Задача 1. Биномиальные коэффициенты

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Биномиальный коэффициент C_n^k — это количество битовых массивов длины n, в которых ровно k битов единичные.

Подробнее о биномиальных коэффициентах можно прочитать, например, в википедии: https://ru.wikipedia.org/wiki/

Нужно вычислить все биномиальные коэффициенты для $n \leqslant 1\,000$ при помощи треугольника Паскаля. Подробнее о нём можно прочитать, опять же, в википедии.

Треугольник Паскаля позволяет вычислять все биномиальные коэффициенты в порядке увеличения n, т.к.:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k \quad \text{при } 0 < k < n$$

После того, как все коэффициенты вычислены, нужно прочитать набор пар n и k из входного файла и выдать для каждой пары соответвующий коэффициент C_n^k.

Формат входных данных

В первой строке содержится целое число Q — количество запросов в файле ($1 \leqslant N \leqslant 10\,000$). В каждой из следующих Q строк содержится по два целых числа n и k, для которых нужно распечатать коэффициент ($0 \leqslant k \leqslant n \leqslant 1\,000$).

Формат выходных данных

Нужно вывести Q вещественных чисел, по одному в строке — биномиальные коэффициенты для запросов из входном файле.

Внимание: хоть биномиальные коэффициенты и целые, они могут быть очень большими. Поэтому вычисляйте их как вещественные числа с использованием типа `double`, и распечатывайте при помощи формата `"%0.10g"` !

Пример

input.txt	output.txt
8 4 0 4 1 4 2 4 3 4 4 10 5 100 50 1000 500	1 4 6 4 1 252 1.008913445e+29 2.702882409e+299

Пояснение к примеру

Первые пять запросов распечатывают коэффициенты для $n = 4$. Последний запрос распечатывает самый большой коэффициент, который может быть запрошен в данной задаче.

Задача 2. Японский кроссворд

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Когда Кролику нечего делать, Кролик не погружается в рефлексию, как Пятачок, и не занимается научными исследованиями, как Винни-Пух. Он вместо этого разгадывает всякие головоломки. И сегодня на повестке дня у него японский кроссворд.

Японский кроссворд представляет собой поле из $N \times M$ клеток. При этом шифр таков: для каждой из N строк выписывается последовательность чисел, задающих размеры групп подряд идущих чёрных клеток, перечисляемых слева направо. Соседние группы чёрных клеток должны быть разделены хотя бы одной белой клеткой. Аналогично составляется шифр по вертикали для каждого из M столбцов, причём нумерация групп чёрных клеток идёт сверху вниз. Кролик любит разгадывать такие кроссворды сам, а от вас, наоборот, требуется составлять шифры по заданным полям.

Формат входных данных

В первой строке входного файла содержатся два целых числа —N и M ($1 \leqslant N, M \leqslant 10$).

В каждой из следующих N строк содержится по M символов, разделённых пробелами. Строки могут содержать только символы **0** и **1**. **0** обозначает белую клетку, а **1** – чёрную.

Формат выходных данных

Выходной файл должен содержать $N + M$ строк.

В i-й строке выходного файла должен быть шифр i-й строки кроссворда: последовательность чисел, соответствующих размерам групп чёрных клеток в этой строке, разделённых пробелами ($1 \leqslant i \leqslant N$).

Далее в j-й строке выходного файла должен быть аналогичным образом записан шифр $(j - N)$-го столбца кроссворда ($N + 1 \leqslant j \leqslant N + M$).

Если в какой-то строке или в каком-то столбце нет черных клеток, то нужно выдавать для них соответствующую пустую строку.

Примеры

input.txt	output.txt
2 2 0 0 1 1	 2 1 1
3 3 1 0 1 0 1 0 1 0 1	1 1 1 1 1 1 1 1 1 1
4 4 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1	2 2 1 2 4 3 1 1 1 2 1 2

Задача 3. Хранение строк

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	**специальное**

Нужно реализовать систему хранения строк в динамической памяти.

Система должна обрабатывать три вида запросов/операций:

0. *Создать строку*: указывается длина создаваемой строки и собственно содержимое строки. Нужно выделить блок памяти при помощи `malloc` и записать в него содержимое.
1. *Удалить строку*: указывается идентификатор ранее созданной строки. Нужно освободить память, выделенную ранее для этой строки, при помощи `free`.
2. *Вывести строку*: указывается идентификатор ранее созданной строки. Нужно распечатать содержимое этой строки в выходной файл.
3. *Сколько символов в строке*: указывается идентификатор ранее созданной строки и один символ. Нужно вывести в выходной файл одно целое число: сколько раз этот символ встречается в указанной строке.

Идентификатором строки является номер запроса на её создание в общей нумерации запросов. При обращении к строке по идентификатору гарантируется, что эта строка ещё **не** была удалена.

Замечание: Не все созданные строки удаляются в запросах. В целях аккуратности удалите все остающиеся строки при помощи `free` в конце программы.

Формат входных данных

В первой строке записано целое число N — количество количество запросов ($1 \leqslant N \leqslant 10^5$). В остальных N строках описаны запросы, по одному запросу в строке.

Запрос начинается с целого числа t, обозначающего тип запроса. Если $t = 0$, то это запрос создания, и тогда далее указана длина строки l и сама строка ($1 \leqslant l \leqslant 10^5$). Если $t = 1$, то это запрос удаления, а если $t = 2$ — то это запрос на вывод. В обоих случаях далее указано целое число k — идентификатор строки ($0 \leqslant k < N$). Если $t = 3$, то это запрос о количестве символов, и тогда далее указан идентификатор строки k и ещё один символ через пробел.

Все строки состоят из произвольных печатаемых символов ASCII (коды от 33 до 126 включительно). То же верно и про символы в запросах на количество.

Сумма всех длин l по всем запросам создания не превышает $5 \cdot 10^5$. Суммарный размер всех строк, которые вам нужно вывести, не превышает $5 \cdot 10^5$. Сумма длин строк по всем запросам о количестве символов не превышает 10^8.

Пример

input.txt	output.txt
12 0 3 aba 2 0 3 0 a 3 0 b 1 0 0 6 malloc 0 5 %d%s% 2 1 2 2 1 1 3 2 % 2 2	aba 2 1 malloc %d%s% 3 %d%s%

Пояснение к примеру

Сначала создаётся строка “aba” с идентификатором 0. Далее она распечатывается, и выводится количество букв 'a' и 'b' в ней. Наконец, нулевая строка удаляется.

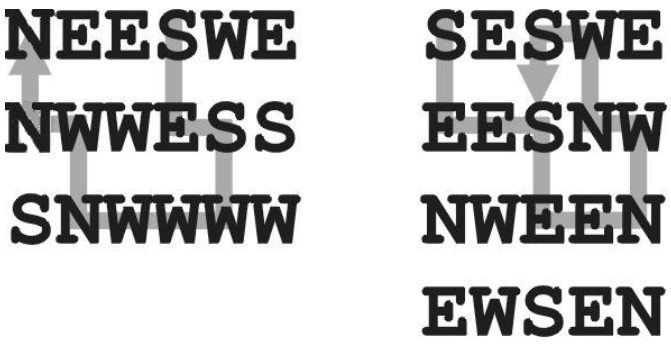
Затем создаются ещё две строки: строка “malloc” и строка “%d%s%” с идентификаторами 1 и 2 соответственно. Потом они распечатываются и строка “malloc” удаляется. В конце выводится количество процентов в строке “%d%s%” и она распечатывается ещё раз.

Задача 4. Движение робота

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Робот был запрограммирован, чтобы следовать инструкциям в пути. Инструкции указывают направление на сетке, куда робот должен переместиться и остановиться. Возможны следующие инструкции:

- `N` – север (вверх по странице),
- `S` – юг (вниз по странице),
- `E` – восток (вправо по странице),
- `W` – запад (влево по странице).



Например, предположим, что робот начинает движение на севере (вверху) сетки (первый пример) и начинает двигаться на юг (вниз). Путь, пройденный роботом, показан на левом рисунке. Робот выполняет 10 инструкций на сетке, прежде чем покинуть ее.

Сравните с тем, что случается на другой сетке (правый рисунок): робот проходит 3 инструкции только один раз, и затем начинается цикл из 8 инструкций, из которого нет выхода.

Ваша задача состоит в том, чтобы написать программу, определяющую, сколько шагов потребуется роботу, чтобы покинуть сетку, или сколько он сделает шагов до цикла, и сколько шагов составляют цикл.

Формат входных данных

В первой строке входного файла содержатся три целых числа, разделенных пробелами: число строк в сетке, число столбцов в сетке, и номер столбца, с которого робот стартует с севера. Возможные стартовые столбцы нумеруются, начиная с единицы слева направо.

Затем идут строки направляющих инструкций. Каждая сетка имеет хотя бы 1 и не более 10 строк и столбцов инструкций. Строки инструкций состоят только из символов `N`, `S`, `E` и `W` без пробелов

Формат выходных данных

Ваша программа должна выводить одну строку, если робот, следуя некоторому числу инструкций, покидает сетку на любой из четырех сторон, или если робот следует инструкциям на некотором количестве клеток, а затем следует инструкциям на некотором количестве клеток циклически.

Примеры

input.txt
3 6 4 NEESWE NWWESS SNWWWW
output.txt
10 step(s) to exit

input.txt
4 5 1 SESWE EESNW NWEEN EWSEN
output.txt
3 step(s) before a loop of 8 step(s)

Пояснение к примеру

Пример входного файла отвечает двум изображенным сеткам и иллюстрирует два вида выходных файлов. За словом `step` всегда следует `(s)` даже если перед ним стоит 1.

Задача 5. Топики

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Одним из простых методов определения темы статьи в газете является выявление наиболее часто встречающихся слов в ней. Естественно, маленькие служебные слова, такие как **the**, **a** или **of** нужно игнорировать, т.к. они не влияют на содержание статьи.

Ваша задача – найти в предложенной статье наиболее часто встречающееся слово, которое не входит в список слов, которые нужно игнорировать.

Формат входных данных

В первой строке входного файла задан список слов, которые нужно игнорировать. Между словами стоит ровно один пробел. Их количество не превышает 20.

Остальные строки входного файла содержат текст, в котором нужно найти наиболее встречающееся слово. Слова не разрываются концом строки. Каждая строка начинается с нового слова. Статья заканчивается двумя звездочками (**). Регистр букв в словах нужно игнорировать. Однокоренные слова с разными окончаниями считаются разными словами. Знаки препинания тоже нужно игнорировать.

Формат выходных данных

В выходной файл необходимо вывести наиболее часто встречающееся в заданной статье слово. Это слово нужно вывести в верхнем регистре.

Пример

input.txt

`the at in on to of is that`
`The state finals of the Texas Computer Education`
`Association Computer Programming Contest is today.`
`Teams from all over the state of Texas are`
`participating in the event. In last year's contest,`
`each team brought their own computer.**`

output.txt
`COMPUTER`

Задача 6. Шифр Шерлока Холмса

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Как известно, Шерлок Холмс использовал в своей деятельности различные способы шифрования. Опишем один из них.

Имеется $2 \cdot N$ строк, каждая из которых имеет $2 \cdot N$ символов. Также имеется квадрат, размером $2 \cdot N \times 2 \cdot N$ клеток, размер каждой клетки совпадает с размером текстового символа. В этом квадрате вырезано K клеток ($0 \leqslant K \leqslant N^2$). Квадрат является ключом к зашифрованному сообщению. Если мы наложим ключ на текст, то в вырезанных клетках получаем первую часть расшифрованного сообщения. Теперь поворачиваем ключ на 90 градусов по направлению часовой стрелки и снова накладываем на текст, получаем следующую часть сообщения, потом опять поворачиваем на 90 градусов, пока не получаем исходное положение. Таким образом, мы четыре раза накладываем квадрат на текст. В каждой позиции ключа буквы, составляющие текст, читаются по строкам сверху вниз, начиная с первой строки и заканчивая последней, в каждой строке они выбираются в порядке слева направо. Каждая расшифрованная часть текста приписывается в конец результирующей строки, пустой в начале.

Ключ будем называть **правильным**, если при повороте ключа мы открываем только новые клетки таблицы, то есть только те, которые в предыдущих положениях ключа были закрыты.

Формат входных данных

В первой строке входного файла записано целое число N ($1 \leqslant N \leqslant 125$).

Далее находится описание ключа: в $2 \cdot N$ строках записано по $2 \cdot N$ цифр (0 или 1), разделенных пробелами (если 1 – клетка вырезана).

Затем идет описание шифра: $2 \cdot N$ текстовых строк, имеющих по $2 \cdot N$ символов – букв латинского алфавита.

Формат выходных данных

В выходной файл необходимо вывести сообщение `No`, если во входных данных задан неправильный ключ. В противном случае необходимо вывести расшифрованный текст.

Примеры

input.txt	output.txt
1 1 0 0 0 zi am	zima
2 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 abcd abcd fbcd abcd	No

Задача 7. Поиск мест

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Группа из K друзей пошла в кино. Они пришли довольно поздно, чтобы выбирать хорошие места. Поэтому приходится искать места поближе друг к другу. Так как они все продвинутые студенты, они решили подойти к проблеме с научной точки зрения, а именно, рассмотреть задачу оптимизации.

В кинотеатре R рядов по C мест в каждом. Студентам доступна информация о занятых и свободных местах. Они хотят сесть как можно ближе друг к другу, не зависимо, в какой части зала такие места найдутся.

Чтобы установить формальный критерий, они решили минимизировать «распространение» своей группы по залу. Распространение определяется как площадь минимального прямоугольника со сторонами, параллельными местам, который содержит все купленные места. Будем считать площадью такого прямоугольника количество всех мест, которые он содержит.

Помогите им написать программу, которая решит их проблему.

Формат входных данных

Входной файл содержит несколько наборов тестов. Каждый набор состоит из нескольких строк. Первая строка набора содержит три целых положительных числа R, C и K ($1 \leqslant R, C \leqslant 300, 1 \leqslant K \leqslant R \times C$).

Каждая из следующих R строк содержит по C символов. j-й символ i-ой строки может быть либо ‘X’, если j-е место в i-м ряду занято, либо ‘.’, если оно свободно.

Гарантируется, что всегда имеется не менее K свободных мест. Входной файл заканчивается строкой, содержащей три нуля, записанных через пробел.

Формат выходных данных

Для каждого набора выходной файл должен содержать строку, в которой записано целое число – минимальное распространение группы.

Пример

input.txt	output.txt
3 5 5 ...XX .X.XX XX... 5 6 6 ..X.X. .XXX.. .XX.X. .XXX.X .XX.XX 0 0 0	6 9