Технокубок 2017/2018. Первый отборочный раунд. Условия и разбор задач.

A. k-Округление

ограничение по времени на тест

1 секунда

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Для заданного натурального числа n его k-округлением назовём такое минимальное целое положительное число x, что xзаканчивается на k или более нулей в системе счисления по основанию 10 и одновременно делится на n.

Например, 4-округлением числа 375 будет число $375 \cdot 80 = 30000$. Число 30000 — минимальное такое число, которое заканчивается на 4 или более нуля и делится на 375.

Напишите программу, которая осуществляет k-округление числа n.

Входные данные

В единственной строке входных данных записаны два целых числа n и k ($1 \le n \le 10^9$, $0 \le k \le 8$).

Выходные данные

Выведите k-округление числа n.

Примеры

входные данные

```
375 4
```

выходные данные

```
30000
```

входные данные

```
10000 1
```

выходные данные

```
10000
```

входные данные

```
38101 0
```

выходные данные

```
38101
```

входные данные

```
123456789 8
```

выходные данные

```
12345678900000000
```

A - k-Округление

Заметим, что число x заканчивается хотя бы на k нулей, если максимальная степень 2, на которую делится число x, не менее k, и максимальная степень 5, на которую делится число, также не менее k. Тогда посчитаем максимальные степени 2 и 5, на которые делится заданное n. Если одна из степеней меньше чем k, домножим на нужное количество 2 и 5.

Так же можно заметить, что ответ это НОК(n, 10k).

B. Какой этаж?

ограничение по времени на тест
1 секунда
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод
стандартный вывод

В доме Поликарпа на каждом этаже *одинаковое* количество квартир. К сожалению, Поликарп не помнит, сколько именно квартир на каждом этаже, зато он помнит, что квартиры пронумерованы от 1 снизу вверх. Таким образом, в нумерации сначала идут квартиры первого этажа, затем второго и так далее. Поликарп не помнит точное количество квартир в доме, вы можете считать дом бесконечно высоким (то есть содержащим бесконечное количество квартир). Обратите внимание, этажи нумеруются с 1.

Про некоторые квартиры Поликарп помнит, на каких этажах они расположены. Гарантируется, что воспоминания Поликарпа непротиворечивы. То есть существует дом с одинаковым количеством квартир на каждом этаже, такой что квартиры из воспоминания Поликарпа имеют те этажи, которые он запомнил.

Может ли он на основании этой информации указать точный этаж для квартиры n?

Входные данные

В первой строке записаны два целых числа n и m ($1 \le n \le 100$, $0 \le m \le 100$), где n — номер квартиры, этаж которой надо определить, и m — количество квартир в воспоминании Поликарпа.

Далее следуют m строк, которые описывают воспоминание Поликарпа: каждая из этих строк содержит пару чисел k_i, f_i ($1 \le k_i \le 100$, $1 \le f_i \le 100$), которая означает, что квартира k_i находится на этаже f_i. Все значения k_i — различны.

Гарантируется, что воспоминания Поликарпа непротиворечивы.

Выходные данные

Выведите номер этажа, на котором расположена квартира n, если его номер однозначно определяется по воспоминанию Поликарпа. Выведите -1, если однозначно восстановить номер этажа невозможно.

Примеры

входные данные

```
10 3
6 2
2 1
7 3
```

выходные данные

```
4
```

входные данные

```
8 4
3 1
6 2
5 2
2 1
```

выходные данные

```
-1
```

Примечание

В первом примере 6-я квартира находится на 2-м этаже, а 7-я уже на 3-м, следовательно, 6-я квартира последняя на своём этаже, а всего на этаже 3 квартиры. Таким образом, 10-я квартира находится на 4-м этаже.

Во втором примере на каждом этаже может быть 3 или 4 квартиры, поэтому мы не можем узнать, на каком этаже находится 8-я квартира.

B - Какой этаж?

Сохраним переменную ans, отвечающую за ответ, которая изначально будет равна -1.

Давайте переберём количество квартир на этаже от 1 до 100. Пусть эта величина равняется cf. Теперь мы должны проверить, что такое количество квартир является корректным для заданного набора входных данных. Если $\lceil \frac{k_i}{cf} \rceil \neq f_i$ хотя бы для какого-нибудь $i \in [1..m]$, то это количество квартир некорректно, пропускаем его.

Теперь же, если $ans \neq -1 \ \&\& \ \lceil \frac{n}{cf} \rceil \neq ans$, то значит, мы не можем однозначно определить, на каком этаже располагается квартира n. Выводим -1. Иначе же просто присвоим $ans = \lceil \frac{n}{cf} \rceil$.

C. Возможно, вы имели в виду...

ограничение по времени на тест
1 секунда
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод
стандартный вывод

Текстовый редактор Beroffice имеет широкие возможности по работе с текстами. Одна из возможностей — автоматический поиск опечаток и формирование предложения по их исправлению.

При наборе текста в Beroffice используются только строчные буквы английского алфавита (то есть 26 букв от `a` до `z`). При наборе слова Beroffice предполагает, что слово набрано с опечаткой, если встречаются три или более согласные буквы подряд. Единственное исключение — если блок подряд идущих согласных букв состоит из одинаковых букв, то этот блок (даже если его длина больше или равна трём) не считается опечаткой. Формально, слово набрано с опечаткой, если в слове существует блок не менее чем из трёх согласных подряд и эти согласные — не одна и та же буква.

Например:

- следующие слова набраны с опечатками: «`hellno`», «`hackcerrs`» и «`backtothefutttture`»;
- следующие слова набраны без опечаток: «`hellllllooooo`», «`tobeornottobe`» и «`oooooo`».

Редактор Beroffice при обнаружении слова с опечаткой вставляет минимальное количество пробелов в слово (разделяя его на несколько слов) так, что каждое из получившихся слов набрано без опечаток.

Реализуйте эту функциональность редактора Beroffice. Считайте гласными только буквы '`a`', '`e`', '`i`', '`o`' и '`u`'. Все остальные буквы следует считать согласными.

Входные данные

В единственной строке входных данных содержится непустое слово, состоящее из строчных букв английского алфавита. Длина слова — от 1 до 3000 букв.

Выходные данные

Выведите заданное слово без изменений, если оно не содержит опечаток. Если слово содержит хотя бы одну опечатку, то вставьте в него наименьшее количество пробелов так,

что каждое из получившихся слов не содержит опечаток. Если решений несколько, то выведите любое из них.

Примеры

входные данные

```
hellno
```

выходные данные

```
hell no
```

входные данные

```
abacaba
```

выходные данные

```
abacaba
```

входные данные

```
asdfasdf
```

выходные данные

```
asd fasd f
```

C - Возможно, вы имели в виду...

Давайте решать задачу жадно. Посмотрим, где находится самая левая опечатка. Очевидно, это будут такие три последовательные буквы si - 1, si и si + 1 такие, что все они согласные и среди них есть хотя бы две различные. Очевидно, что выгоднее всего будет "порезать" строку после позиции i, потому что в таком случае наш префикс останется корректным, а в суффикс мы перенесём всего лишь одну букву. Таким образом, можно пройти слева направо по строке и резать её в позициях с некорректными тройками букв. Только необходимо не забывать, что после разреза строки в позиции i мы должны дальше рассматривать её с позиции i + 2, а не i + 1.

D. Телефонная книга Поликарпа

ограничение по времени на тест

4 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

В телефонной книге Поликарпа записаны n телефонных номеров. Каждый номер — это целое 9-значное число, которое начинается с отличной от 0 цифры. Все номера — различны.

На телефоне Поликарпа установлена последняя версия операционной системы Berdroid. При частичном наборе номера Berdroid подсказывает все номера из телефонной книги, подстрокой которых является набранная последовательность цифр. Например, если в телефонной книге Поликарпа записаны три номера 123456789, 100000000 и 100123456, то:

- при наборе последовательности цифр 00 будут подсказаны номера 100000000 и 100123456,
- при наборе последовательности цифр 123 будут подсказаны номера 123456789 и 100123456,
- а при наборе последовательности цифр 01 будет подсказан только номер 100123456.

Для каждого номера в телефонной книге Поликарпа найдите наименьшую по длине последовательность цифр, которую надо набрать, чтобы Berdroid подсказал только один этот номер.

Входные данные

Первая строка входных данных содержит единственное целое число n ($1 \le n \le 70000$) — количество номеров в телефонной книге Поликарпа.

Далее следуют сами номера по одному номеру в строке. Каждый номер — это целое положительное 9-значное число, которое начинается с цифры от 1 до 9. Все номера — различны.

Выходные данные

Выведите ровно n строк: i-я строка должна содержать наиболее короткую непустую последовательность цифр, при наборе которой будет подсказан только один i-й номер из телефонной книги. Если таких последовательностей несколько, выведите любую из них.

Примеры

входные данные

```
3
123456789
100000000
100123456
```

выходные данные

```
9
000
01
```

входные данные

```
4
123456789
193456789
134567819
934567891
```

выходные данные

```
2
193
81
91
```

D - Телефонная книга Поликарпа

Для каждой строки необходимо найти кратчайшую подстроку, которая не встречается как подстрока в других строках. Для этого для каждой подстроки будем хранить номер строки, в которой она встречается, или -1, если она встречается более чем в одной строке. После этого пройдемся по всем подстрокам, для которых значение не равно -1, и попробуем обновить ответ для той строки, которая записана как значение.

E. Перенумерация тестов

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Совсем недавно завершилась Всеберляндская олимпиада по программированию! Теперь Владимир хочет добавить прошедшее соревнование в качестве тренировки на популярный сайт Codehorses.

К сожалению, в архиве олимпиады не царит идеальный порядок. Например, файлы с тестами названы не в соответствии с некоторой логикой, а произвольным образом.

Владимир хочет переименовать файлы с тестами так, чтобы их имена составляли различные целые числа от 1 без промежутков, то есть имели имена «`1`», «`2`», ..., «n», где n — общее количество файлов с тестами.

Некоторые из файлов содержат тесты из условия (примеры), а остальные — это просто файлы с обыкновенными тестами. Возможна ситуация, когда примеров нет, а также ситуация, когда все тесты являются примерами. Владимир хочет переименовать файлы так, чтобы примеры из условия составляли первые несколько тестов, а все следующие по порядку файлы содержали обыкновенные тесты.

Единственная операция, которую Владимир осуществляет — это переименование файлов с помощью операции командной строки «`move`». В результате своей работы Владимир хочет написать скрипт, каждая строка которого имеет вид «`move file_1 file_2`», которая обозначает, что файл «`file_1`» следует переименовать в файл «`file_2`». Если файл «`file_2`» существовал на момент выполнения строки скрипта, то этот файл будет затёрт. После строки «`move file_1 file_2`» файл «`file_1`» более не существует, но обязательно существует файл «`file_2`», содержащий содержимое файла «`file_1`» до этой команды «`move`».

Помогите Владимиру написать скрипт, который содержит минимальное количество строк, такой что после его выполнения:

- все примеры составляют несколько первых тестов и файлы имеют названия «1», «2», ..., «e», где e — количество файлов с примерами;
- остальные файлы содержат оставшиеся обыкновенные тесты и имеют названия «$e + 1$», «$e + 2$», ..., «n», где n — общее количество всех тестов.

Входные данные

В первой строке записано целое число n ($1 \le n \le 10^5$) — количество файлов с тестами.

Далее следует n строк, каждая описывает один файл с тестом. Строка имеет вид «name_i type_i», где «name_i» — имя файла, а «type_i» равно «1», если i-й файл содержит пример из условия и «0» — если обычный тест. Имя каждого файла — строка из цифр и строчных букв латинского алфавита, длина имени файла содержит от 1 до 6 символов. Гарантируется, что все имена файлов различны.

Выходные данные

В первую строку выведите минимальное количество строк в скрипте Владимира. Далее выведите сам скрипт, каждая строка должна иметь вид «move file_1 file_2», где «file_1» — название существующего в момент исполнения строки файла, а «file_2» — строка из цифр и строчных букв латинского алфавита, длина имени файла должна быть от 1 до 6 символов.

Примеры

входные данные

```
5
01 0
2 1
2extra 0
3 1
99 0
```

выходные данные

```
4
move 3 1
move 01 5
move 2extra 4
move 99 3
```

входные данные

```
2
1 0
2 1
```

выходные данные

```
3
move 1 3
move 2 1
move 3 2
```

входные данные

```
5
1 0
11 1
111 0
1111 1
11111 0
```

выходные данные

```
5
move 1 5
move 11 1
move 1111 2
move 111 4
move 11111 3
```

E - Перенумерация тестов

Сразу выбросим все корректные тесты из рассмотрения. Будем называть тестами типа 1 примеры, а тестами типа 2 — обычные тесты.

Далее назовём свободными все такие позиции i $(i \in [1..n])$, что в оставшемся наборе нет теста с названием, соответствующим номеру i. Если есть хотя бы одна свободная позиция — это хороший случай. Иначе, очевидно, мы не сможем за одну операцию move переместить какой-нибудь тест на своё место, потому что мы сначала должны освободить строку, соответствующую его номеру.

Можно показать, что нам всегда достаточно иметь хотя бы одну свободную позицию. Если она есть, то давайте поставим тест, соответствующий типу этой позиции, на своё место. Очевидно, что такой тест найдётся, иначе бы это означало, что все тесты этого типа стоят на своих местах. Далее может возникнуть два случая — либо мы освободили какую-то другую позицию (когда мы перемещали тест с названием, соответствующим некоторой корректной позиции), либо же мы не освободили ничего (когда мы перемещали тест с названием, не соответствующим какой-то корректной позиции).

Выгоднее в первую очередь перемещать тесты, соответствующие корректным позициям, потому что иначе у нас просто не будут появляться новые свободные позиции для тестов. Очевидно, что когда у нас закончатся тесты, которые имеют

корректные номера, мы больше никогда не получим новой свободной позиции.

Делая так, можно добиться оптимального ответа, потому что будет произведено либо cnt действий, где cnt — количество некорректных тестов в изначальном наборе, либо cnt + 1 действий. cnt + 1 может получиться только тогда, когда нам иногда необходимо совершить дополнительное действие, чтобы освободить какую-либо позицию.

Очевидно, что это количество действий оптимально, потому что если у нас изначально было cnt некорректных тестов, то мы должны сделать хотя бы cnt операций, чтобы переместить каждый тест на своё место. Так как выше мы показали, что если есть свободная позиция, то ей точно соответствует какой-либо некорректный тест, то единственный плохой случай возникает тогда, когда у нас нет свободных позиций. Тогда нам необходимо сделать ещё одну операцию, чтобы освободить позицию.

F. Турне волшебника

ограничение по времени на тест

2 секунды

ограничение по памяти на тест

256 мегабайт

ввод

стандартный ввод

вывод

стандартный вывод

Все жители Берляндии ждут беспрецедентного турне волшебника в голубом вертолете по городам Берляндии!

Известно, что в Берляндии n городов, некоторые пары которых соединены двусторонними дорогами. Каждая пара городов соединена не более чем одной дорогой. Не гарантируется, что дорожная сеть *связна*, то есть возможно, что из какого-то города невозможно добраться до других, двигаясь по дорогам.

Турне волшебника будет состоять из туров. Во время каждого тура:

- волшебник высадится в некотором городе x с вертолета;
- даст представление и бесплатно покажет кино в городе x;
- проедет по дороге в соседний город y;
- даст представление и бесплатно покажет кино в городе y;
- проедет по дороге в соседний с y город z;
- даст представление и бесплатно покажет кино в городе z;
- улетит на вертолете из города z.

Известно, что волшебник не любит путешествовать по дорогам: по каждой дороге он может проехать лишь единожды (независимо от направления). Иными словами для дороги между a и b он может либо проехать один раз из a в b, либо проехать один раз из b в a, либо вообще не проезжать по этой дороге.

Волшебник хочет организовать максимальное количество туров, не нарушив правила изложенные выше. Помогите волшебнику!

Обратите внимание, что волшебник может посещать один город многократно, ограничение на посещения касается исключительно дорог.

Входные данные

В первой строке записаны два целых числа n, m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — количество городов и количество дорог в Берляндии соответственно.

Далее следуют описания дорог, по одному описанию в строке. Каждое описание — это пара целых чисел a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), где a_i и b_i — номера городов, которые соединены i-й дорогой. Гарантируется, что нет двух дорог, соединяющих одну и ту же пару городов. Все дороги — двусторонние. Города пронумерованы от 1 до n.

Возможно, что сеть дорог в Берляндии не является связной.

Выходные данные

В первую строку выведите w — максимальное количество туров волшебника. В следующих w строках выведите сами туры в формате x, y, z — три числа через пробел, номера посещенных за тур городов в порядке посещения.

Примеры

входные данные

```
4 5
1 2
3 2
2 4
3 4
4 1
```

выходные данные

```
2
1 4 2
4 3 2
```

входные данные

```
5 8
5 3
1 2
4 5
5 1
2 5
4 3
1 4
3 2
```

выходные данные

```
4
1 4 5
2 3 4
```

```
1 5 3
5 2 1
```

F - Турне волшебника

Очевидно, что задачу можно решать для каждой компоненты связности независимо. Опишем алгоритм, который позволит набирать ровно $\lfloor \frac{m}{2} \rfloor$ туров, где m — количество ребер в компоненте связности. Рассмотрим произвольное дерево обхода в глубину. Будем в процессе обхода, выходя из вершины v, обрабатывать все ребра, исходящие из этой вершины вниз по дереву, а также ребро, соединяющее v с ее родителем в дереве (если v — не корень). Если ребер, идущих вниз, четное количество, разобьем их на пары. Иначе добавим в разбиение ребро, выходящее из родителя, и не будем его рассматривать для родителя. Такой алгоритм для всех вершин, кроме корня, найдет пару для каждого ребра. Будет не более одного ребра без пары, поэтому ответ будет оптимальным.