Задача A
Калькулятор

Студентам ППИ необходим калькулятор, который может делать расчёты с числами различной длины (размерности). Для разработки программы руководство института решило использовать самих студентов. Вам требуется написать модуль сложения целых неотрицательных чисел.

Входные данные (calc.in):

Две строки, на каждой из них по одному числу в десятичной системе счисления без ведущих нулей. Числа не превышают 10^{10000}.

Выходные данные (calc.out):

Результат сложения двух чисел в десятичной системе счисления.

Пример:

calc.in	calc.out
1234567890 998877661111	1000112229001

Задача B
Кукла

Современные копировальные аппараты так качественно делают цветные копии денежных банкнот, что без специальной экспертизы и не отличишь от оригинала. Эти технологии доступны широким массам и их использование приобретает катастрофический характер. Но так как при копировании сохраняются и номера купюр, то иногда можно определить наличие фальшивки по совпадению номеров на 2-х и более купюрах. Ваша задача написать программу, которая определит, является ли данная пачка денег «куклой» или нет, используя только сравнение серийных номеров купюр.

Вроде лёгкая задача, но есть одно «но»: злоумышленники используют сильно помятые деньги и после сканирования программа распознавания номеров может ошибиться, но не более чем в одной цифре (статистические данные).

Входные данные (money.in):

Входной файл содержит произвольное количество строк. Первая строка содержит единственное число n – количество купюр ($n \le 100000$). Последующие n строк содержат по одному серийному номеру из семи цифр.

Выходные данные (money.out):

Выведите «YES», если в пачке есть хотя бы две купюры с совпадением не менее шести цифр из семи, в противном случае выведите «NO».

Пример:

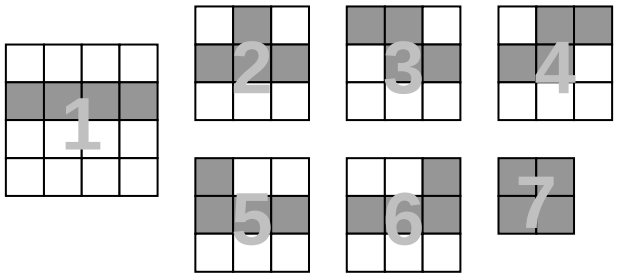
money.in	money.out
2 1234567 1244567	YES
2 1234567 1243567	NO

Задача C
Тетрис

Напишите игру «Тетрис». Далее правила. Падают различные геометрические фигуры внутри плоского «стакана», данные фигуры человек пытается разместить таким образом, чтобы полностью заполнить горизонтальные линии, за это он получает игровые очки. Полностью заполненная горизонтальная линия стирается, а всё содержимое стакана выше этой линии опускается на одну линию вниз.

Ваша задача написать программу, которая по входным данным будет определять, сколько очков набрал игрок и как выглядит тетрис-стакан на момент проигрыша игрока.

Далее указаны все фигурки, которые могут падать в стакан.



Манипуляции, которые может осуществлять игрок:

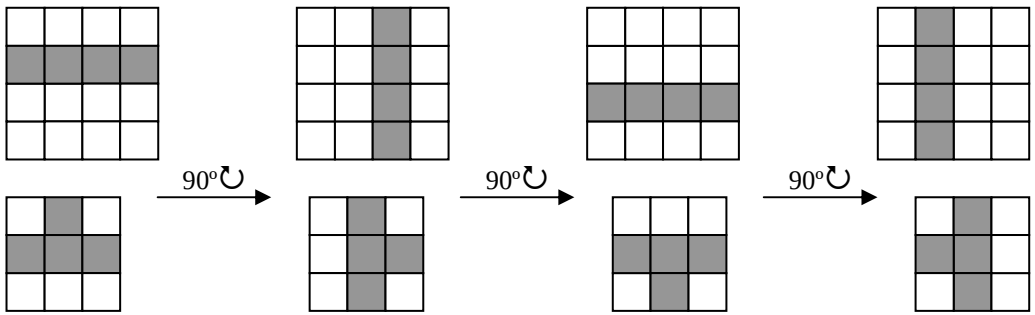
U – поворот фигуры на 90^0 по часовой стрелке (см. рисунок ниже);

D – бросить фигуру вниз (следующее действие игрока относится уже к следующей падающей фигуре);

L – передвижение фигуры влево на 1 деление;

R – передвижение фигуры вправо на 1 деление;

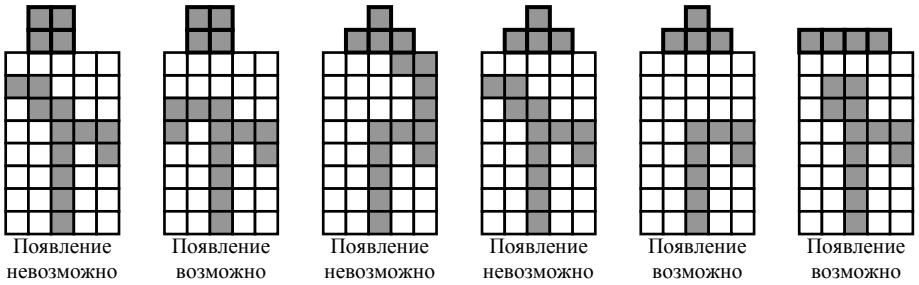
S – никаких действий не выполнил.



После каждого из действий игрока фигура перемещается вниз на одно деление, если это возможно, иначе появляется следующая фигура или наступает конец игры.

Фигура появляется в стакане, только если она целиком может разместиться без наложения на уже имеющиеся фигуры, иначе происходит конец игры. Фигура появляется вверху в центре. В случае несимметричности (например, первая фигура в стакане шириной в пять квадратиков), она появится на один квадратик левее (в левом верхнем углу для рассмотренного выше примера).

Манипулирование фигурой начинается с момента её полного появления в стакане.



Входные данные (tetris.in):

Первая строка содержит два числа разделённые пробелом: ширину и высоту стакана ($4 \le W, H \le 10$). Во второй строке записана последовательность номеров выпадающих фигур (строка представляет собой не более 100 цифр), после выпадения последней фигуры последовательность выпадения начинается сначала. Третья строка представляет ничто иное, как действия игрока.

Выходные данные (tetris.out):

Первая строка – количество заработанных очков (1 линия – 1 очко). Со следующей строки – внешний вид стакана (Буква O – отсутствие фигуры в данном квадратике, # – наличие фигуры).

Пример:

tetris.in	tetris.out
5 8 127 UDDLDD	0 ####0 ##000 ###00 0###0 00#00 00#00 00#00 00#00
4 4 37 USSSSSSSSSSSSSSSSS	0 0000 00#0 0##0 0#00
4 4 7 LDRDDDDDDDDDDLLLRR	2 0##0 0##0 0##0 0##0

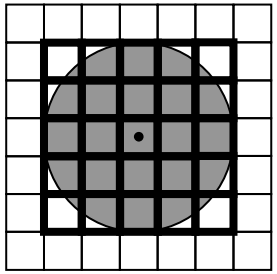
Задача D

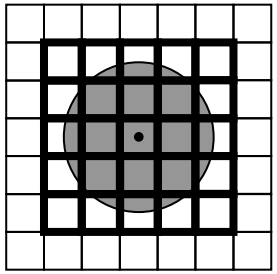
Драка

Как говорится: «После драки кулаками не машут». Вот и мы не будем махать, а закрасим капли крови, оставшиеся на полу (что за драка без крови).

Пол у нас состоит из мелкой квадратной плитки. А потому местоположение крови (круги диаметром d и координатами центра x,y) будем задавать в целых положительных числах. Клетка с координатами (1, 1) находится в левом верхнем углу.

Каждое пятно должно быть закрашено в форме квадрата с минимально возможной площадью (краска нынче дорогая).





Входные данные (fight.in):

Первая строка содержит 2 числа: размер пола в клетках ($1 \le W, H \le 200$). Вторая строка содержит одно число: количество пятен крови ($n < 10000$). Последующие n строк содержат описания пятен крови ($1 \le x, y, d$). Центр пятна крови всегда находится на пересечении диагоналей плитки. Кровь не попала за пределы пола. Капля крови с единичным радиусом испачкает ровно 1 плитку.

Смотри пример пятен диаметрами 5 и 4 соответственно. Клетки, выделенные утолщённой обводкой, необходимо закрасить.

Выходные данные (fight.out):

Одно единственное число – количество плиток, которое надо закрасить.

Пример:

fight.in	fight.out
10 10 2 5 5 3 5 6 3	12

Задача E

Массив

В программировании часто выделяют место под хранение N-го количества данных одного типа с помощью объявления массива. В C и C++ форма записи объявления выглядит так: name[a_1][a_2]..[a_m]. Это объявлен m-мерный массив под хранение N = a1·a2·…·am данных.

Задача: необходимо выяснить, сколько существует различных способов объявить массив ровно под N данных при $a_i \geq 2$.

Входные данные (array.in):

Одно единственное число N ($1 \leq N \leq 1000000$).

Выходные данные (array.out):

Одно единственное число – количество способов объявить массив ровно под N элементов.

Пример:

array.in	array.out
12	8

Подсказка: [2][2][3]; [3][2][2]; [2][3][2]; [3][4]; [4][3]; [2][6]; [6][2]; [12].

Задача F

Крестики-нулики

Дана позиция игры в крестики-нулики. Остался один ход до победы крестиков. Найдите этот ход.

Входные данные (xo-xo-xo.in):

Представляют собой 3 строки по три символа в каждой.

X – крестик;

O – нулик (буква);

* – сюда ещё никто не ходил ☺

Выходные данные (xo-xo-xo.out):

Выведите поле в формате ввода, но с победным крестиком.

Пример:

xo-xo-xo.in	xo-xo-xo.out
X0X 0X0 **0	X0X 0X0 X*0
X0X 0*0 X0X	X0X 0X0 X0X
X*X *** ***	XXX *** ***