -4 2 4 4 7 5 6	0
8 4 3 2 1 2 3 3 1 2 3 3 -2 -3 8 2 1 3 2 -4 2 4 4 7 5 6	2 8 1 5 1

Пояснение к примеру

Дерево голосования и предлагаемые изменения для второго примера представлены ниже на картинке. Буквы *а*, *б*, *в* обозначают голоса, отданные за кандидатов 1, 2, 3 соответственно, а знак вопроса обозначает «воздержавшийся» голос.



Задача 8. Начинающий урбанист

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В последнее время Новый государственный университет (НГУ) строится быстрыми темпами. Постоянно то тут, то там возникают новые корпуса, здания, общежития вдоль легендарного Студенческого проспекта. Вася очень любит ходить пешком вдоль любимого проспекта, однако проблема в том, что здания НГУ находятся по разным сторонам улицы. Чтобы перейти от одного здания к другому иногда приходится долго идти до ближайшего пешеходного перехода. Поэтому он решил написать программу на компьютере, чтобы понять, как перенести пешеходные переходы вдоль Студенческого проспекта наиболее выгодным способом для пешеходов. Он хочет, чтобы как можно больше переходов находилось напротив зданий НГУ. При этом перенос должен быть минимальным по длине. Помогите написать программу Васе, ведь ему по итогам вычислений нужно ещё подготовить обращение в мэрию.

Вот как Вася схематически записал план проспекта и зданий, прежде чем писать программу. Проспект Студенческий представляется прямой линией. Пешеходные переходы считаются точками этой прямой. Все корпуса и здания НГУ стоят параллельно проспекту, поэтому можно считать их отрезками на прямой. Каждое из них задаётся своими левой и правой границами. Пешеходный переход располагается напротив какого-то здания, если соответствующая точка на проспекте находится между левой и правой границами здания, включая граничные точки. Переходы были установлены с соблюдением определённых норм, поэтому, чтобы избежать проблем, Вася решил сохранить расстояния между ними. Он хочет перенести все переходы в одну сторону на одинаковое расстояние.

Формат входного файла

В первой строке входного файла заданы два целых числа n, m, где n — количество пешеходных переходов на Студенческом проспекте, m — количество различных строений НГУ ($1 \le n \le 10^4$, $1 \le m \le 10^3$).

Во второй строке через пробел перечислены целые числа a_i, задающие координаты пешеходных переходов на проспекте ($1 \le i \le n$, $0 \le a_i \le 10^6$).

В третьей строке через пробел записаны m пар целых чисел l_i, r_i, каждая пара описывает координаты границ здания ($1 \le i \le m$, $0 \le l_i < r_i \le 10^6$).

Формат выходного файла

В выходной файл необходимо вывести два целых неотрицательных числа: расстояние, на которое нужно сдвинуть пешеходные переходы, и количество переходов, которые окажутся напротив каких-либо зданий в результате сдвига. Количество переходов должно быть максимально возможным. Если ответ при этом не определяется однозначно, необходимо дополнительно минимизировать расстояние.

Обратите внимание, что двигать переходы по проспекту можно в любом из двух направлений.

Примеры

input.txt	output.txt
3 1 1 2 4 5 6	4 2
4 2 1 6 6 1 4 5 3 5	1 2

Задача 9. Дальномер

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

На чемпионате мира по робототехнике 3175 года участникам нужно создать робота, выполняющего различные задания в лабиринте. Специально для MCLXXVI Открытой Всесибирской олимпиады им. И.В. Поттосина жюри чемпионата решило сделать виртуальную версию соревнования, в которой смогли бы участвовать программисты, ловкость рук которых не позволяет собирать роботов, достойных чемпионата мира.

Лабиринт располагается на прямоугольном участке клеточного поля размером W на H. В самих клетках лабиринта препятствий нет, там может легко находиться робот. Однако между клетками могут находиться непроницаемые стенки. Отметим, что у каждой стенки различаются две ее стороны. Лабиринт всегда полностью окружён стенками, и выбраться за его пределы робот никак не может.

Современные технологии позволяют создавать порталы в стенках. Любой портал соединяет одну сторону какой-либо стенки с одной стороной той же или другой стенки. При влетании произвольной частицы в стенку с той стороны, на которой расположен вход в портал, частица вылетает из другой стенки, на которой находится парный вход в этот портал, со стороны этого входа. Обратите внимание, что частица всегда влетает и вылетает из портала перпендикулярно его поверхности, а значит направление частицы может измениться в результате пролёта через портал. Пример пролёта через портал можно увидеть на картинке в конце условия этой задачи.

Чтобы было интереснее выполнять задания в лабиринте, конфигурация лабиринта меняется в процессе соревнования. Изначально внутри лабиринта нет ни одной стенки, хотя есть стенки вдоль всей его границы. В произвольный момент времени на поле может появиться новая стенка между какими-нибудь клетками, или вдруг исчезнуть старая стенка. Более того, может появиться портал, соединяющий две незанятые стороны стенок, или исчезнуть старый портал. При исчезновении портала стенки, на которых были расположены входы в него, остаются на месте, лишь освобождаются от входов в портал соответствующие две их стороны.

Порталы двусторонние, то есть если частица может свободно пролететь по некоторому пути, то она может также свободно пролететь по этому пути и в обратном направлении. Однако если на стенке расположен вход в портал, то зайти в него можно только с одной стороны. Более того, с обеих сторон одной стенки может быть по входу в портал, причём это могут быть как входы в разные порталы, так и парные входы в один общий портал.

Модельный робот, который будут использовать программисты в виртуальном соревновании, оснащён дальномером. Дальномер должен находиться в некоторой клетке лабиринта, и для того чтобы его использовать, его нужно направить параллельно стороне лабиринта. При применении дальномера из него вылетает луч вдоль его направления, который летит прямо до первого попадания в непроницаемую стенку. Если луч попадает в портал, то он пролетает через него по описанным выше правилам, причём попаданием в стенку это не считается. В результате использования дальномера участнику-программисту нужно сообщить, сколько клеток пролетел луч дальномера до попадания в стенку. Следует заметить, что это число может быть бесконечным, если выпущенный луч летает по кругу. Сам робот и его дальномер прозрачные и никак не влияют на полёт луча.

Жюри чемпионата обладает более чем достаточной ловкостью рук для собирания робо-

тов, однако в алгоритмическом программировании они не так сильны. Им нужно реализовать программу, которая в любой момент времени может смоделировать работу дальномера и сообщить пройденное лучом расстояние. Они смогли написать программу, которая это делает, но из-за гигантских масштабов соревнования их программа работает слишком медленно. Поэтому они просят вашей помощи.

Формат входного файла

В первой строке входного файла записаны три целых числа H, W и Q: H — высота лабиринта в клетках, W — ширина лабиринта в клетках, Q — количество запросов ($1 \le H \le 10^6$, $1 \le W \le 10^6$, $1 \le Q \le 2 \cdot 10^5$).

В каждой из остальных Q строк приведено по одному запросу. Запросы перечислены в порядке их выполнения.

Смежными называются две разные клетки, имеющие общую сторону. Для каждой клетки можно указать координаты (r, c), где r — номер строки, а c — номер столбца на поле ($1 \le r \le H$, $1 \le c \le W$). На поле можно определить четыре координатных направления, каждое из которых обозначается отдельной буквой:

- `D` — вниз, по увеличению r,
- `U` — вверх, по уменьшению r,
- `R` — вправо, по увеличению c,
- `L` — влево, по уменьшению c.

Координаты клетки (r, c) вместе с направлением d задают положение потенциальной стенки: она расположена на границе между клеткой с координатами (r, c) и смежной клеткой, в которую можно попасть, шагнув из этой клетки по направлению d. Если в этом месте стоит стенка, то можно назвать ту её сторону, с которой лежит клетка (r, c), *ближней* стороной.

Запрос может быть одного из пяти следующих видов:

- Добавление или удаление стенки: *type* r c d

  Здесь (r, c, d) вместе определяют положение потенциальной стенки по описанному выше правилу, а *type* определяет тип запроса. Если *type* равен `W+`, то в заданном месте появляется стенка, которой до этого не было. Если *type* равен `W-`, то стенка, которая стоит в заданном месте, пропадает. Гарантируется, что в момент удаления стенки ни на одной из её сторон нет входа в портал. Стенки, окружающие лабиринт, никогда не удаляются.

- Создание или удаление портала: *type* r_1 c_1 d_1 r_2 c_2 d_2

  Здесь (r_1, c_1, d_1) вместе определяют по вышеописанному правилу положение первой стенки, а (r_2, c_2, d_2) — положение второй стенки. Гарантируется, что в этих двух местах действительно есть стенки. Входы в портал располагаются на *ближних* сторонах этих стенок. Гарантируется, что эти две стороны отличаются, хотя стенки могут совпадать.

  type определяет тип запроса. Если *type* равен `P+`, то заданные две стороны стенок соединяются новым порталом, при этом гарантируется, что на этих сторонах до этого входов в портал не было. Если *type* равен `P-`, то портал между заданными двумя сторонами стенок удаляется, при этом гарантируется, что до этого был портал, соединяющий эти две стороны.

- Использование дальномера: `M?` r c d

  Дальномер ставится в клетку лабиринта с координатами (r, c), и направляется по направлению d.

Все координаты лежат в пределах лабиринта, каждое направление задаётся одной из четырёх описанных выше букв.

Формат выходного файла

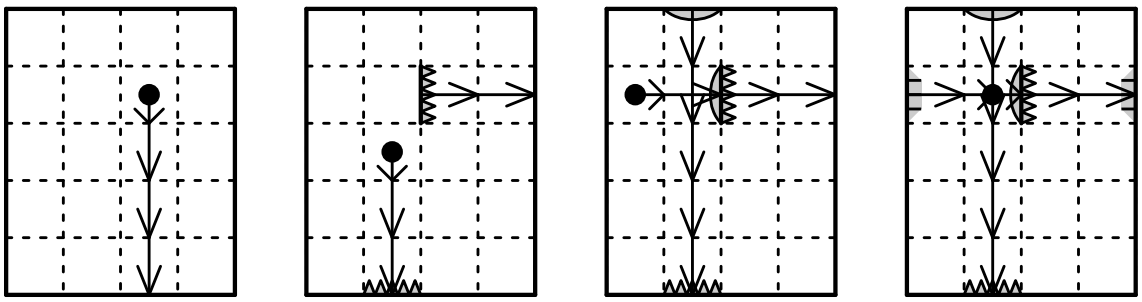
Для каждого запроса с использованием дальномера в выходной файл нужно вывести результат его использования — количество клеток, которое пролетит луч дальномера до первого попадания в стенку. Если луч будет лететь бесконечно, то в качестве ответа на запрос нужно вывести число -1 . Ответы на запросы требуется выдавать в порядке их выполнения.

Пример

input.txt	output.txt
5 4 18 M? 2 3 D W+ 2 2 R W- 2 3 L W+ 2 2 R M? 2 1 R P+ 2 3 L 5 2 D M? 3 2 D M? 2 3 L M? 2 2 R P+ 2 2 R 1 2 U M? 2 1 R P+ 2 1 L 2 4 R M? 2 2 R P- 1 2 U 2 2 R M? 2 2 L W+ 4 2 U P+ 4 2 U 3 2 D M? 5 2 U	3 1 4 5 0 8 -1 8 4

Пояснение к примеру

Ниже представлены применения дальномера: первое, третье, шестое и седьмое во входном файле.



(1): 3 (3): 4 (6): 8 (7): -1

Задача 10. Улей

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

На плоскости, замощённой правильными шестиугольниками, заданы соты пчелиного улья. Рабочие пчёлы сначала неверно поняли проектную документацию, и теперь им необходимо повернуть улей вокруг соты с пчелиной королевой на 60° по часовой стрелке.

Улей состоит из гексагональных сот. Все соты ориентированы так, что сверху и снизу расположены вершины шестиугольника, а слева и справа рёбра, которыми сота соединена с соседними сотами в ряду. Каждый следующий ряд смещён относительно предыдущего на половину соты. Ось Ox направлена вдоль горизонтального ряда сот слева направо. Ось Oy направлена вверх под углом 60° к оси Ox. Оси пересекаются в соте с координатами $(0, 0)$. Также в пояснении к примеру приложены иллюстрации, на которых изображена нумерация ячеек.

Формат входного файла

В первой строке входного файла записано три целых числа N, X и Y, где N — количество сот в улье (кроме соты королевы), X и Y — координаты соты королевы ($1 \le N \le 10^4; |X|, |Y| \le 10^4$). Далее в каждой из следующих N строк находятся пары целых чисел x и y — координаты очередной соты пчелиного улья ($|x|, |y| \le 10^4$). Координаты всех сот во входном файле различны.

Формат выходного файла

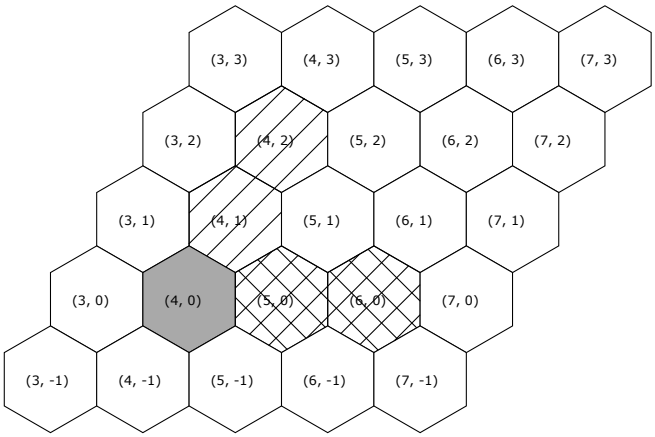
Для каждой из N сот в порядке перечисления во входном файле необходимо вывести в выходной файл на отдельной строке два целых числа — её координаты после поворота.

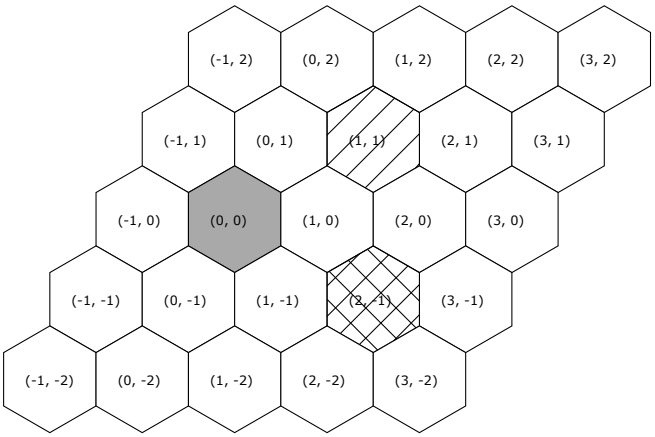
Пример

input.txt	output.txt
2 4 0 4 1 4 2	5 0 6 0
1 0 0 1 1	2 -1

Пояснение к примеру

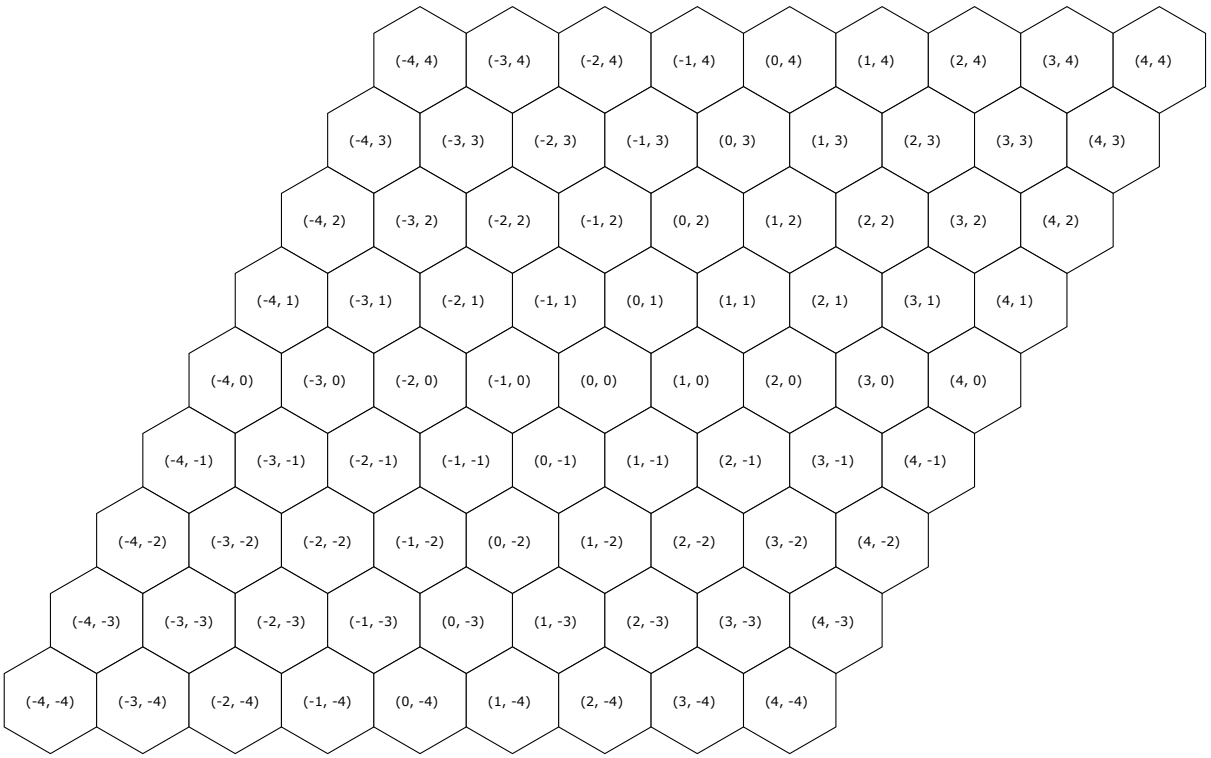
Ульи из примеров представлены на рисунках ниже. Сота пчелиной королевы обозначена серым цветом, соты в начальном положении заштрихованы параллельными прямыми, соты после поворота заштрихованы пересекающимися прямыми.





Иллюстрация

На этом улье можно рисовать. ;-)



Задача 11. Побочные эффекты

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Программист разрабатывает приложение. Поскольку он состоит в клубе «Юные друзья функционального программирования», то хочет знать, сколько из его функций содержат побочные эффекты.

В начальный момент времени про функции из программы известно, присутствуют в них побочные эффекты или нет. Также в начальный момент времени ни одна функция не вызывает другую. Но программист решает изменить логику работы приложения и начинает вставлять вызовы одних функций в другие. Новые функции программист не пишет. Если функция вызывает функцию с побочными эффектами, то вызывающая функция также начинает обладать побочными эффектами, и так далее по цепочке вызовов. Рекурсия в вызовах допустима. Также из одной функции можно вызывать другую несколько раз.

Необходимо определить, сколько функций с побочными эффектами присутствует в программе после каждого добавления вызова функции программистом.

Формат входного файла

В первой строке входного файла записано три целых числа N, K и M, где N — общее количество функций, K — начальное количество функций с побочными эффектами, M — количество вызовов функций, добавляемых программистом ($1 \le N, K, M \le 10^5; K \le N$). Функции нумеруются по порядку от 1 до N.

Далее в одной строке следуют K различных чисел от 1 до N — номера функций с побочными эффектами в начальный момент времени.

В каждой из следующих M строк находятся пары целых чисел a и b, которые означают, что программист добавил в программу вызов функции с номером b из функции с номером a ($1 \le a, b \le N$).

Формат выходного файла

Для каждого из M добавлений вызова функции в выходной файл необходимо вывести на отдельной строке количество функций с побочными эффектами на момент после добавления этого вызова.

Пример

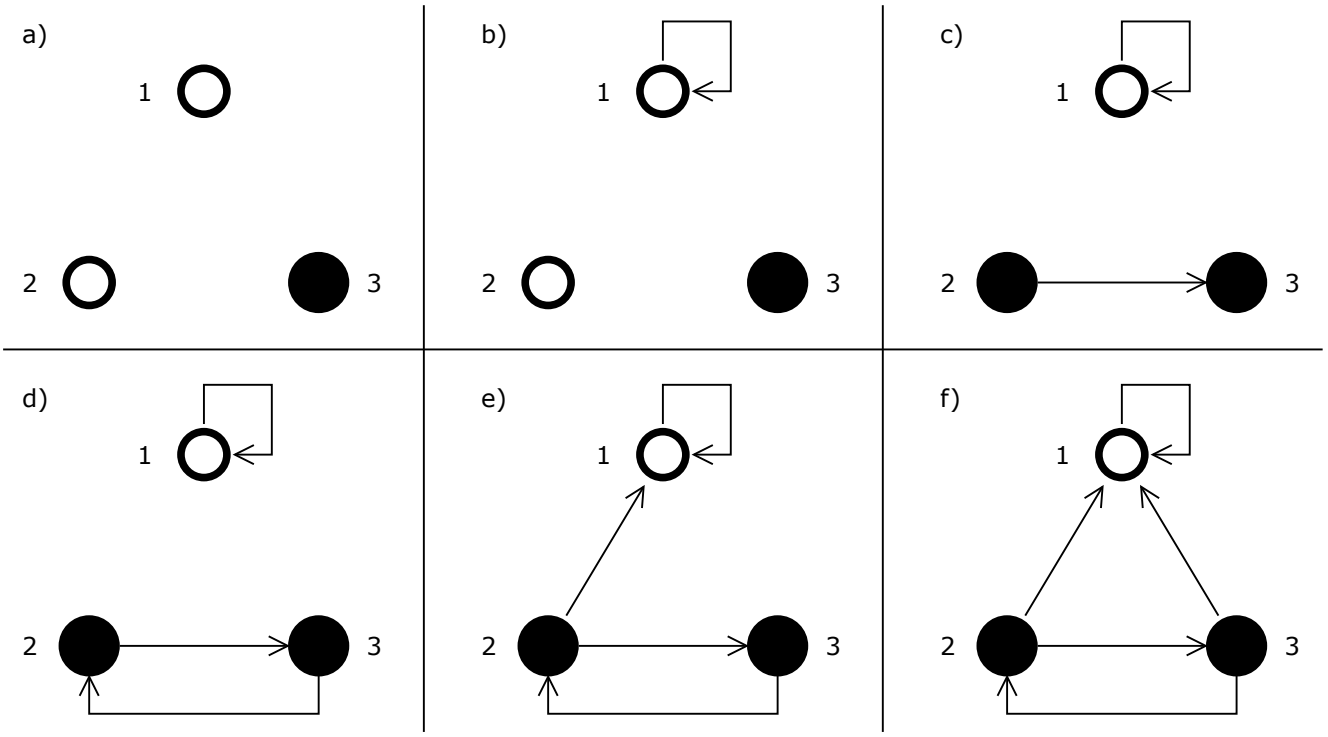
input.txt	output.txt
3 1 5 3 1 1 2 3 3 2 2 1 3 1	1 2 2 2 2

Пояснение к примеру

На рисунке `a)` изображены функции до добавления вызовов. На рисунках от `b)` до `f)` показаны последовательные добавления вызовов функций. Белый круг с черной обводкой

обозначает функцию без побочных эффектов, черный — с побочными эффектами, стрелка изображает вызов функции.

Сначала добавляется рекурсивный вызов функции 1 из самой себя, от чего, очевидно, ответ не изменяется. После добавления вызова функции 3 из функции 2 в программе становится две функции с побочными эффектами: 2 и 3, поскольку функция 3 изначально обладала побочными эффектами. Последующие добавления вызовов не увеличивают количество функций с побочными эффектами.



Задача 12. Шифрующая машина

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Боря создал новую шифрующую машину “SumUp” для личного пользования. Машине подаётся на вход целое число x, машина выполняет последовательность операций согласно записанной в ней программе, и выдаёт на выход целое число y. Аня знает внутреннюю программу машины и хочет научиться восстанавливать входное число x по выходному числу y.

Машина настроена на работу с целыми числами в диапазоне от 0 до $M-1$ для некоторого целого положительного числа M. У машины есть k регистров, которые нумеруются индексами от 0 до $k-1$, и обозначаются `R0`, `R1`, `R2` и так далее. Перед началом работы программы значение регистра `R0` равно входному числу x, а все остальные регистры имеют значение ноль. После выполнения всех операций программы содержимое регистра `R0` выдаётся из машины в качестве выходного числа y.

Программа состоит из последовательности команд, которые выполняются в порядке их записи. Есть только два вида команд:

1. `add` *регП* *регИ* — прибавить значение регистра-источника *регИ* к значению регистра-приёмника *регП*.
2. `inc` *регП* — увеличить значение регистра-приёмника *регП* на единицу.

В любой момент значение любого регистра является целым числом в пределах от 0 до $M-1$ включительно. Если при выполнении команды происходит переполнение, то в регистр-приёмник записывается остаток от деления результата на M. Входное число x должно лежать в пределах от 0 до $M-1$, иначе машина работать не будет. Разумеется, выходное число y тоже всегда лежит в этих пределах.

Совершенно случайно, получив физический доступ к машине, Аня успела сфотографировать используемый в машине текст программы. Однако число M, обозначающее рабочий диапазон машины, на фотографию не попало. Аня лишь помнит, сколько цифр использовалось в этом числе, а также некоторые из этих цифр. Кроме того, Ане известно выходное значение y, открыто распространяемое Борей.

Аня использует полный перебор для решения возникшей перед ней проблемы. Она перебирает все возможные числа M и x. Для каждой пары она проверяет, получится ли y в качестве выходного значения, если установить рабочий диапазон машины M и подать на вход x. Каждый раз, когда значение y успешно получается, она дописывает число x себе на бумажку.

От вас требуется посчитать сумму всех чисел, записанных на бумажке Аней, после завершения перебора. Поскольку эта сумма может быть большой, нужно вывести остаток от деления суммы на простое число 10^9+7.

Формат входного файла

Во входном файле записано несколько независимых тестов. В первой строке записано T — количество тестов в файле ($1 \le T \le 10$). В оставшейся части файла идут описания T тестов.

Каждое описание начинается со строки с тремя целыми числами k, m и y, где k — количество регистров в машине, m — количество команд в программе, y — выходное значение, которые нужно получить ($1 \le k \le 10$, $0 \le m \le 15$, $0 \le y < 10^{15}$).

В следующих m строках приведены команды программы в порядке их выполнения, по одной команде в строке.

Каждая команда состоит из двух или трёх частей, части отделены друг от друга одним пробелом. Первая часть обозначает тип команды (`add` для прибавления, `inc` для инкремента), вторая — регистр, значение которого изменяется, и третья, если есть, — регистр, значение которого прибавляется. Регистр обозначается латинской буквой `R`, сразу после которой идёт индекс регистра в диапазоне от 0 до $k - 1$ включительно.

В последней строке дана информация о числе M. В ней записана строка, состоящая только из десятичных цифр и знаков вопроса. Знак вопроса означает, что на его месте в записи числа M может стоять любая десятичная цифра. Строка непустая, длина строки не превосходит 15 символов. Гарантируется, что первый символ в строке — цифра, отличная от нуля.

Формат выходного файла

Для каждого теста, записанного во входном файле, нужно вывести в выходной файл одно целое число: сумму чисел, выписанных Аней на бумажке, по модулю $10^9 + 7$. Если на бумажку не записано ни одного числа, необходимо вывести 0 в качестве ответа на тест.

Ответы для тестов выводите в том же порядке, в котором тесты заданы во входном файле.

Пример

input.txt	output.txt
4 1 1 17 inc R0 3? 2 1 17 inc R0 1? 2 2 10 inc R1 add R0 R0 2? 2 2 1 add R1 R0 add R0 R1 1?3	160 32 135 745

Пояснение к примеру

В первом тесте программа увеличивает на 1 значение входного числа x. Рабочий диапазон M колеблется в диапазоне от 30 до 39. Легко видеть, что при любом из этих значений M входное число x должно быть равно 16.

Второй тест совпадает с первым, только M принимает значения от 10 до 19. Поскольку выходное число y должно входить в рабочий диапазон, только $M = 18, 19$ подходят.

В третьем тесте значение входного числа x удваивается, а команда инкремента никак не влияет на выходное значение. Для любого из десяти значений M входное значение $x = 5$ подходит. Кроме того, для любого чётного M также подходит $x = 5 + \frac{M}{2}$.