Задача A. Проверьте правильность ситуации

Имя входного файла:	xzeros.in
Имя выходного файла:	xzeros.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

Напишите программу, которая по изображению поля для игры в «Крестики-нолики» определит, могла ли такая ситуация возникнуть в результате игры с соблюдением всех правил.

Напомним, что игра в «Крестики-нолики» ведется на поле 3×3. Два игрока ходят по очереди. Первый ставит крестик, а второй – нолик. Ставить крестик и нолик разрешается в любую еще не занятую клетку поля. Когда один из игроков поставит три своих знака в одной горизонтали, вертикали или диагонали, или когда все клетки поля окажутся заняты, игра заканчивается.

Формат входных данных

Вводится три строки по три числа в каждой, описывающих игровое поле. Число 0 обозначает пустую клетку, 1 – крестик, 2 – нолик. Числа в строке разделяются пробелами.

Формат выходных данных

Требуется вывести слово YES, если указанная ситуация могла возникнуть в ходе игры, и NO в противном случае.

Примеры

xzeros.in	xzeros.out
1 1 1 1 1 1 1 1 1	NO
2 1 1 1 1 2 2 2 1	YES
1 1 1 2 0 2 0 0 0	YES
0 0 0 0 1 0 0 0 0	YES
1 1 1 2 2 2 0 0 0	NO

Задача B. Найдите последовательность

Имя входного файла:	words.in
Имя выходного файла:	words.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

Вася и Петя играют в следующую игру. Они взяли некоторую последовательность символов и дальше получают из нее новые последовательности, отбрасывая несколько первых символов исходной последовательности (разрешается в том числе не отбрасывать ни одного символа, но не разрешается отбрасывать сразу все символы). Каждый называет по одной такой последовательности. Выигрывает тот, чья последовательность будет идти раньше в алфавитном порядке.

Напомним, что если мы сравниваем две последовательности, и у них первые K символов совпадают, а (K+1)-е символы отличаются, то раньше будет идти по алфавиту та, в которой (K+1)-й символ идет раньше по алфавиту. Если же одна последовательность является началом другой, то раньше по алфавиту идет более короткая из них.

Напишите программу, которая по данной последовательности определит, что нужно назвать Васе, чтобы не проиграть Пете.

Формат входных данных

Во первой строке входного файла записано число N — длина исходной последовательности ($1 \le N \le 1000$). Во второй строке идет сама последовательность. Последовательность состоит только из заглавных латинских букв.

Формат выходных данных

В выходной файл выведите выигрышную последовательность.

Примеры

words.in	words.out	Комментарии
4 MAMA	A	Рассматриваются строки MAMA, AMA, MA, A. Выигрышная строка A
4 ALLO	ALLO	Выигрышной является исходная строка
5 BBABB	ABB	

Задача C. Округлите равенство

Имя входного файла:	round.in
Имя выходного файла:	round.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

Дано верное равенство вида $a_1+a_2+...+a_N=b_1+b_2+...b_M$, где $a_1,a_2,...,a_N,,b_1,b_2,...,b_M$ – некоторые действительные (не обязательно целые) числа. Требуется «округлить» это равенство, т.е. получить новое верное равенство $c_1+c_2+...+c_N=d_1+d_2+...+d_M$, где $c_1,c_2,...,c_N,d_1,d_2,...,d_M$ — целые числа, и при этом c_1 получено округлением числа a_1 до целого вверх или вниз (так, например, число 1.7 разрешается округлить как до 1, так и до 2), c_2 получено округлением a_2, …, c_N – округлением a_N, d_1 – округлением b_1, …, d_M – округлением b_M. Если какое-то из чисел в исходном равенстве было целым, оно должно остаться без изменений.

Формат входных данных

Во входном файле задано сначала число N, затем N чисел $a_1, a_2, ..., a_N$, затем число M, затем числа $b_1, b_2, ..., b_M$. Каждое число задается на отдельной строке. M и N – натуральные числа, не превышающие 1000. Остальные числа — вещественные, каждое из них по модулю не превышает 1000 и содержит не более 6 цифр после десятичной точки. При этом $a_1+a_2+...+a_N=b_1+b_2+...b_M$.

Формат выходных данных

Если «округлить» равенство можно, то в выходной файл выведите сначала числа $c_1,c_2,...,c_N$, а затем числа $d_1,d_2,...,d_M$. Все числа должны быть целыми и выведены без десятичной точки. Числа должны разделяться пробелами или переводами строки. Если решений несколько, выведите любое из них.

Если округлить исходное равенство до верного целочисленного равенства невозможно, выведите одно число 0.

Примеры

round.in	round.out	Комментарии
3 0.15 -3.000 2.7 1 -0.15	0 -3 2 -1	Обратите внимание, что число –3 может округляться только в –3, в то время как 0.15 можно округлить как до 0, так и до 1, 2.7 – до 2 или до 3, –0.15 – до –1 или до 0. Приведенное решение не является единственным: так же верным является, например, такое округление: 1+(–3)+2=0
2 1.7 2.5 3 1 2.000 1.20	1 3 1 2 1	Приведенное решение 1+3=1+2+1 не является единственным. Верными ответами также являются 2+2=1+2+1 и 2+3=1+2+2.
1 0.5 1 0.5	1 1	Здесь верными являются как ответ 1=1, так и 0=0.

Задача D. Определите порядок действий

Имя входного файла:	order.in
Имя выходного файла:	order.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

Ученику второго класса рассказали правила, как нужно выполнять арифметические действия, чтобы вычислить значение арифметического выражения, состоящего из чисел, скобок и знаков арифметических операций + (сложение) и * (умножение). После этого ему дали упражнения — несколько задач, в которых требуется расставить порядок выполнения действий. Помогите ему.

Правила вычисления выражения, рассказанные ученику, звучат так. Если в выражении вообще нет скобок, то сначала выполняются все операции умножения слева направо, а затем — операции сложения также слева направо. Если же в выражении есть скобки, то находится самая левая пара скобок (открывающая и закрывающая), содержащая внутри себя бесскобочное выражение, которое может быть вычислено по вышеописанным правилам. Дальше это выражение (вместе со скобками) мысленно удаляется из выражения и заменяется числом – результатом. Если в выражении остались скобки, то процедура повторяется.

Напишите программу, которая для корректного выражения будет определять порядок выполнения арифметических действий. Поскольку сами числа в этой задаче нам будут не существенны, мы заменим их на знаки #.

Формат входных данных

Во входном файле записана одна строка, состоящая из символов #, +, *, (,). Длина строки не превышает 250 символов. Строка соответствует правильному арифметическому выражению.

Формат выходных данных

В выходной файл нужно вывести ту же строку, заменив знаки операций + и * в ней натуральными числами, задающими порядок выполнения действий в соответствии с описанными правилами.

Примеры

order.in	order.out
#+#*#	#2#1#
#+#+(#+#)	#2#3(#1#)
#+(#+#*#)*#+#	#4(#2#1#)3#5#
#+#+#+#+#+#+#+#+#+#	#1#2#3#4#5#6#7#8#9#10#
#+#+(#+(#+#))+(#+#)	#4#5(#2(#1#))6(#3#)

Задача E. Сложите конфетки

Имя входного файла:	candy.in
Имя выходного файла:	candy.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

Васю угостили конфетами, а он решил частью конфет поделиться со своим младшим братом Петей. Однако он хочет разделить конфеты не поровну, а по-братски.

Для этого Вася решил сыграть сам с собой в следующую игру. Он разложил конфеты на столе в несколько кучек, которые расположил в ряд. Если кучек не меньше двух, то из первой и последней в этом ряду кучек он выбирает ту, в которой меньше конфет. Пусть в наименьшей кучке оказалось B конфет. Тогда он B конфет перекладывает из первой кучки во вторую, и также B конфет перекладывает из последней кучки в предпоследнюю. При этом, естественно, одна из кучек (или даже две, если в первой и последней кучках конфет было поровну) становится пустой, и Вася забывает про ее существование. Он повторяет эти операции до тех пор, пока на столе не останется одна или две кучки.

Если останется одна кучка, то Вася все конфеты съест сам, а если две — то конфеты из первой кучки он съест сам, а из второй кучки отдаст Пете.

Напишите программу, которая по исходному распределению конфет в кучках определит, чем закончится Васина игра.

Формат входных данных

Начальное расположение кучек конфет будет описываться K парами чисел A_i, N_i, которые обозначают, что на столе выложено подряд N_i кучек конфет, по A_i конфет в каждой.

Во входном файле задано сначала число K ($1 \le K \le 10^5$). Затем идет K пар чисел A_i, N_i ($1 \le A_i \le 100$, $1 \le N_i \le 10^8$). Общее количество кучек так же не превысит 10^8.

Формат выходных данных

В выходной файл выведите сначала 1 или 2 — количество кучек конфет, которые останутся в конце игры. Затем выведите соответственно одно или два числа — количества конфет в оставшихся кучках.

Примеры

candy.in	candy.out	Пояснения
3 2 2 3 2 2 1	2 11 1	Исходно конфеты расположены в следующих кучках: 2 кучки по 2 конфеты, две кучки из 3-х конфет и одна из 2-х: 2 2 3 3 2 Далее по 2 конфеты перекладывается из 1 и 5 кучек во 2 и 4, при этом и 1 и 5 кучки становятся пустыми, т.е. на столе остается только 3 кучки: 4 3 5 Теперь по 4 конфеты перекладываются из 1 и 3 кучек во 2-ю и на столе остается две кучки, игра заканчивается с результатом 11 1
1 1 7	1 7	Изначально у нас 7 кучек по 1 конфете в каждой. После первого перекладывания получится следующая конфигурация: 2 1 1 1 1 2 После второго: 3 1 3 После третьего останется одна кучка с 7 конфетами
5 1 1 2 1 3 1 4 1 5 2	2 15 5	Изначально имеется 6 кучек: 1 2 3 4 5 5 После первого перекладывания получим: 3 3 4 6 4 После второго: 6 4 9 1 После третьего: 5 5 10 Наконец, получим: 15 5

Задача F. Упорядочите шарики

Имя входного файла:	sort.in
Имя выходного файла:	sort.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

При игре в новую игру (некоторый гибрид боулинга и бильярда) используется N шариков, пронумерованных числами от 1 до N. В начале игры эти шарики должны быть выложены в линию в порядке своих номеров. В процессе игры их порядок может меняться.

Для того, чтобы упорядочить шарики перед началом следующей партии, используется следующее устройство. Это устройство состоит из головки, которая, перемещаясь над шариками, может «засасывать» и «выплевывать» шарики. Чтобы получить большее представление об этом устройстве, представьте себе пылесос, который может засасывать шарики, перемешаться в нужное место, и там, включаясь на продув в обратном направлении, шарики «выплевывать».

При засасывании шарика все шарики, которые находились правее засасываемого, сдвигаются влево. «Выплюнуть» шарик можно между любыми двумя шариками (а также перед первым шариком или после последнего), тогда выплевываемый шарик вставляется между этими шариками, и все шарики, которые находятся правее вставляемого, сдвигаются вправо.

В устройство может быть одновременно засосано больше одного шарика, при этом при выплевывании шарика первым выплевывается последний засосанный шарик, затем - предпоследний и т.д. (т.е. устройство работает по принципу стека). Шарики выплевываются по одному, т.е. можно выплюнуть только один шарик, остальные оставив внутри устройства (при этом дальше можно как продолжать «выплевывать» шарики в том же или в другом месте, так и засасывать новые шарики).

Наиболее энергоемкой из описанных операций является операция засасывания шарика, поэтому хочется минимизировать количество именно таких операций.

Напишите программу, которая по данному начальному расположению шариков определит минимальное количество операций засасывания, которое нужно, чтобы расположить шарики в порядке их номеров.

Формат входных данных

Во входном файле задано сначала число N — количество шариков ($1 \le N \le 1000$). Далее идет N чисел, задающих номера шариков в порядке слева направо в их текущем расположении (каждое число — от 1 до N, и каждое из чисел встречается в последовательности ровно один раз).

Формат выходных данных

В выходной файл выведите одно число — минимальное количество операций засасывания шарика, которое потребуется, чтобы расположить шарики в порядке их номеров.

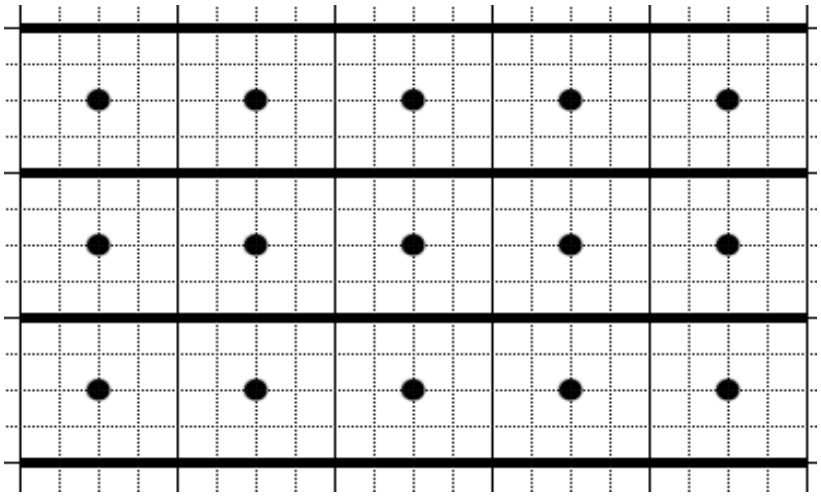
Примеры

sort.in	sort.out	Комментарии
3 2 1 3	1	Можно засосать, например, шарик номер 2 и выплюнуть его между 1-м и 3-м шариком.
4 4 3 2 1	3	Можно действовать, например, так. Сначала засосем шарик номер 1, затем – шарик номер 2. Затем переместимся в начало и перед 4-м шариком выплюнем шарик (это будет шарик номер 2). Дальше засосем шарик номер 3, и выплюнем его между шариками 2 и 4. Дальше переместимся в начало и там выплюнем шарик номер 1. Впрочем, это не единственный возможный вариант упорядочения шариков в этом примере.

Задача G. Посчитайте лампы

Имя входного файла:	lamps.in
Имя выходного файла:	lamps.out
Максимальное время работы на одном тесте:	**2 секунды**
Максимальный объем используемой памяти:	64 мегабайта

На новой станции метро, которую планируют открыть в конце этого года, будет N эскалаторов (эскалаторы пронумерованы подряд числами от 1 до N). Эскалаторы имеют длину L и расположены на расстоянии H друг от друга. Шириной эскалатором пренебрежем. Между каждыми двумя соседними эскалаторами (точно посередине) будет установлен ряд ламп. В ряду будет K ламп. Лампы устанавливаются по следующему принципу: всю длину эскалатора L разбивают на K равных отрезков и в середине каждого отрезка устанавливают по лампе (см. рисунок). Всего будет установлено $(N–1)*K$ ламп.



На приведенном рисунке N=4 (эскалаторы показаны жирными горизонтальными линиями), L=20, H=4, K=5.

Васе удалось проникнуть на эту станцию еще до ее открытия, и даже прокатиться на эскалаторе. Он выбрал эскалатор номер J. Посчитайте, в скольких точках эскалатора (включая его начало и конец) Вася будет видеть не все лампы (так как их будут загораживать другие лампы).

Формат входных данных

Во входном файле записаны числа N, L, H, K, J. Все числа — натуральные. $2 \le N \le 35$, $1 \le L \le 1000$, $1 \le H \le 1000$, $1 \le K \le 35$, $1 \le J \le N$.

Формат выходных данных

В выходной файл выведите одно число — ответ задачи.

Примеры

lamps.in	lamps.out
2 20 4 5 1	0
4 20 4 5 2	11

Задача H. Введите одностороннее движение

Имя входного файла:	oneway.in
Имя выходного файла:	oneway.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

В Тридевятом Царстве было N городов, некоторые из которых были соединены дорогами. К сожалению, в последнее время добраться из одного города в другой стало очень сложно из-за возникших автомобильных пробок. В целях борьбы с пробками было решено все дороги сделать односторонними, т.е. разрешить проезд по каждой дороге только в одном направлении. При этом требуется, чтобы по-прежнему можно было из любого города попасть в любой другой.

Формат входных данных

Во входном файле записано сначала число N — количество городов ($1 \le N \le 1000$). Затем записано число M — количество дорог ($1 \le M \le 100000$). Далее идет M пар чисел, задающих дороги (каждая дорога описывается номерами городов, которые она соединяет). Не бывает дорог из некоторого города в тот же город. Между двумя городами может быть несколько дорог. Гарантируется, что до введения одностороннего движения можно было попасть из любого города в любой другой.

Формат выходных данных

В выходной файл нужно выдать M пар чисел, соответствующих дорогам (дороги должны быть выданы в том же порядке, в котором они заданы во входном файле). Для каждой дороги сначала должен быть записан номер города, из которого по ней можно будет уехать после введения одностороннего движения, а затем — номер города, куда эта дорога ведет.

Если ввести одностороннее движение так, чтобы можно было из любого города попасть в любой другой, нельзя, выходной файл должен содержать одно число 0.

Примеры

oneway.in	oneway.out
4 6 1 2 1 2 2 3 2 4 4 3 1 4	1 2 2 1 2 3 4 2 3 4 1 4
2 1 1 2	0

Задача I. Определите победителя

Имя входного файла:	game.in
Имя выходного файла:	game.out
Максимальное время работы на одном тесте:	**2 секунды**
Максимальный объем используемой памяти:	**256 мегабайт**

Вася и Петя играют в следующую игру. Они берут колоду из 36 карточек. На каждой карточке написано число от 1 до 9 и каждая карточка покрашена в один из 4 цветов так, что есть ровно по 9 карточек каждого цвета и они пронумерованы числами от 1 до 9. Карты перемешиваются, и игрокам раздается по 18 карт.

Дальше игроки по очереди делают ходы. За один ход игрок может выложить на стол одну карточку по следующим правилам. Карточку с цифрой 5 можно выкладывать на стол в любой момент. Карточку с другой цифрой можно выкладывать только если на стол уже выложена карточка того же цвета, на которой написано число на 1 большее или на 1 меньшее, чем на данной карточке (не важно, была ли эта карточка выложена вами или вашим противником, и была ли она выложена на предыдущем ходе или раньше). Если игрок может выложить хоть какую-то карточку, он обязан делать ход. Если ни одну карточку игрок выложить не может, он пропускает ход.

Выигрывает тот, кто первым выложит все свои карточки на стол.

Напишите программу, которая по информации о том, кому какие карточки достались, определяет, кто выиграет при оптимальной игре обоих игроков.

Формат входных данных

Во входном файле записаны 18 пар чисел, описывающих карточки, которые достались первому игроку. Каждая карточка описывается двумя числами — номером цвета (от 1 до 4) и цифрой, которая написана на карточке (от 1 до 9). Второму игроку, соответственно, достались все остальные карточки.

Формат выходных данных

В выходной файл выведите одно число (1 или 2) — номер игрока, который выиграет при оптимальной игре обоих игроков.

Пример

game.in	game.out
1 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 2 1 2 2 2 3 2 4 2 5 2 6 2 7 2 8 2 9	1

Задача J. Соберите числа

Имя входного файла:	bricks.in
Имя выходного файла:	bricks.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 мегабайта

У Васи в распоряжении оказался набор кубиков. Вася решил на каждой грани каждого кубика написать по цифре и дальше использовать кубики для того, чтобы складывать из них числа. Вася хочет написать цифры так, чтобы уметь складывать любое число от 1 до некоторого числа K. Посчитайте такое максимальное K, до которого Вася сможет выкладывать все числа, если в распоряжении у Васи оказалось N кубиков. Заметьте, что если на какой-то грани какого-то кубика написана цифра 6, то эту же грань можно использовать и как цифру 9, просто перевернув соответствующий кубик.

При выкладывании числа Вася не обязан использовать все кубики. Ведущие нули в числах не нужны.

Рассмотрим примеры.

Пусть N=1. Тогда, написав на гранях кубика цифры от 1 до 6, Вася сможет выкладывать числа от 1 до 6. Тем самым, K=6.

Пусть N=2. Тогда, написав на гранях одного кубика цифры от 1 до 6, а на гранях другого цифры 0, 1, 2, 3, 7, 8, Вася сможет выложить любое число от 1 до 43.

Формат входных данных

Во входном файле записано одно число N ($1 \le N \le 1000000$).

Формат выходных данных

В выходной файл выведите максимальное значение K такое, что имея N кубиков Вася может так написать на их гранях цифры, чтобы было возможно выложить любое число от 1 до K.

Примеры

bricks.in	bricks.out
1	6
2	43