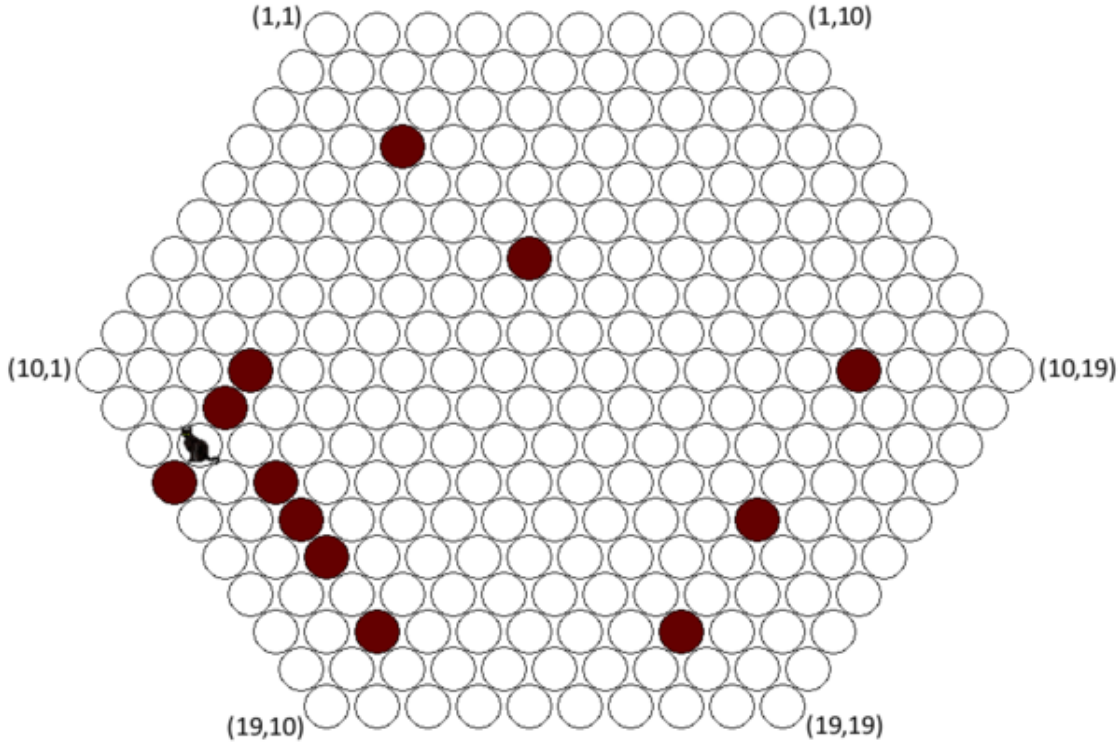
cat: Поймать кота

«Окружай» — простая пошаговая игра. Играют два игрока. Один из них — вы, другой — злой кот. Кот находится в конечном шестиугольном поле. Во время своего хода кот обязательно переходит в одну из соседних шестигональных не заблокированных клеток. На вашем ходу вы можете заблокировать любую клетку до конца игры. Вы начинаете игру, делая первый ход.

Кот выигрывает игру, если достигает границы поля. Вы выигрываете игру, если сможете полностью заблокировать кота — если у него не останется допустимых ходов.

Возможная игровая ситуация изображена на картинке ниже. Заблокированные клетки заполнены. Также на картинке изображена система координат. Кот начинает в клетке (10, 10).



Формат входных данных

Во входном файле нет ничего.

Поведение кота полностью детерминировано. Вам дана реализация алгоритма, по которому действует кот, в файле catlib.py. Также у вас есть несколько программ для взаимодействия с котом:

Скрипт cat-commandline.py — просто консольный интерфейс к библиотеке. Он считывает ваши ходы с консоли и пишет ходы кота на консоль. Так же именно так работает чекер в этой задаче.

Скрипт cat-pygame.py — графическая оболочка игры.

Каждый раз, когда вы выигрываете или проигрываете игру, библиотека автоматически добавляет лог игры в файл log.txt. Так, если вы выиграли игру в графическом интерфейсе, в логе вы найдете решение задачи, которое должно получить OK.

Формат выходных данных

Вывод вашей программы для простой подзадачи должен содержать любую последовательность валидных ходов, которая ловит кота.

Вывод вашей программы для простой подзадачи должен содержать последовательность валидных ходов, длиной не больше 20, которая ловит кота.

В выходной файл выведите k строк с координатами клеток, в которые вы совершали ходы, точно скопированные из `log.txt`, и ничего больше, без строк «NEW GAME», «WINNER:», лишних пробелов и переводов строк.

Примеры

стандартный ввод	стандартный вывод
	4 4 5 5 6 6

Замечание

Пояснение к примеру: такое решение имеет корректный формат вывода, но получает WA.

kirk: Кирк или Пикард?

Если вы торопитесь и не имеете под рукой псевдослучайного генератора, попробуйте попросить ближайшего Треккера сказать, кто им больше нравится: Кирк или Пикард? Эта задача тоже про псевдослучайные генераторы. Мы дадим вам большую часть вывода псевдослучайного генератора, а вы заполните пробелы.

Мы сгенерировали последовательность псевдослучайных битов с помощью линейного конгруэнтного генератора (возможно, с довольно-таки плохими свойствами). Для большей информации, почитайте статью на Википедии. Последовательность состоит из 4 000 000 значений $\{0,\ 1,\ 2\}$. 0 и 1 это биты, выданные генератором. 2 это пробел – бит с неизвестным значением, таких ровно 4 000.

В простой версии угадайте как минимум 2 000 пробелов верно. В сложной версии вам нужно угадать 2 500 пробелов.

Формат входных данных

Входной файл `kirk-picard.in` содержит одну последовательность в системе счисления по основанию 81. Более точно, последовательность разделена на группы по четыре значения, и группа $(a,\ b,\ c,\ d)$ закодирована символом с ASCII кодом $(33\ +\ 27a\ +\ 9b\ +\ 3c\ +\ d)$.

Формат выходных данных

Вам необходимо отправить только выходной файл из 4 000 нулей или единиц – заполненные пробелы в том же порядке, как во входном файле.

Примеры

стандартный ввод	стандартный вывод
++2+a1	0110

Замечание

В примере закодирована последовательность `010101010122010121010121`. Она не совсем случайна, похоже, что в ней просто чередуются нули и единицы. Ответ к примеру верен на 75%.

hashset: Взлом hashset-а

Нет другой настолько же часто используемой структуры данных на практике как хешсеты и хешмэпы. Их постоянно используют, и почти всегда оправданно. К сожалению, часто люди переоценивают их мощь. Интернет полон вредных советов вида «просто напиши хешсет, все будет работать за $O(1)$», или «не беспокойся, на реальных данных все всегда работает». Мы надеемся, что вы не повторите такой ошибки в будущем. В этой задаче вам нужно взломать стандартный хешсет.

Посмотрим на следующий код на языке C++:

```
1 // C++11
2 #include <iostream>
3 #include <unordered_set>
4 int main() {
5     long long tmp;
6     std::unordered_set<long long> hashset;
7     while (std::cin >> tmp) hashset.insert(tmp);
8 }
```

Мы скомпилировали эту программу с помощью компилятора `g++ (GCC) 8.1.1 20180712 (Red Hat 8.1.1-5)` и запустили. Программа просто считывает набор чисел и вставляет их в хешсет. Должно работать быстро, не так ли?

Вам нужно вывести не больше 50 000 64-битных чисел, вставка которых таким алгоритмом в хешсет займет на сервере больше двух секунд.

Формат выходных данных

Файл, который вы отправляете, должен содержать сначала число n ($1 \leqslant n \leqslant 50\,000$) — количество элементов в последовательности, а потом последовательность из n 64-битных чисел от -2^{63} до $2^{63} - 1$, разделенных пробелами.

Примеры

стандартный ввод	стандартный вывод
4 1 2 3 4	

judging: Тестирующая система — чтобы тестировать!

У этой задачи есть две подзадачи, которые решаются отдельно. У них общий принцип: вам дано описание задачи и набор программ. Каждая из программ содержит функцию `solve`, которая пытается решать эту задачу. Все программы написаны на легкочитаемом подмножестве Python и не должны содержать каких-либо специфичных фич языка.

Ваша цель – определить, какие из программ корректно решают задачу, а именно, для всех возможных корректных входов, программа завершается за конечное время и возвращает верный результат.

1 Простая подзадача

Даны два числа n и k ($1 \leqslant k \leqslant n \leqslant 100$). Необходимо вычислить биномиальный коэффициент $\binom{n}{k}$. Например, для $n = 10$ и $k = 2$ нужно вывести 45.

Программы содержатся в архиве `judging-easy.zip`. Каждая из них содержит функцию `solve`, которая принимает два параметра: n и k. Выходной файл должен содержать ровно 11 слов, разделенных пробельными символами, i-е из которых описывает программу `j1_i.py`. Выведите `good`, если она верна, и `bad` иначе.

2 Сложная подзадача

Задан связный неориентированный взвешенный граф. Найдите длину кратчайшего пути между двумя заданными вершинами графа. Программы содержатся в архиве `judging-hard.zip`. Каждая программа содержит функцию `solve`, ожидающую 5 параметров:

- n: количество вершин в графе ($2 \leqslant n \leqslant 100$). Вершины пронумерованы от 0 до $n - 1$.
- m: количество ребер в графе ($n - 1 \leqslant m \leqslant \frac{n(n-1)}{2}$),
- `edge_list`: список ребер графа. Каждое ребро это тройка (p, q, w), где p и q это концы ребра, а w — его вес ($1 \leqslant w \leqslant 1000$). Никакие два ребра не соединяют одни и те же вершины.
- *start* и *end*: вершины, между которыми мы хотим найти кратчайший путь.

Для удобства, в архиве также лежат файлы `j2_sample.{in,out}` и `j2_read_input.py`: пример графа, ответ для него, и парсер графа в удобный для функции `solve` формат.

boundingbox: Баундинг Бокс

У Мирко-старшего есть сын — Мирко-младший. Недавно, старший Мирко купил младшему много пластиковых шаров разных размеров. Младший многому научился, играя с ними.

Со временем младший становится все старше и старше, поэтому старший дает ему все более сложные задачи. Однажды Мирко попросил Мирко положить все свои шары в коробку, чтобы их было удобнее хранить. К сожалению, Мирко не справился быстро с этой задачей. Поможете ли вы ему?

Вам дан пустой ящик размеров $w \cdot h \cdot d$. Также вам дано несколько шаров разного размера. Вам нужно поместить все шары в ящик. Все шары должны быть полностью внутри шара. Никакие два шара не могут пересекаться. Любое такое расположение будет принято. В частности, при построении своего расположения вы можете игнорировать силы гравитации и оставлять шары плавающими в воздухе без опоры.

Формат входных данных

В первой же строке теста к задаче, который вы можете скачать, дано целое число t — количество тестов во входном файле. Каждому тесту предшествует пустая строка.

Первая строка теста содержит **вещественные** числа w, h, d ($1 \leqslant w, h, d, \leqslant 250$) — размерности ящика. Ящик имеет противоположные углы в координатах $(0, 0, 0)$ и (w, h, d).

Вторая строка теста содержит целое число n — количество различных типов шаров. Каждый тип шаров описывается строкой с двумя числами c, r. Целое число c ($1 \leqslant c \leqslant 150$) означает количество копий этого типа шаров которое у вас есть. **Вещественное** число r ($0.001 \leqslant r \leqslant 15$) означает радиус каждого шара такого типа. Всего в тесте не больше 150 шаров, и все вещественные числа даны не больше, чем с восемью знаками после запятой.

В легкой подзадаче $1 \leqslant n \leqslant 2$, в сложной подзадаче $1 \leqslant n \leqslant 5$.

Формат выходных данных

Для каждого теста, выведите несколько строк. Выводите пустую строку после каждого теста.

Вывод для каждого теста должен содержать столько строк, сколько всего шаров должно быть в ящике. Для каждого шара выведите четыре числа i, x, y, z — тип шара, и позицию центра шара (x, y, z) в ящике. Типы шаров нумеруются от 1 до n в порядке их появления во входных данных.

Шары могут быть выведены в любом порядке, но убедитесь, что вы вывели ровно все множество шаров, которое у вас есть. Все шары должны поместиться в ящике. Гарантируется, что решение существует.

Ваш ответ должен иметь абсолютную погрешность не больше, чем 10^{-6}.

Два шара с радиусами r_1 и r_2 соответственно пересекаются, если их центры расположены ближе, чем на расстоянии $r_1 + r_2 - 10^{-6}$. Точка считается не внутри ящика, если она выходит за одну из сторон ящика хотя бы на $10^{-}6$.

Примеры

стандартный ввод	стандартный вывод
1 8 8 8 2 1 4 2 0.9	1 4 4 4 2 1 7 1 2 1 7 7

knowledge: Тест на знание

Соревнования по программированию это круто! Даже если вы ничего не знаете, вы все еще можете успешно решать много задач, если вы умненькие. Однако, этого не будет достаточно для этой задачи.

Вам задан кроссворд. Решите его.

Формат входных данных

Входной файл `knowledge-easy/hard.in` содержит один кроссворд. В первой строке находятся два числа r и c – количество строк и столбцов. Следующие r строк содержат по c символов `#` и `.`: белые и черные квадраты в кроссворде. Следующая строка содержит число a — количество горизонтальных слов. Следующие a строк содержат по два числа r_i и c_i - координаты самой левой буквы в этом слове; а также подсказку для этого слова. После этого идет число d — количество вертикальных слов. Следующие b строк содержат по два числа r_i и c_i - координаты самой верхней буквы в этом слове; а также подсказку для этого слова.

У кроссворда единственное верное решение.

Формат выходных данных

Вам необходимо отправить только выходной файл: возьмите кроссворд из входного файла и замените все `#` на маленькие буквы английского алфавита (`a-z`). Выходной файл должен содержать ровно r строк, каждая из c символов.

Примеры

стандартный ввод	стандартный вывод
3 4 ..#. #### ..#. 1 1 0 Your favorite competition 1 0 2 Traveling salesman problem acronym	..t. ipsc ..p.

hiddentext: Скрытый текст

Обычная практика для показа документов с секретной информацией — применить к тексту какую-нибудь постобработку, сделав чтение важных частей документа невозможных, например сильным размыванием изображением или пикселизацией картинки.

К сожалению, на практике такие методы могут оказаться недостаточно хороши. Взломщик может восстановить скрытое изображение, по крайней мере частично. Это и будет вашей целью в этой задаче.

Вам даны два изображения. В простом изображении у вас есть четкая картинка с черными буквами и белым фоном, а в сложной версии картинка содержит некоторый шум. В обоих случаях, часть изображения была отредактирована чтобы спрятать информацию. Восстановите информацию, и отправьте доказательство в систему.

Формат входных данных

Вам дан Portable Network Graphics (PNG) файл.

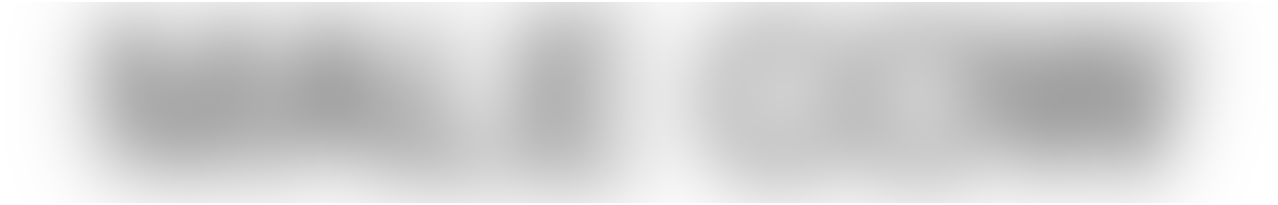
Формат выходных данных

Для каждого файла отправьте файл с решением задачи, содержащий одно английское слово ВСЕМИ ЗАГЛАВНЫМИ БУКВАМИ. Если вы отправите правильный ответ, но не ЗАГЛАВНЫМИ БУКВАМИ, вы получите неверную посылку.

Слово, которое вам нужно отправить, однозначно задается скрытой частью файла. В сложном файле искомое слово содержит от 6 до 9 букв, и начинается с буквы P.

Примеры

Обработанное изображение:



Исходное изображение:

Ответ — BULL.

dazzling: Ослепительные цифры

Числа, в которых цифры повторяются, ужасно скучные. Вы только посмотрите на них: 44 — скучное число, 474 — тоже скучное, 1 000 000 — настолько скучное, что аж тошнит.

Числа, в которых все цифры различны, невероятно интересные. При лишь одном взгляде на такое число сразу обнаруживаешь что-нибудь прикольное! Вы только посмотрите: 52107 — прекрасное число, а 9087432156 одно из самых красивых чисел на земле.

Вам дано число n. Представьте n в виде суммы $n = a_1 + \ldots + a_k$, где каждое из чисел a_i не должно быть скучным, а число слагаемых k должно быть минимально возможным.

При этом, сами числа a_i могут повторяться, и одна и та же цифра может встречаться несколько раз в разных числах a_i.

Формат входных данных

Входной файл `dazzling.in` содержит число t — количество тестовых примеров. Описание каждого тестового примера состоит из единственного числа n.

В простой версии задачи $t = 1000$, $n \leqslant 10^4$. В сложной версии задачи $t = 30\,000$, $n \leqslant 2 \cdot 10^9$.

Формат выходных данных

Для каждого тествого примера выведите сначала число k — искомое число слагаемых, а затем сами слагаемые.

Примеры

стандартный ввод	стандартный вывод
3 123 99 999123456	1 123 2 90 9 2 20469135 978654321

code: Изучение кода

В этой задаче вам дана программа, которая каким-то образом, рано или поздно, выдает одно читаемое английское слово. Ваша цель: восстановить это слово любыми силами.

Формат входных данных

Входной файл `code-inception.py/cc` содержит программу на Python или C++. Каждая программа выдает одно и то же слово.

Формат выходных данных

Вам необходимо отправить только выходной файл с одним искомым словом **большими** английскими буквами.

Примеры

стандартный ввод	стандартный вывод
`for x in "olleh"[::-1]:` `print(x)`	`HELLO`

Замечание

Заметьте, что выводить слово нужно большими буквами.

makemeaninteger: Сделай*из-меня+[число!]

В этой задаче вам необходимо с помощью языка JavaScript (стандарт ECMA-262) получить все числа от 0 до 1000. К сожалению, в программе можно использовать лишь символы `![]+-*`.

Формат входных данных

Входной файл пуст.

Формат выходных данных

Вам необходимо отправить только выходной файл из 1001 строки: для каждого i от 0 до 1000 строка с номером $i + 1$ должна содержать программу на языке JavaScript, которая выдает число, равное i (для результаты программы должен выполняться предикат `typeof(result)=«number»` и его значение совпадает с i).

В простой версии каждая программа должна иметь длину не более 5000 символов. В сложной версии каждая программа должна иметь длину не более 200 символов.

Примеры

стандартный ввод	стандартный вывод

	`+![]`
	`+!![]`
	`... (еще 999 строк)`

Замечание

Для тестирования можно воспользоваться браузером Google Chrome: откройте Developer Console (Ctrl-Shift-I / Command-Alt-I), после этого просто вводите выражения вида «+![]» и нажимайте Enter.