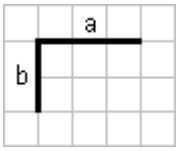
Задача A. Уголки

Имя входного файла:	`corners.in`	Ограничение по времени:	0.6 секунд
Имя выходного файла:	`corners.out`	Ограничение по памяти:	64 Мб



Пара чисел (*a, b*) называется «уголком» и представляет собой два отрезка соединённых под прямым углом: *a* – длина горизонтального отрезка, *b* – длина вертикального отрезка; верхний конец отрезка *b* стыкуется с левым концом отрезка *a*. Если *a=0* или *b=0,* то «уголок» является просто отрезком.

Проверьте, можно ли из заданного набора «уголков» составить прямоугольник. «Уголки» можно поворачивать, но нельзя отражать. Надо использовать все «уголки».

Входные данные

В первой строке входного файла записано число *N* *(1<=N<=10)* – количество «уголков». Далее следуют *N* пар целых чисел (a_i, b_i) – описания «уголков» $(0<=a_i, b_i<=100)$.

Выходные данные

Выходной файл должен содержать два числа *W* и *H* – ширину и высоту прямоугольника, при этом *W>=H*. Если решений не существует, следует вывести «*–1*»; если решений несколько, следует вывести решение с максимальным значением *W*.

Примеры

corners.in	corners.out
10 1 1 1 1 1 0 1 0 1 0 1 0 0 2 0 2 0 2 0 2	7 1
5 1 1 1 1 1 1 1 1 5 0	-1

Задача B. Шестизначный код

Имя входного файла:	kod.in	Ограничение по времени:	0.1 секунд
Имя выходного файла:	kod.out	Ограничение по памяти:	64 Мб

Представьте, что вы забыли шестизначный код от сейфа, в котором находится «ключ от квартиры, где деньги лежат», но в памяти остался целый ряд соотношений между цифрами. Возможны шесть видов соотношений: >, <, =, <> (не равно), <=, >=. Например, «5<>3» означает, что пятая и третья цифры забытого числа различны.

Посчитайте, какое количество вариантов надо перебрать, чтобы достать заветный ключ.

Входные данные

В первой строке входного файла записано число N ($1 \le N \le 30$). Далее следует N строк, каждая из которых содержит одно соотношение между цифрами кода (внутри строки пробелов нет!).

Выходные данные

В выходной файл выведите единственное число – количество подходящих вариантов кода.

Пример

kod.in	kod.out
3 1<2 3=1 3>4	12000

Задача C. Мелкие купюры

Имя входного файла:	`money.in`	Ограничение по времени:	0.5 секунд
Имя выходного файла:	`money.out`	Ограничение по памяти:	64 Мб

Представьте, что у вас есть N различных прямоугольных купюр (разные страны, разные номиналы). Вы не знаете реальную стоимость этих денег, но решили отделить «мелкие» купюры. Купюра называется «мелкой», если её можно полностью накрыть какой-нибудь имеющейся купюрой. При этом купюры можно поворачивать, но соответствующие стороны должны быть параллельны.

Входные данные

В первой строке входного файла записано натуральное число N (2<=N<=200 000). Каждая из следующих N строк описывает одну купюру, т.е. содержит два натуральных числа – линейные размеры купюры. Все размеры не превосходят 1 000 000.

Выходные данные

В выходной файл выведите единственное число – количество мелких купюр.

Пример

money.in	money.out
3 1 10 9 9 10 3	1

Задача D. Параллелограммы

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

На плоскости заданы N точек. Никакие три из них не лежат на одной прямой. Требуется определить количество различных параллелограммов, все четыре вершины которых лежат в данных точках.

Входные данные

В первой строке входного файла находится натуральное число N, 4<=N<=1414. В каждой из следующих N строк содержатся два целых числа, разделенных пробелом: координаты одной точки. Все координаты не превосходят 1000000 по абсолютной величине. Гарантируется, что никакие три из данных точек не лежат на одной прямой.

Выходные данные

В единственной строке выходного файла следует вывести одно целое неотрицательное число – количество различных параллелограммов, все четыре вершины которых лежат в данных точках.

Примеры

parall.in	parall.out
4 0 0 0 1 1 0 1 1	1
6 0 0 3 3 1 2 2 1 3 8 0 5	2

Задача E. Ещё один код

Имя входного файла:	`kod2.in`	Ограничение по времени:	0.5 секунда
Имя выходного файла:	`kod2.out`	Ограничение по памяти:	64 Мб

Представьте, что вы опять забыли шестизначный код от сейфа, в котором находится «ключ от квартиры, где деньги лежат», но не полностью – какие-то цифры вы помните, а какие-то нет. Однако вы точно знаете, что этот код делится без остатка на некоторое натурально число N.

Попробуйте восстановить утраченный код. Если это можно сделать несколькими способами, то найдите минимальный код из возможных.

Входные данные

В первой строке входного файла находится натуральное число N, 2<=N<=10000. Во второй строке содержится запись забытого кода – строка, состоящая из цифр и знаков "?", которые обозначают забытые цифры. Никаких других символов, кроме цифр от 0 до 9 и вопросительных знаков во второй строке нет, а первый символ в ней не равен 0. Количество символов во второй строке не превосходит 250, причём среди них не более 25 знаков "?".
Первая цифра восстановленного числа должна быть не равна 0. Гарантируется, что задача имеет решение, т.е. найдётся хотя бы одно число, удовлетворяющее условиям.

Выходные данные

В единственной строке выходного файла следует вывести одно целое неотрицательное число – забытый код.

Пример

kod2.in	kod2.out
25 ?7?	175

Задача F. Стек

Имя входного файла:	stack.in	Ограничение по времени:	0.1 секунда
Имя выходного файла:	stack.out	Ограничение по памяти:	64 Мб

Есть N карточек, на каждой из которых написано одно целое положительное число – цена карточки. Карточки сложены стопкой – одна на другой.
Игрок располагает суммой K рублей. За один ход игрок может взять верхнюю карточку в стопке и

- забрать её себе, заплатив столько рублей, сколько указано на карточке (если у него есть нужная сумма), либо
- отбросить эту карточку, заплатив за это 1 рубль.

После этого игрок может прекратить игру, а может сделать следующий ход и т.д.
Цель игрока – собрать у себя как можно больше карточек.

Напишите программу, которая определяет число K – наибольшее количество карточек, которые может собрать игрок, и число Q – наименьшее количество рублей, которые ему необходимы, чтобы собрать K карточек (разумеется, Q не превосходит S).

Входные данные

В первой строке входного файла записаны числа N (1<=N<=5000) и K (1<=K<=50000), разделенные пробелом. Затем идут числа на карточках – по одному числу в строке – в том порядке, в каком они лежат в стопке (сверху вниз). Числа на карточках не превосходят 1000.

Выходные данные

Выходной файл должен состоять из двух строк: в первой строке должно стоять число K, во второй – число Q.

Пример

stack.in	stack.out	Примечание
5 8 4 3 2 7 3	2 6	Игрок отбрасывает первую карточку, забирает вторую и третью и заканчивает игру.
7 11 10 3 2 1 4 2 4	4 10	Игрок отбрасывает первую и пятую карточки, забирает вторую, третью, четвёртую и шестую.

Задача G. Субпалиндром

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

Субстрокой данной строки назовём строку, которая может быть получена из данной строки вычёркиванием некоторых (возможно, что и ни одного) символов.

Палиндромом называется непустая строка, которая одинаково читается в обоих направлениях: слева направо и справа налево. Примеры палиндромов: PALILAP, AAAA, QQ, D; примеры непалиндромов: PALIN, ABCD, QD.

Субпалиндромом данной строки назовём любую её непустую субстроку, которая является палиндромом.

Например, в строке **PASCAL** имеется 9 субпалиндромов: 6 однобуквенных субпалиндромов (**P,A,S,C,A,L**), 1 двухбуквенный (**AA**) и 2 трёхбуквенных (**ASA, ACA**).

Дана некоторая строка. Требуется подсчитать количество ее субпалиндромов. Точнее, требуется найти остаток от деления этой величины на 29876543.

Входные данные

В единственной строке входного файла задана некоторая строка символов. Она содержит не более 5000 символов, причём все символы являются большими буквами латинского алфавита (от ‘A’ до ’Z’).

Выходные данные

В единственной строке выходного файла следует выводить одно целое неотрицательное число – остаток от деления количества субпалиндромов данной строки на 29876543**.**

Пример

palindr.in	palindr.out
ABCBA	13

Задача H. Система счисления

Имя входного файла:	`nobinary.in`	Ограничение по времени:	0.1 секунд
Имя выходного файла:	`nobinary.out`	Ограничение по памяти:	64 Мб

В двоичной системе счисления запись $a_n a_{n-1} a_{n-2} \ldots a_2 a_1 a_0$ означает число, равное $a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$, причём каждая из величин a_k, k=0,1,…,n, равна 0 или 1 и называется двоичной цифрой. Впрочем, это и так всем известно.

Рассмотрим другую систему счисления. Назовём её недвоичной. В ней, в точности, как и в двоичной системе, запись $a_n a_{n-1} a_{n-2} \ldots a_2 a_1 a_0$ означает число, равное $a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$, только каждая из цифр a_k, k=0,1,…,n, может быть любым числом от 0 до 9. Например, недвоичная запись 25 в обычной десятичной системе записывается как 9 ($2 \cdot 2^1 + 5 \cdot 2^0 = 9$), а недвоичная запись 321 в десятичной системе записывается как 17 ($3 \cdot 2^2 + 2 \cdot 2^1 + 1 \cdot 2^0 = 17$). Легко видеть, что одно и то же число в недвоичной системе может быть записано разными способами. Например, числу 11 соответствуют недвоичные записи 19, 27, 35, 43, 51, 107, 115, 123, 131, 203, 211, 1003 и 1011 – всего 13 различных способов.

Задача состоит в том, чтобы для заданного числа N определить количество различных способов записать его в недвоичной системе. Поскольку результат может быть очень большим, то выводить следует остаток от деления результата на 29876543.

Входные данные

В единственной строке входного файла находится число N, $1 <= N < 2^{63}$ (конечно, записанное в десятичной системе; ответ тоже надо выводить в десятичной записи).

Выходные данные

В выходной файл надо вывести одно число – остаток от деления количества представлений числа N в недвоичной системе на 29876543.

Пример

nobinary.in	nobinary.out
11	13

Задача I. Больше-меньше числа

Имя входного файла:	`lmnum.in`	Ограничение по времени:	0.1 секунды
Имя выходного файла:	`lmnum.out`	Ограничение по памяти:	64 Мб

Пусть $d_1d_2d_3\ldots d_{n-1}d_n$ – десятичная запись целого положительного числа X ($d_1 > 0$).
Число X называется «больше-меньше числом», если его цифры удовлетворяют соотношениям

$$d_1 < d_2 > d_3 < d_4 > d_5 \ldots$$

Число X называется «меньше-больше числом», если его цифры удовлетворяют соотношениям

$$d_1 > d_2 < d_3 > d_4 < d_5 \ldots$$

Все однозначные числа (и только они) являются одновременно и «больше-меньше числами», и «меньше-больше числами».
Выпишем все «больше-меньше» и «меньше-больше» числа в порядке возрастания и пронумеруем их, начиная с 1.
Задание состоит из нескольких частей. Требуется найти

- число, следующее за данной числом X;
- номер заданного числа X;
- число, имеющее заданный порядковый номер N;
- количество «больше-меньше чисел» среди первых N выписанных чисел.

.

Входные данные

Входной текстовый файл состоит из двух строк. В первой строке находится целое положительное число X ($1<=X<10^{17}$), причём это число является «больше-меньше числом», или «меньше-больше числом»,. Во второй строке находится целое положительное число N ($N<10^{17}$).

Выходные данные

Выходной файл состоит из четырёх строк, в каждой из которых записан ответ на одно из заданий в том порядке, в каком они идут в условии.

Пример

lmnum.in	lmnum.out
9 5	10 9 5 5

Задача J. Язык орангутанга

Имя входного файла:	`lang.in`	Ограничение по времени:	10 секунд
Имя выходного файла:	`lang.out`	Ограничение по памяти:	64 Мб

Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook. Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook. Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook. Ook! Ook.

Программа на языке орангутанга, печатающая “Hello, world!”

Большинство школьников Харькова программируют на Паскале, замечательном языке программирования для обучения. Однако, в последнее время в результате тлетворного влияния моды всё больше учеников начинают программировать на языке орангутанга. Этот факт особенно беспокоит жюри V чемпионата Харькова, так как их тестирующая система пока не поддерживает этот новый язык.

В результате социологического опроса участников выяснилось, что у каждого школьника есть ближайшие друзья-программисты, с мнением которых он считается. Если больше половины ближайших друзей школьника программируют на языке орангутанга, то школьник поддаётся их дурному влиянию и, в свою очередь, влияет на оставшихся друзей. К сожалению, никакое количество друзей не способно переубедить школьника перейти обратно с языка орангутанга к старому доброму паскалю.

На собрании жюри было решено, что распространение языка орангутанга можно было бы сдержать, если бы больше школьников дружили между собой.

Требуется подружить двух участников чемпионата таким образом, чтобы максимизировать количество школьников, которые останутся верны паскалю.

Входные данные

Входной файл содержит число участников чемпионата N ($2 \le N \le 500$), за которым идёт N чисел c_i, где $c_i = 0$, если i-й участник программирует на паскале, и $c_i = 1$, если он пишет на языке орангутанга. Далее следует число пар друзей M ($0 \le M \le 1000$), за которым идут M пар чисел a_i b_i ($1 \le a_i$, $b_i \le N$), означающих, что школьники a_i и b_i — друзья.

Выходные данные

В выходной файл нужно вывести два числа — номера участников, которых следует подружить. Если решений несколько, вывести любое из них. Если никакая пара новых друзей не улучшит результат, следует вывести число `-1`.

Примеры

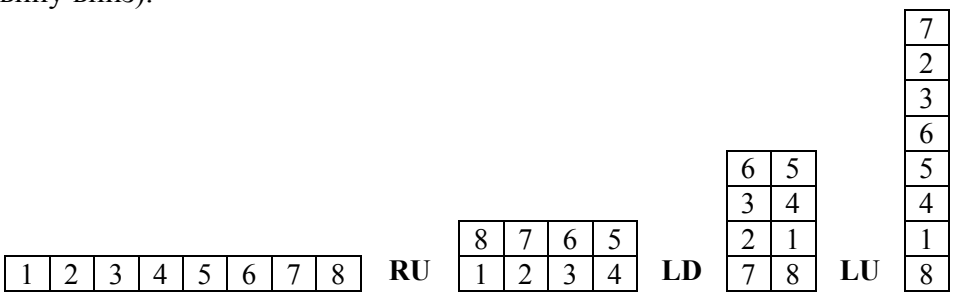
lang.in	lang.out
3 1 0 0 2 1 2 1 3	2 3
9 1 0 0 0 0 1 1 1 0 7 1 2 2 3 3 4 4 5 6 2 7 3 8 4	9 2

Задача K. Кубики

Имя входного файла:	21.in	Ограничение по времени:	0.3 секунд
Имя выходного файла:	21.out	Ограничение по памяти:	64 Мб

Пусть K – некоторое натуральное число. Имеется $N=2^K$ кубиков, расставленных в ряд. Кубики пронумерованы слева направо числами от 1 до N.

Операция «переворачивания» кубиков заключается в следующем: разделяем ряд на две половины и, не трогая одну из половинок, переворачиваем вторую половинку на первую или под первую. Таким образом, операцию переворачивания мы будем задавать одним из перечисленных способов: **LU** (переворачиваем левую половину вверх), **LD** (переворачиваем левую половину вниз), **RU** (переворачиваем правую половину вверх) или **RD** (переворачиваем правую половину вниз).



Операцию перекладывания кубиков продолжаем до тех пор, пока все кубики не будут выставлены в столбик.

В этой задаче величина K будет константой равной 21. Требуется определить порядок следования чисел на столбике из кубиков. Точнее говоря, нас интересуют только 1024 верхних числа – их и надо выводить.

Входные данные

Входной текстовый файл содержит описание сгибов и состоит из 21 строки – каждая строка содержит два символа: LU, LD, RU или RD. M-я строка содержит описание M-го сгибания (1<=M<=21).

Выходные данные

Выходной файл должен состоять ровно из 1024 строк. M-я строка (1<=M<=1024) содержит M-ое сверху число.

если бы было K=3, а выводить надо было бы верхние 5 чисел, то файлы выглядели бы так:

Пример

21.in	21.out
LD RD LU	8 1 4 5 6