Задача A. ЗОРП 1

Имя входного файла:	`knapsack-1.in`
Имя выходного файла:	`knapsack-1.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Перед вами лежат n котиков. Каждый котик характеризуется своим весом w_i и своей мимимишностью c_i. Вы хотите выбрать некоторое число котиков суммарным весом не более чем S так, чтобы их суммарная мимимишность была максимально возможной.

Формат входных данных

В первой строке содержатся два целых числа n и S — число котиков и максимальный допустимый суммарный вес ($1 \leqslant n \leqslant 100$, $1 \leqslant S \leqslant 10^4$). Следующие n строк содержат по два целых числа w_i и c_i — вес и мимишность i-го котика ($1 \leqslant w_i \leqslant 10^4$, $0 \leqslant c_i \leqslant 10^7$).

Формат выходных данных

В первой строке выведите суммарную мимишность выбранных котиков. Во вторую строку выведите целое число k — количество выбранных котиков. В третьей строке выведите k чисел — номера выбранных котиков. Если оптимальных ответов несколько, то разрешается вывести любой из них.

Примеры

knapsack-1.in	knapsack-1.out
3 10 1 2 4 3 8 9	11 2 3 1

Задача B. ЗОРП 2

Имя входного файла:	`knapsack-2.in`
Имя выходного файла:	`knapsack-2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Перед вами лежат n котиков. Каждый котик характеризуется своим весом w_i и своей мимимишностью c_i. Вы хотите выбрать некоторое число котиков суммарным весом не более чем S так, чтобы их суммарная мимимишность была максимально возможной.

Формат входных данных

В первой строке содержатся два целых числа n и S — число котиков и максимальный допустимый суммарный вес ($1 \leqslant n \leqslant 100$, $1 \leqslant S \leqslant 10^9$). Следующие n строк содержат по два целых числа w_i и c_i — вес и мимишность i-го котика ($1 \leqslant w_i \leqslant 10^7$, $0 \leqslant c_i \leqslant 10^4$). Гарантируется, что сумма всех c_i не превосходит 10^4.

Формат выходных данных

В первой строке выведите суммарную мимишность выбранных котиков. Во вторую строку выведите целое число k — количество выбранных котиков. В третьей строке выведите k чисел — номера выбранных котиков. Если оптимальных ответов несколько, то разрешается вывести любой из них.

Примеры

knapsack-2.in	knapsack-2.out
3 10 1 2 4 3 8 9	11 2 3 1

Задача C. Программирование вслепую

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам нужно решить *задачу о сумме подмножеств*, вариант задачи о рюкзаке. У вас есть n предметов, имеющих массу w_i грамм, и рюкзак вместимостью c грамм. Нужной найти подмножество предметов максимальной суммарной массы, не превосходящей c.

Слишком просто? Но это нужно делать вслепую!

В начале вам даются только два числа n и c — количество предметов и вместимость рюкзака соответственно. Тестирующей системе известны положительные веса w_i и две битовые строки A и B, каждая содержит n бит. Каждая строка представляет из себя возможное решение задачи: если i-й бит равен 1, тогда i-й предмет участвует в решении. A хранит предыдущий выбранный кандидат на решение, B — новый предложенный кандидат на решение, полученный инвертированием случайного бита в A. Вы можете одобрить или отклонить это предложение. Ваша задача — остановиться, когда B будет соответствовать оптимальному решению.

Изначально A инициализировано каким-то значением, которое вам не известно. На каждом шаге A копируется в B, и затем бит на случайной позиции в B инвертируется. Вам даётся масса, соответствующая B, равная $\sum\limits_{i=1}^{n} w_i \cdot B(i)$, где $B(i)$ равен i-му биту в B.

В ответ, программа может выполнять одну из следующих трех действий:

- `stop` — финальное действие, после его выполнения ваша программа должна завершиться;
- `accept` — скопировать значение B в A;
- `decline` — проигнорировать B.

После каждого не финального действия начинается новый шаг. Вы можете выполнить не более 1000 действий.

Протокол взаимодействия

При запуске программы на вход подаются три целых числа n, c и масса, соответствующая B ($1 \leqslant n \leqslant 20, 0 \leqslant c \leqslant 10^9$).

Ваша программа должна вывести выполняемое действие в стандартный поток вывода и после ответа проверяющей системы считать массу, соответствующую B, и так далее.

Программа должна завершиться сразу после выполнения действия `stop`. После вывода каждой команды необходима выводить символ перевода строки и сбрасывать поток стандартного вывода.

Гарантируется, что $1 \leqslant w_i \leqslant 10^8$.

Примеры

Выполненная операция	Ответ тестирующей системы	Значение A	Значение B
Запуск	`2 5 3`	`00`	`01`
`decline`	`2`	`00`	`10`
`accept`	`5`	`10`	`11`
`stop`		`11`	

Веса предметов в примере — 2 и 3.

Задача D. Сокровища

Имя входного файла:	`dowry.in`
Имя выходного файла:	`dowry.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дочь короля Флатландии собирается выйти за прекрасного принца. Принц хочет подарить принцессе сокровища, но он не уверен какие именно бриллианты из своей коллекции выбрать.

В коллекции принца n бриллиантов, каждый характеризуется весом w_i и стоимостью v_i. Принц хочет подарить наиболее дорогие бриллианты, однако король умен и не примет бриллиантов суммарного веса больше R. С другой стороны, принц будет считать себя жадным всю оставшуюся жизнь, если подарит бриллиантов суммарным весом меньше L.

Помогите принцу выбрать набор бриллиантов наибольшей суммарной стоимости, чтобы суммарный вес был в отрезке $[L, R]$.

Формат входных данных

Первая строка содержит число n ($1 \leqslant n \leqslant 32$), L и R ($0 \leqslant L \leqslant R \leqslant 10^{18}$). Следующие n строк описывают бриллианты и содержит по два числа — вес и стоимость соответствующего бриллианта ($1 \leqslant w_i, v_i \leqslant 10^{15}$).

Формат выходных данных

Первая строка вывода должна содержать k — количество бриллиантов, которые нужно подарить принцессе. Вторая строка должна содержать номера даримых бриллиантов.

Бриллианты нумеруются от 1 до n в порядке появление во входных данных.

Если составить подарок принцессе невозможно, то выведите 0 в первой строке вывода.

Примеры

dowry.in	dowry.out
3 6 8 3 10 7 3 8 2	1 2

Задача E. Песочные часы

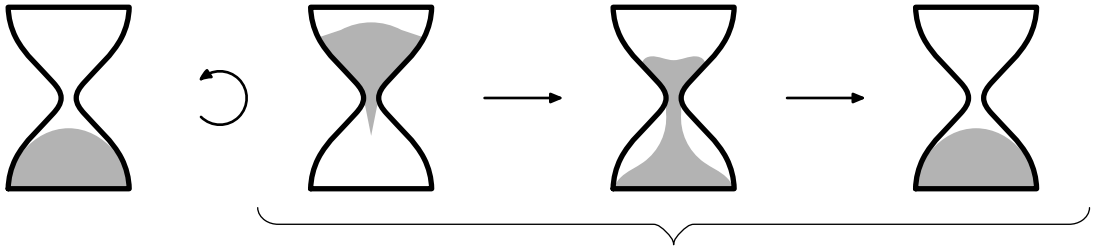
Имя входного файла:	`clocks.in`
Имя выходного файла:	`clocks.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	4 мегабайта

Обратите внимание на нестандартное ограничение по памяти в этой задаче.

Каждый раз, приходя из школы, Петя разогревает себе суп. Петя давно установил, что для достижения оптимальной температуры, суп надо греть в течении ровно t минут. Однажды у Пети в часах села батарейка. И тут неожиданно выяснилось, что это были единственные часы в доме.

Порывшись на чердаке, Петя нашел двое старых песочных часов — на a и на b минут соответственно. Каждые песочные часы состоят из двух половинок, одна из которых исходно заполнена песком.

Для того, чтобы использовать часы, их ставят на одно из оснований, при этом песок из верхней половины начинает постепенно пересыпаться в нижнюю.



Песок полностью пересыпается из верхней половины в нижнюю за a минут в первых часах и за b минут во вторых

Песок пересыпается равномерно и с одинаковой скоростью, вне зависимости от количества песка, оставшегося в верхней половине. В первых часах весь песок пересыпается за a минут, во вторых — за b минут.

В тот момент, когда Петя ставит суп на огонь, весь песок в каждых часах находится в нижней половине. В этот момент Петя может перевернуть какие-либо часы, либо и те и другие сразу. Далее Петя может переворачивать часы в момент, когда в одних из них заканчивает пересыпаться песок. В один из таких моментов Петя должен снять суп с плиты.

Петя хочет узнать, как ему действовать, чтобы снять суп с плиты ровно через t минут.

Формат входных данных

Во входном файле заданы целые числа a, b и t ($1 \leqslant a, b \leqslant 500$, $1 \leqslant t \leqslant 10^5$).

Формат выходных данных

Выведите последовательность инструкций для Пети. Каждая инструкция — это пара `<событие>: <действие>`.

События бывают трех типов:

- `Initially` — начальный момент времени, в последовательности должна быть ровно одна инструкция, помеченная этим событием, она должна быть первой;
- `When A stops` — когда заканчивает пересыпаться песок в первых часах;
- `When B stops` — когда заканчивает пересыпаться песок во вторых часах.

Действия бывают четырех типов:

- `flip A` — перевернуть первые часы;
- `flip B` — перевернуть вторые часы;
- `flip A and B` — перевернуть и те и другие часы;
- `ready` — снять суп с плиты, инструкция с таким действием должна быть ровно одна, она должна быть последней.

Если подогреть суп с использованием этих песочных часов не удастся, выведите в выходной файл одно слово — “`Impossible`”.

Примеры

clocks.in	clocks.out
5 7 9	Initially: flip A and B When A stops: flip A When B stops: flip A and B When A stops: ready
2 4 11	Impossible

Задача F. Редукция дерева

Имя входного файла:	`tree.in`
Имя выходного файла:	`tree.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Задано неориентированное дерево, содержащее n вершин. Можно выбрать некоторое ребро и удалить его, при этом инцидентные ему вершины не удаляются. Таким образом можно удалить из дерева некоторый набор рёбер. В результате дерево распадается на некоторое количество меньших деревьев. Требуется, удалив наименьшее количество рёбер, получить в качестве хотя бы одной из компонент связности дерево, содержащее ровно p вершин.

Формат входных данных

Первая строка входного файла содержит пару натуральных чисел n и p ($1 \leqslant p \leqslant n \leqslant 1000$). Далее в $n-1$ строке содержатся описания рёбер дерева. Каждое описание состоит из пары натуральных чисел a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$) — номеров соединяемых ребром вершин.

Формат выходных данных

В первую строку выведите наименьшее количество рёбер q в искомом наборе.

Примеры

tree.in	tree.out
11 6 1 2 1 3 1 4 2 6 2 7 1 5 2 8 4 9 4 10 4 11	2