Олимпиада «Спортивное программирование на Урале»

2019-2020 учебный год

Задания первого (отборочного) этапа

Задача A. A+B

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам заданы два целых числа A и B. Выведите $A + B$.

Формат входных данных

В первой строке дано единственное целое число A ($-100 \leqslant A \leqslant 100$). Во второй строке дано единственное целое число B ($-100 \leqslant B \leqslant 100$).

Формат выходных данных

В единственной строке выведите число $A + B$.

Пример

стандартный ввод	стандартный вывод
7 8	15

Задача B. Шахматы с параметром

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

Заветная мечта студента Антона состоит в том, чтобы обыграть его одногруппника Даниила в параметрические шахматы. Правила этой интеллектуальной игры весьма сложны, к тому же могут меняться в зависимости от желания игроков и времени суток, однако (k, l)-конь неизменно остается одной из важнейших ее фигур, а потому Антон решил потратить несколько вечеров на то, чтобы хорошенько изучить тактику игры им. В параметрические шахматы играют на прямоугольной доске размером $n \cdot m$ клеток. (k, l)-конь ходит по следующим правилам: за один ход он перемещается в клетку, отстоящую от исходной на k клеток в одном направлении (горизонтальном или вертикальном) и на l клеток в направлении, перпендикулярном ему.

Например, обычный шахматный конь — это $(1, 2)$-конь, или, что то же самое, $(2, 1)$-конь. Стоя на клетке с координатами $(4, 4)$ такой конь имеет право ходить на клетки $(5, 6)$, $(3, 6)$, $(6, 5)$, $(2, 5)$, $(6, 3)$, $(2, 3)$, $(5, 2)$ и $(3, 2)$ при условии, что размеры доски позволяют это сделать.

Антон поставил на доску одного белого (k, l)-коня и несколько белых пешек. Теперь его интересует, можно ли, сделав несколько ходов (k, l)-конем, перейти в клетку с координатами (x, y), и если да, то за какое наименьшее количество ходов можно это сделать. Естественно, ходить на клетки, на которых уже стоят пешки, нельзя.

Формат входных данных

Первая строка содержит целые числа n и m — размеры поля, $1 \leqslant n, m \leqslant 200$.

Вторая строчка содержит целые числа k и l — параметры коня, $0 \leqslant k, l \leqslant 200$.

Третья строка содержит целые числа x_0, y_0, x, y, где x_0, y_0 — номера столбца и строки, в которых (k, l)-конь стоит изначально, x и y — соответственно, номера столбца и строки клетки, в которую (k, l)-конь должен прийти. Столбцы нумеруются слева направо от 0 до $n - 1$,строки — сверху вниз от 0 до $m - 1$. Гарантируется, что клетки с координатами (x_0, y_0) и (x, y) находятся на доске.

Далее идет m строк,в каждой из которых содержится n чисел, каждое из которых равно нулю или единице. i-я строка из этих m описывает $(i-1)$-ю горизонталь поля, j-й ее элемент равен единице тогда и только тогда, когда в клетке с координатами $(j - 1, i - 1)$ стоит пешка.

Гарантируется, что на клетках (x_0, y_0) и (x, y) пешки не стоят.

Формат выходных данных

В случае, если правила параметрических шахмат не позволяют привести (k, l)-коня из клетки (x_0, y_0) в клетку (x, y), выведите -1. В противном случае выведите минимальное количество ходов, за которое Антон может осуществить задуманное.

Примеры

стандартный ввод	стандартный вывод
4 3 1 2 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0	3
4 4 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1	-1

Замечание

В первом примере $(2, 1)$-конь может переместиться с клетки $(0, 0)$ на $(1, 0)$ за три хода по следующему маршруту: $(0, 0) \to (1, 2) \to (3, 1) \to (1, 0)$

Во втором примере $(1, 1)$—конь добраться из клетки $(0, 0)$ в $(1, 0)$ не способен в принципе: несложно видеть, из клетки с четной суммой координат такой конь может прийти лишь в клетку, в которой сумма координат вновь будет четной. У клетки $(0, 0)$ сумма координат четная, а у $(1, 0)$— нет. Значит, искомого маршрута не существует.

Задача C. Число

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	8 мегабайт

Скоро пройдет олимпиада «Спортивное программирование на Урале», и у жюри возникли проблемы с размещением участников по аудиториям. Тайным голосованием было решено, что аудиторий будет всего 4 с номерами 00, 01, 10, 11, а процедура размещения по аудиториям будет проходить следующим образом. Каждому участнику будет выдан уникальный номер, а последние 2 цифры этого числа в двоичном представлении и будут номером аудитории участника. Так как данный механизм размещения очень сложен для жюри, помогите им реализовать его!

Формат входных данных

Дано число n ($1 \leqslant n \leqslant 10^{10000000}$).

Формат выходных данных

Выведите номер аудитории, в которой будет писать контест участник с данным номером.

Примеры

стандартный ввод	стандартный вывод
1	01
6	10

Замечание

Пояснение ко 2 примеру: $6_{10} = 110_2$. Обратите внимание на ограничение по памяти. Если вы сохраните все число в память, то получите вердикт **Превышено ограничение по памяти**.

Задача D. Конфеты

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Валерий очень любит конфеты. Сегодня перед завтраком он нашёл n конфет у себя в холодильнике. Валерий хочет есть конфеты каждый час, причём каждый раз он хочет съедать больше конфет, чем в предыдущий час. Валера заинтересовался, какое максимальное количество часов он сможет есть конфеты описанным выше способом.

Формат входных данных

В первой и единственной строке дано единственное целое число n ($1 \leqslant n \leqslant 10^4$).

Формат выходных данных

В первой и единственной строке выведите ответ на задачу — максимальное количество часов, которые сможет есть конфеты.

Примеры

стандартный ввод	стандартный вывод
1	1
3	2

Задача E. Безумное чаепитие

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Шляпник и Мартовский Заяц сидят за круглым столом и пьют чай.

Внезапно Шляпник начинает повторять странные действия: он встаёт со своего места, делает несколько шагов вокруг стола и садится на ближайший к нему стул.

Мартовский Заяц боится, что однажды Шляпник прогонит его со стула. Помогите Зайцу узнать, успеет ли он допить чай!

Шляпник шагает всегда в одном направлении.

Формат входных данных

В единственной строке через пробел написаны три целых числа: $2 \leqslant n \leqslant 10^{18}$ — количество стульев за круглым столом, $1 \leqslant m \leqslant n - 1$ — количество стульев между Шляпником и Мартовским Зайцем (в направлении движения Шляпника), $0 \leqslant k \leqslant 10^{18}$ — количество стульев, которые проходит Шляпник каждый раз.

Формат выходных данных

Выведите единственное целое число — число итераций, через которое Шляпник настигнет Мартовского Зайца, или -1, если это не произойдёт никогда.

Примеры

стандартный ввод	стандартный вывод
10 4 2	2
5 1 2	3
4 2 4	-1

Замечание

Пояснение ко второму примеру:

Встав первый раз, Шляпник обойдёт Мартовского Зайца и сядет на следующий за ним стул. Затем Шляпник остановится на стуле, который стоит рядом с его первоначальным местом. Наконец, пройдя ещё два стула, Шляпник окажется на месте Мартовского Зайца.

Задача F. Тип треугольника

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам даны три точки с целыми координатами. Определить, являются ли эти точки треугольником и если да, то является ли этот треугольник равнобедренным или равносторонним.

Формат входных данных

Даны три строки, каждая из которых содержит два целых числа $x, y (-100 \leqslant x, y \leqslant 100)$ — координаты вершины треугольника.

Формат выходных данных

Выедите 0 если это не треугольник, 1 если у всех трёх сторон совпадает длина, 2 если в этом треугольнике есть ровно две стороны с одинаковой длиной, иначе выведите 3.

Примеры

стандартный ввод	стандартный вывод
1 5 2 4 3 7	3
1 2 3 4 5 6	0

Олимпиада «Спортивное программирование на Урале»

2019-2020 учебный год

Задания второго (заключительного) этапа

Задача A. Задача для Джорджа

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Младший брат Боба, Джордж, очень любит шахматы и хочет стать гроссмейстером. Боб решил помочь ему и для этого придумал следующую задачу.

Дана бесконечная доска, клетки которой пронумерованы по спирали, начиная с одного (подробнее смотрите на рисунке). На клетке с номером 1 стоит шахматный Конь, который ходит в сторону наименьшего возможного числа, **которое он ещё не посещал**. Ходы коня подчиняются правилам шахмат (Конь может сходить на любую ближайшую к нему клетку, не расположенную с ним на одной горизонтали, вертикали и диагонали).

Джордж ещё маленький и без вашей помощи не сможет решить эту задачу.

Помогите Джорджу узнать номер клетки, на которой окажется Конь, если он передвинет фигуру ровно N раз.

Формат входных данных

В единственной строке дано единственное целое число N — количество передвижений, которые хочет совершить Джордж Конем. ($0 \leqslant N \leqslant 10^9$)

Формат выходных данных

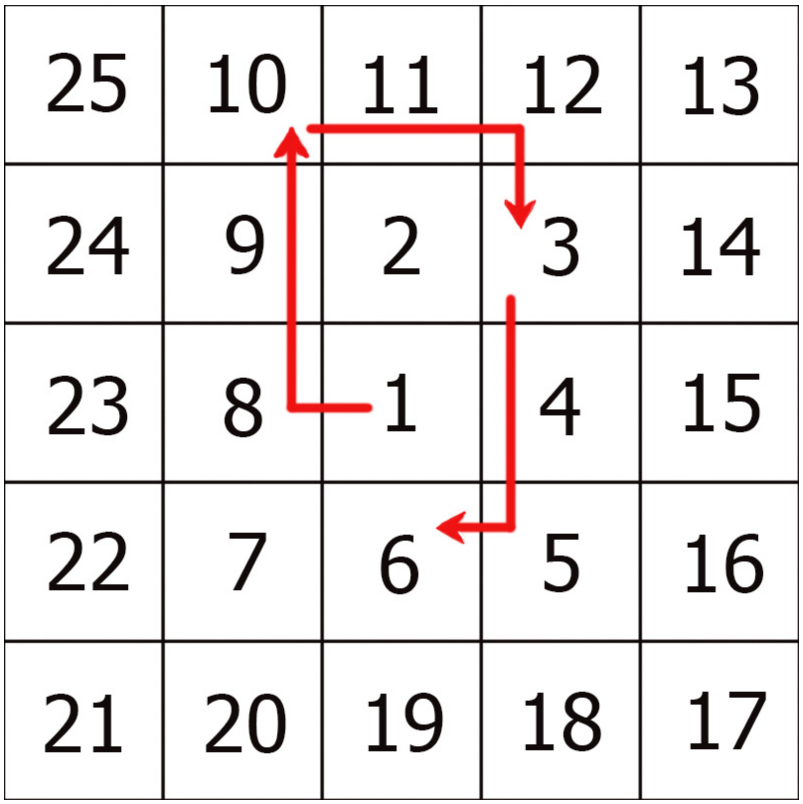
Выведите номер клетки, на которой окажется фигура после N перемещений. Если невозможно совершить все N перемещений, то выведите 0.

Примеры

стандартный ввод	стандартный вывод
1	10
3	6

Замечание

Ниже представлена иллюстрация первых трёх перемещений слона



Задача B. Ваня и игра в напёрстки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В один прохладный вечер Ваня прогуливался по своему родному городу и увидел старца, который предлагал каждому сыграть с ним в напёрстки.

Напомним, что игра в напёрстки описывается очень просто — есть 3 стаканчика, в k-й из них изначально положили шарик. Далее старец меняет стаканчики местами 5 раз, после чего спрашивает, где находится шарик.

Иван согласился сыграть в такую игру, ведь что может быть проще: запоминаешь местоположение шарика и следишь. Как выяснилось, не так уж и легко, ведь он проиграл.

Но нелёгкая занесла вас к Ване, который просит о помощи — уж очень он хочет выиграть в эту игру у старца.

Поможете ему?

Формат входных данных

В первой строке дано единственное целое число k ($1 \leqslant k \leqslant 3$) — изначальное местоположение шарика.

В следующих пяти строках даны пары чисел f_i, s_i ($1 \leqslant f_i, s_i \leqslant 3, f_i \neq s_i$) — позиции стаканчиков, которые старец поменял местами.

Формат выходных данных

В одной единственной строке выведите число — номер стаканчика, в котором будет находиться шарик находится шарик после всех операций проделанных старцем.

Примеры

стандартный ввод	стандартный вывод
2 1 2 2 3 3 1 1 2 2 1	3
1 1 2 2 3 3 1 1 2 2 1	1

Задача C. МП-Хорошая строка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Одним солнечным днём, безымянный студент Екатеринозаводского университета нашёл учебник по странной для него науке — стрингологии. В первой же главе на него обрушились определения.

Палиндром — это строка, читающаяся одинаково как слева направо, так и справа налево.

Подстрока — это фрагмент строки с начиная некоторой позиции i до позиции j ($i \leqslant j$)

Хорошая строка — это строка, состоящая только из символов ‘0’ и ‘1’, которая обладает следующим свойством: для двух любых одинаковых символов в позициях l и r ($l < r$) выполняется одно из двух условий:

1. Строка $s_l\ s_{l+1}\ ...\ s_{r-1}\ s_r$ состоит либо только из нулей, либо только из единиц.
2. Строка $s_r\ s_{r+1}\ ...\ s_n\ s_1\ ...\ s_{l-1}\ s_l$ также состоит либо только из нулей, либо только из единиц.

Максимально палиндромная (или просто МП) хорошая строка — это хорошая строка, для которой истинно следующее условие: число различных палиндромов, являющихся её подстроками, максимально среди всех хороших строк, которые состоят из того же количества символов ‘0’ и символов ‘1’.

В конце главы наш герой увидел следующую задачу. Дана строка из нулей и единиц, требуется какое минимальное число пар символов нужно поменять местами так, что после этого строка станет МП-хорошей. Но ему так и не удалось справится с этой проблемой...

А вы сможете?

Формат входных данных

На входе дана непустая строка s ($1 \leqslant |s| \leqslant 10^5$), состоящая только из символов ‘0’ и ‘1’.

Формат выходных данных

Выведите единственное целое число — минимальное число раз, которое поменяет два символа местами, чтобы строка стала МП-хорошей.

Примеры

стандартный ввод	стандартный вывод
110000	0
10010101001	2
011111100100	1

Замечание

В первом тесте изначальная строка уже является МП-хорошей, больше чем 6 различных палиндромов-подстрок получить нельзя.

Во втором тесте мы можем поменять местами первый и пятый, а также седьмой и последний символы и получить строку ‘00011111000’, она также является МП-хорошей

В третьем тесте за одно действие получим МП-хорошую строку ‘111111100000’

Задача D. Метро 3030

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Свершилось! Боб и его команда прошли в финал ICPC 3030, поэтому они много готовятся, чтобы победить. Закончив очередную тренировку, они поняли, что совершенно не следили за временем, и единственный транспорт, на котором они сейчас могут добраться до общежития — это метро. Внезапно Боб, осознал, что он единственный, у кого есть деньги, и хватит их только на маршрут с наименьшим количеством пересадок. Он хотел было написать алгоритм, который сможет его найти, но оказалось, что все ноутбуки разрядились, поэтому Боб обратился за помощью к вам.

В 31 веке сделали открытие, которое позволило передвигаться поездам настолько быстро, что кажется, как будто они мгновенно перемещаются с одного места на другое, поэтому первым делом эту технологию внедрили в метро. Метро имеет N станций, на каждой из которых стоят поезда. В некоторые моменты времени со станции i поезд мгновенно перемещается на станцию j, в тот же момент времени поезд со станции j моментально перемещается на станцию i. Таким образом происходит некий транспортный обмен. Расписание содержит ровно M таких моментов, упорядоченных по времени, и для каждого из них указаны номера станций. Боб и команда находятся на станции A, а общежитие на станции B.

Помогите ребятам найти маршрут с наименьшим количеством пересадок от станции A до станции B. **За один момент времени можно проехать лишь на одном поезде.**

Формат входных данных

В первой строке даны целые числа N, M, A и B — количество станций, количество транспортных обменов, номер станции, на которой находится Боб с командой и номер станции общежития соответственно. ($1 \leqslant N \leqslant 10^5; 0 \leqslant M \leqslant 10^5; 1 \leqslant A, B \leqslant N$)

Далее следует строк, каждая из которых содержит три целых числа t_k, i_k, j_k — время, когда происходит транспортный обмен, и номера станций, между которыми он происходит. ($1 \leqslant t_k \leqslant 10^9; 1 \leqslant i_k, j_k \leqslant N$).

Список транспортных сообщений во входных данных отсортирован по неубыванию времени.

Формат выходных данных

В первой строке выведите количество станций в маршруте с наименьшим количеством пересадок. Во второй строке через пробел выведите номера станций, на которых побывают ребята.

Если не ребята не смогут попасть в общежитие, в единственной строке выведите `«You are late»` без кавычек.

Примеры

стандартный ввод	стандартный вывод
6 7 1 6 1 1 2 3 1 4 4 2 3 6 4 5 8 2 6 12 3 4 12 5 6	3 1 2 6
6 5 4 2 3 6 4 5 3 5 7 4 3 8 2 5 9 5 6	You are late

Задача E. Аи-Ай-Ак-Ат

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В этой задаче не будет никаких долгих вступлений и привычных, для всех опытных участников наших соревнований, сказок.

Дан массив $a_1, a_2, \ldots, a_N$, состоящий из N целых чисел, и целое неотрицательное число X. Необходимо посчитать количество таких четвёрок индексов $i < j < k < t$, что $(a[i] + (a[j]\ and\ a[k]))\ xor\ a[t] = x$.

Операции *and* и *xor* для двух чисел можно определить следующим образом: запишем числа в двоичной системе счисления и дополним меньшее из них ведущими нулями так, чтобы его длина была равна длине второго числа. После этого результатом операции *and* является число, каждый разряд которого в двоичной системе равен единице тогда и только тогда, когда соответствующие разряды чисел равны единице. Аналогично результатом операции *and* является число, каждый разряд которого в двоичной системе равен нулю тогда и только тогда, когда соответствующие разряды чисел равны друг другу. Так для чисел 6 и 10 их двоичным представлением будет 110 и 1010 соответственно. Дополним ведущими нулями — получим 0110 и 1010. Таким образом, результатом операции *and* будет 10 (в двоичной системе сичсления) или 2 (в десятичной). А результатом операции *xor* — 1100 (в двоичной системе сичсления) или 12 (в десятичной).

В языках программирования операция *and* обычно обозначается словом *and* или символом «&», а операция *xor* — словом *xor* или символом «^».

Формат входных данных

В первой строке даны два целых числа N и X. ($0 \leqslant N \leqslant 3000; 0 \leqslant X \leqslant 10^9$)

Во второй строке дано N целых чисел a_i. ($0 \leqslant a_i \leqslant 2000$)

Формат выходных данных

В единственной строке выведите число — ответ на задачу.

Примеры

стандартный ввод	стандартный вывод
4 7 1 2 3 4	1
8 6 5 2 9 2 6 1 9 10	3

Задача F. Остров сокровищ

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Боб — заядлый игрок в настольные игры, поэтому как только в его любимом магазине настолок появляется что-то новое, он идет это покупать. В этот раз его коллекция пополнилась игрой «Остров сокровищ: Все или Ничего», и Боб тут же начал изучать правила.

В комплекте этой игре имеется поле размером $N \times M$ и фигурка капитана пиратов. Каждая клетка поля относится к одному из двух типов: `Океан '.'` и `Суша '#'`. Любое связное множество граничащих клеток типа `Суша` образует остров. Две клетки называются граничащими, если они имеют общую сторону.

В начале игры капитан пиратов ставится на поле типа `Океан` по заданным координатам (i, j). По клеткам типа `Океан` капитан может передвигаться вправо и вниз, а по клеткам `Суша` в любые 4 стороны. Сойти с острова он может только, если справа или снизу от клетки, на которой он находится, расположен `Океан`.

Так как персонаж не простой юнга, а капитан, ему нужно обязательно найти спрятанное сокровище, а для этого необходимо посетить все острова. Помогите Бобу понять, возможно ли это сделать.

Формат входных данных

В первой стоке даны три целых числа N, M ($2 \leqslant N, M \leqslant 1000$) — размеры поля и количество островов на нем соответственно.

В следующей строке даны два числа r, c ($1 \leqslant r \leqslant N, 1 \leqslant c \leqslant M$) — номер строки и столбца, координаты капитана в начале игры. Гарантируется, что изначально капитан находится в клетке типа `Океан`.

В следующих N строках задано поле $N \times M$, состоящее из символов `'.'` и `'#'`.

Формат выходных данных

Выведите «`YES`», если капитан сможет посетить все острова, или «`NO`» в противном случае.

Примеры

стандартный ввод	стандартный вывод
5 8 3 5 ###.###. ..#..... #.#..... #.#..#.. ..####..	YES
5 8 3 5 .##.###. ..#..... #.#..... #.#..#.. ..####..	NO
5 5 1 1 .#.#. #.#.# .#.#. #.#.# .#.#.	NO

Задача G. Редактор текста

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Компания K. разрабатывает новый текстовый редактор, который будет с лёгкостью обрабатывать файлы, занимающие петабайты места. Для того чтобы реализовать эти планы, компании необходим MVP (minimum value product) перед встречей с инвесторами. К сожалению, ничего ещё не готово, поэтоме CEO компании делегировал разработку MVP вам.

Текст в редакторе представляет собой последовательность строк. Каждая строка представляет собой последовательность из нуля или более символов. Изначально текст состоит из единственной пустой строки. В каждый момент времени определена позиция курсора — место, куда будет вставлен следующий символ. Курсор всегда находится либо между символами либо в конце либо в начале строки. Позицию курсора можно изменять с помощью нажатия кнопок влева, вправо, вверх, вниз.

Нажатие кнопки влево перемещает курсор на одну позицию влево, если курсор находится в начале строки, то он перемещается в конец предыдущей строки, если курсор находится в начале первой строки, не происходит ничего.

Нажатие кнопки вправо перемещает курсор на одну позицию вправо, если курсор находится в конце строки, то он перемещается в начало следующей строки, если курсор находится в конце последней строки, не происходит ничего.

Нажатие кнопки вверх перемещает курсор на ту же позицию предыдущей строки, если предыдущая строка короче, чем текущая позиция курсора, курсор перемещается в конец предыдущей строки, если курсор находится в первой строке, он перемещается в начало первой строки.

Нажатие кнопки вниз перемещает курсор на ту же позицию следующей строки, если следующая строка короче, чем текущая позиция курсора, курсор перемещается в конец следующей строки, если курсор находится в последней строке, он перемещается в конец последней строки.

Редактор должен поддерживать следующие операции:

- ввести символ (символ — это строчные и заглавные латинские буквы);
- перевести курсор влево, вправо, вверх, вниз согласно правилам, описанным выше;
- перевод строки — переносит всё содержимое строки после курсора на вновь созданную строку, которая следует за текущей, если курсор находится в конце строки, вновь созданная строка будет пустой;
- удалить символ слева от курсора, если курсор находится в начале строки, то текущая строка объединяется с предыдущей, если курсор находится в начале первой строки, то ничего не происходит.

Ваша задача состоит в том, чтобы вывести состояние текста после выполнения всех операций.

Формат входных данных

В единственной строке дано описание операций с текстом. Операции ввода символов заданы соответствующей латинской буквой. Остальный операции имеют вид «^X», где X принимает одно из следующих значений: `B` соответствует операции удаления, `E` соответствует операции перевода строки, `U,L,D,R` соответствуют операции перемещения курсора вверх, влево, вниз, вправо соответственно.

Формат выходных данных

Вывод состоит из одной или нескольких строк. В конце каждой строки следует выводить символ доллара «$», за которым не должно следовать лишних символов.

Пример

стандартный ввод	стандартный вывод
`Hell^Eworld^Uo`	`Hello$` `world$`

Олимпиада «Спортивное программирование на Урале»

2019-2020 учебный год

Разбор заданий второго (заключительного) этапа

Задача A: Задача для Джорджа

Если попробовать вывести требуемую последовательность чисел, можно заметить, что она не имеет закономерности, но на 2015 шаге последовательность завершается, так как конь заходит в тупик. Дальнейшие шаги перебирать не нужно, ответ для них уже отсутствует.

Так как последовательность довольно быстро заканчивается и не уходит далеко от центра, можно перебрать все ходы, пока последовательность не зайдет в тупик, для этого достаточно матрицы 100x100. Из-за не слишком большого размера последовательности ее можно вычислять “на лету”.

Задача B: Ваня и игра в напёрстки

Для решения задачи достаточно произвести описанные в условия действия. Например, можно воспользоваться массивом, в котором 0 - пустой стаканчик, 1 - стаканчик с шариком. Меняя содержимое указанной пары ячеек, мы можем “перемещать” шарик между стаканчиками.

Задача C: МП-Хорошая строка

Идея : переберем шаблон хорошей строки, вычисляя количество символов, которые не совпадают.

Решение: первое, что нужно понять - если строка “хорошая”, то она также имеет максимальное число подстрок-палиндромов среди эквивалентных “хороших” по числу 0 и 1. Это утверждение можно доказать: если делать циклический сдвиг “хорошей” строки, то число её палиндромов не меняется, а значит они все - максимально палиндромные.

После этого обратим внимание на то, что число несовпавших символов можно вычислить, взяв только строку из нулей и посмотрев на отрезке `[i; i+l-1]` (`i` - позиция, откуда начинается 0-подстрока, а `l` - её размер, т.е. число нулей в изначальной) количество уже находящихся там нулей. Полученная разница между длиной и найденным числом - число несовпадений пополам, то есть то, сколько раз нам понадобится поменять местами два символа из строки. Понятно, что на место единиц в изначальной мы поместим недостающие нули.

Таким образом, надо рассмотреть все позиции `i`, где это возможно, и проделать такую же операцию с 1-подстрокой. Для вычисления можно использовать префикс-сумму по числу нулей на префиксе. Получаем решение O(n).

Задача D: Метро 3030

Решим сначала задачу для случая, когда все моменты времени различны.

Для каждой станции будем поддерживать наименьшее количество поездок, позволяющее доехать до этой станции от начальной. Будем поддерживать это, например, в массиве *ar*. Инициализируется массив следующим образом: элемент с индексом *A* изначально равен 0 (до начальной станции можно добраться за 0 поездок), каждый другой элемент будет равен бесконечности или просто очень большому числу.

Получив информацию, что между *i*-й и *j*-й станциями проедет пара поездов, обновим массив следующим образом: в *i*-й элемент положим минимум из самого *i*-го элемента и из *j*-го элемента, увеличенного на единицу. В *j*-й элемент положим минимум из самого *j*-го элемента и из *i*-го элемента, увеличенного на единицу. Выполнить эти два действия нужно одновременно, это можно сделать следующим кодом:

```
tmp = min(ar[i], ar[j] + 1)
ar[j] = min(ar[j], ar[i] + 1)
ar[i] = tmp
```

Это будет соответствовать тому, что теперь до *i*-й станции можно доехать или старым способом (или никаким, его раньше нельзя было), или доехав до *j*-й станции и совершив одну дополнительную поездку на текущем поезде. Аналогично с *j*-й станцией.

Выполнив эти операции в цикле для каждого из поездов по порядку, мы получим массив *ar*, где для каждой станции записано наименьшее количество поездов, требуемое, чтобы попасть в нее. Ответом будет *B*-й элемент массива, или *You are late*, если в *B*-м элементе записана бесконечность.

Наконец, разберемся, как решать задачу, когда в один момент времени может проехать несколько пар поездов. Разница в том, что если две пары станций даны во входном файле один за другим, но поезда отправляются в один и тот же момент, то мы можем совершить только один переезд.

Общая идея такая же: нужно обновить значения массива *ar* во всех элементах, соответствующих станциям, куда в текущий момент времени приехал поезд. Однако, как и в предыдущем случае, все обновления нужно совершить одновременно, не используя в вычислениях только что обновленные значения. Для этого можно завести, например, новый массив *ar2*, хранящий промежуточные значения массива *ar*. Инициализировать его не обязательно.

Получив информацию, что между всеми парами станций *i[a]* и *j[a]*, где *a* лежит на отрезке от *left* до *right*, проедет пара поездов, запишем новые значения в массив *ar2*, вычислив их таким же образом, как это делали в предыдущем случае. Выполнить эти действия можно сделать следующим псевдокодом:

```
for a = left ... right:
    ar2[i[a]] = min(ar[i[a]], ar[j[a]] + 1)
    ar2[j[a]] = min(ar[j[a]], ar[i[a]] + 1)
```

Таким образом, новые, только что посчитанные значения массива *ar2* не используются в вычислениях. Выполнив все вычисления, нужно занести их обратно в массив *ar*, просто скопировав из массива *ar2*. Однако нужно копировать только те значения, которые могли обновиться. Сделать это можно следующим псевдокодом:

```
for a = left ... right:
    ar[i[a]] = ar2[i[a]]
    ar[j[a]] = ar2[j[a]]
```

Осталось разбить поезда на отрезки подряд идущих поездов, отправляющихся в один и тот же момент времени, и запустить для каждого отрезка поездов сначала первый, потом второй цикл. Итоговая асимптотическая сложность решения - $O(N+M)$.

Задача E: Аи-Ай-Ак-Ат

Преобразуем формулу, воспользовавшись тем, что $x\ xor\ y\ xor\ y = x$:

$$(a[i] + (a[j]\ and\ a[k]))\ xor\ a[t] = x$$
$$a[i] + (a[j]\ and\ a[k]) = x\ xor\ a[t]$$
$$a[j]\ and\ a[k] = (x\ xor\ a[t]) - a[i]$$

Далее воспользуемся принципом meet-in-the-middle. Переберем все пары индексов *i* и *t* таких, что $i < t$. Для каждой такой пары нужно посчитать количество пар индексов *j* и *k* таких, что $i < j < k < t$ и $a[j]\ and\ a[k] = (x\ xor\ a[t]) - a[i]$. Помогает нам в этом то, что значение справа от знака равенства мы знаем, так как мы зафиксировали *i* и *t*.

Чтобы быстро находить количество пар индексов *j* и *k*, воспользуемся предподсчетом. Заранее переберем все пары индексов *j* и *k* такие, что $j < k$, и сгруппируем их по значению $a[j]\ and\ a[k]$. Таким образом, для каждого возможного значения $a[j]\ and\ a[k]$ (их не более 2048, так как все числа не превосходят 2000) мы узнаем все пары $j < k$, дающие это значение.

Вернемся к решению. Для текущей пары *i* и *t* нужно найти все подходящие пары *j* и *k*, такие, что $a[j]\ and\ a[k] = (x\ xor\ a[t]) - a[i]$. У нас уже предподсчитаны все такие пары *j* и *k*, осталось найти среди них количество таких, что $i < j$ и $k < t$.

Чтобы сделать это, во-первых, устроим циклы по *i* и *t* так, чтобы переменная *i* перебиралась по возрастанию во внешнем цикле. Во-вторых, для каждого значения $a[j]\ and\ a[k]$ заведем дерево Фенвика, где будем хранить только значения *k*. А именно, каждое такое дерево Фенвика будет построено над массивом длины *N*, где в *k*-м элементе будет храниться количество индексов *j* таких, что $a[j]\ and\ a[k]$ равно номеру этого дерева. Итого у нас получится массив из 2048 деревьев Фенвика длины *N* каждое.

Это позволит нам считать количество пар *j* и *k* таких, что $a[j]\ and\ a[k] = (x\ xor\ a[t]) - a[i]$ и $k < t$, для этого нужно запросить сумму первых $t - 1$ элементов дерева номер $(x\ xor\ a[t]) - a[i]$. Разберемся теперь, как учесть условие $i < j$. В самой первой итерации цикла $i = 1$, поэтому нам не нужны пары *j* и *k*, в которых $j = 1$. Для этого на ходу переберем все пары *j* и *k*, которые надо исключить; для этого нужен только цикл по *k* внутри внешнего цикла по *i*. Удалим все такие пары из соответствующих деревьев Фенвика, а именно для каждого $k > i$ вычтем из дерева Фенвика номер $a[i]\ and\ a[k]$ единицу из индекса *k*. Итого в деревьях Фенвика останется только информация о таких парах *j* и *k*, в которых $j > 1$. Поэтому количество пар индексов *j* и *k* при $i = 1$ посчитается корректно, с учетом всех условий.

Переходя к $i = 2$, заметим, что нам нужно выполнить условие $j > 2$, а у нас выполняется условие $j > 1$. Точно таким же способом выкинем из деревьев Фенвика все пары *j* и *k*, где $j = 2$, после чего посчитаем количество пар *j* и *k* как префиксную сумму дерева Фенвика. Будем выполнять те же действия на каждой итерации внешнего цикла: переберем все $k > i$ и для каждого такого *k* вычтем из дерева Фенвика номер $a[i]\ and\ a[k]$ единицу из индекса *k*. Отсюда у нас на каждом шаге будут оставаться только такие пары *j* и *k*, что $j > i$.

В конце просуммируем все полученные нами префиксные суммы. Итого в решении есть предподсчет из двух вложенных циклов длины *N* каждый и нахождение ответа, имеющее внешний цикл и два внутренних цикла длины *N* каждый. В каждом из внутренних циклов выполняется запрос к дереву Фенвика, поэтому итоговая асимптотическая сложность по времени - $O(N^2 \cdot log(2048))$ при затратах на память порядка $O(N \cdot 2048)$.

Задача F: Остров сокровищ

Построим на основе поля граф, в котором каждая вершина - это либо клетка воды, либо остров. Для этого переберем все клетки поля. Если клетка поля еще не обрабатывалась, то добавляем новую вершину в граф и помечаем эту клетку пройденной. Если при этом клетка является сушей, то заодно пометим все клетки острова, к которому она принадлежит, и запомним, к какой вершине они принадлежат. Сделать это можно с помощью поиска в глубину. Так мы получили множество вершин графа и для каждой клетки поля знаем, к какой вершине она принадлежит.

Теперь добавим ребра графу. Для этого опять переберем все клетки поля. Для каждой клетки будем смотреть на две соседние: справа и снизу (если они есть). Пусть мы находимся в клетке *[i, j]*, которой соответствует вершина *u*. Пусть соседней клетке соответствует вершина *v*. Если u != v, то добавим в граф ребро *(u, v)*.

В результате описанных действий получен искомый ориентированный граф, найдем в нем компоненты сильной связности. Для каждой компоненты посчитаем количество островов, которые ей принадлежат. Для этого достаточно посчитать количество вершин, которые принадлежат данной компоненте и являются островами.

Теперь обойдем наш исходный граф в порядке топологической сортировки и для каждой вершины посчитаем максимальное количество островов, которое можно посетить из компоненты связности, к которой принадлежит эта вершина. Изначально для каждой компоненты эта величина равна количеству островов в компоненте. Обновлять величину будем так: получив очередную вершину исходного графа, смотрим, к какой компоненте она принадлежит, и обновляем соответствующую компоненту, перебирая соседей этой компоненты и поддерживая максимум из текущего значения и количества островов в текущей компоненте, а также максимальное количество островов, которое можно посетить из соседней компоненты.

Проверим, равно ли максимальное количество островов, которое можно посетить из компоненты, к которой принадлежит вершина, соответствующая стартовой клетке, общему числу островов. Если да, то ответ *YES*, в противном случае *NO*.

Задача G: Редактор текста

Для решения задачи достаточно реализовать действия, описанные в условия.

Для поддержания содержимого текста создадим массив строк. Будем посимвольно считывать введенную строку и обрабатывать поступающие команды, поддерживая текущую позицию в массиве (удобно хранить координаты *x* и *y*, где *x* - номер строки, *y* - номер символа в ней). Важно не забывать проверять выход за границы массива.

Олимпиада «Спортивное программирование на Урале»

2019-2020 учебный год

Критерии определения призеров и победителей

Критерии определения призеров и победителей заключительного этапа 2019-2020 учебного года

Согласно решению жюри Олимпиады “Спортивное программирование на Урале”, были установлены следующие критерии для присуждения дипломов I, II, III степеней:

Степень диплома	**Разбалловка**	**Участников**
		100
5-11 классы	Максимум 7 задач	
I	7 задач	1
II	От 4 задач до 6 включительно	9
III	3 задачи	14
Количество призеров и победителей заключительного этапа		24
Процент призеров и победителей от общего числа участников заключительного этапа		24%