The 2nd Universal Cup

Stage 16: Run Twice

December 30-31, 2023

This problem set should contain 11 problems on 24 numbered pages.

Based on

Petrozavodsk Programming Camp

Задача A. Скобочно-палочные последовательности

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Определим множество *правильных скобочно-палочных последовательностей* R рекурсивно. Это множество строк, которые можно получить, применяя только следующие правила:

- $\varepsilon \in R$ (пустая строка)
- $A, B \in R \Rightarrow AB \in R$ (конкатенация)
- $A, B \in R \Rightarrow (A|B) \in R$

Например, последовательности, содержащие две тройки «(|)», выглядят так: «((|)|)», «(|(|))», «(|)(|)».

Придумайте, как сопоставить правильным скобочно-палочным последовательностям заданной длины целые числа, и реализуйте это сопоставление.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение кодирует скобочно-палочные последовательности целыми числами. В первой строке записано слово «encode». Вторая строка содержит целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Каждый тестовый случай занимает две строки: первая содержит целое число n — количество троек «(|)» в последовательности ($1 \leqslant n \leqslant 25$), а вторая — $3n$ символов без пробелов, которые образуют правильную скобочно-палочную последовательность из n троек.

Выведите t строк, по одной на каждый тестовый случай. В i-й строке выведите целое число x_i, которым вы хотите закодировать i-ю последовательность из входных данных ($0 \leqslant x_i \leqslant 2 \cdot 10^{18}$).

Второй запуск

При втором запуске решение декодирует скобочно-палочные последовательности из целых чисел. В первой строке записано слово «decode». Вторая строка содержит целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Каждый тестовый случай занимает две строки: первая содержит целое число n — количество троек «(|)» в последовательности ($1 \leqslant n \leqslant 25$), а вторая — целое число, выведенное в этом тестовом случае при первом запуске.

Выведите t строк, по одной на каждой тестовый случай. В i-й строке выведите скобочно-палочную последовательность из i-го тестового случая.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Как видно, это решение кодирует символы цифрами 1, 2 и 3, после чего выводит в качестве числа получившуюся строку из цифр. Увы, для больших n строки окажутся слишком длинными.

стандартный ввод	*стандартный вывод*
encode 3 1 (|) 4 ((((|)|)|)|) 5 (|(|))((|(|))|)	123 111123232323 121233112123323

стандартный ввод	*стандартный вывод*
decode 3 1 123 4 111123232323 5 121233112123323	(|) ((((|)|)|)|) (|(|))((|(|))|)

Задача B. Чётные и нечётные сочетания

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Сочетанием из n элементов по k будем называть k-элементное подмножество n-элементного множества $\{1, 2, \ldots, n\}$. Чтобы записать сочетание, перечислим его элементы в порядке возрастания. Например, сочетания из 3 элементов по 2 выглядят так: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$.

Будем называть сочетание *чётным*, если количество элементов в нём — чётное число, и *нечётным* в противном случае. Зафиксируем $n > 0$ и рассмотрим два множества: A_n, множество всех чётных сочетаний из n элементов, и B_n, множество всех нечётных сочетаний из n элементов. Можно доказать, что A_n и B_n содержат одинаковое количество сочетаний.

Для каждого $n = 1, 2, \ldots, 50$ задача такова. Постройте любую биекцию (взаимно однозначное соответствие) между множествами A_n и B_n. После этого по данному элементу одного из этих множеств выводите соответствующий ему элемент другого множества.

Протокол взаимодействия

Для проверки того, что вы действительно построили биекцию, в этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске в первой строке записано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Далее следуют их описания.

Каждый тестовый случай задаёт сочетание и занимает две строки. В первой из них записаны через пробел два целых числа n и k ($1 \leqslant n \leqslant 50$, $0 \leqslant k \leqslant n$). Во второй записаны через пробел k целых чисел $a_1, a_2, \ldots, a_k$ — элементы сочетания ($1 \leqslant a_1 < a_2 < \ldots < a_k \leqslant n$). Если $k = 0$, то вторая строка пуста.

В ответ на каждый тестовый случай выведите сочетание из другого множества, соответствующее заданному. Формат сочетания в выводе — точно такой же, как во вводе. У выведенного сочетания n должно совпадать с заданным, а k должно иметь другую чётность. Других ограничений на выбор соответствия нет.

Второй запуск

При втором запуске формат ввода точно такой же, как при первом запуске. Но в каждом тестовом случае дано не исходное сочетание, а то, которое было выведено при первом запуске.

В ответ на каждый тестовый случай, как и при первом запуске, выведите сочетание из другого множества, соответствующее заданному. Формат сочетания в выводе — точно такой же, как во вводе. Поскольку соответствие должно быть взаимно однозначным, выведенное при втором запуске сочетание должно совпадать с тем, которое было дано при первом запуске. Это и будет проверять программа жюри.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	стандартный вывод
6 3 0 2 1 1 3 3 1 2 3 3 1 1 3 1 2 3 1 3	3 3 1 2 3 2 2 1 2 3 0 3 2 2 3 3 2 1 3 3 2 1 2

стандартный ввод	стандартный вывод
6 3 3 1 2 3 2 2 1 2 3 0 3 2 2 3 3 2 1 3 3 2 1 2	3 0 2 1 1 3 3 1 2 3 3 1 1 3 1 2 3 1 3

Задача C. Поиск частей

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мебибайт

Два робота, Карл и Клара — часть секретной сети для передачи сообщений.

Робот Клара получила секретное сообщение. Это сообщение имеет форму чёрно-белого прямоугольника из r строк и c столбцов, содержащего $r \times c$ пикселов. Для каждого пиксела задана яркость — целое число от 0 до 255 (байт): 0 соответствует чёрному цвету, 255 — белому, а числа между этими — различным оттенкам серого.

Клара не знает, есть ли в сообщении какой-то смысл, но выглядит оно как «белый шум»: можно считать, что каждый пиксел равновероятно и независимо от остальных покрашен в один из 256 возможных цветов.

Работа Клары — отвечать на вопросы робота Карла. Каждый вопрос сформулирован как маленький чёрно-белый прямоугольник, который нужно найти в исходном сообщении и выяснить, какие у него координаты.

Но перед тем, как отвечать на вопросы, исходное сообщение нужно уничтожить. А память Клары ограничена — её размер всего 400 кибибайт — и сообщение может в неё не поместиться...

Как действовать Кларе, чтобы, тем не менее, ответить правильно на все вопросы?

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Во вводе и выводе *байты* — это целые числа от 0 до 255 включительно, и записываются они в шестнадцатеричном формате: запись каждого байта состоит ровно из двух символов, а каждый символ — цифра `0`–`9` или заглавная буква `A`–`F`.

Первый запуск

При первом запуске решение получает сообщение и заполняет память Клары. В первой строке записано слово «`message`». Во второй строке записаны через пробел целые числа r и c — количество строк и столбцов в сообщении ($20 \leqslant r, c \leqslant 2000$). Следующие r строк содержат по c байтов, записанных через пробел — само сообщение. Байты выбраны заранее, независимо друг от друга, генератором псевдослучайных чисел, и все значения из диапазона 0–255 равновероятны.

В первой строке выведите целое число m — размер записи в памяти Клары ($0 \leqslant m \leqslant 409\,600$). Во второй строке выведите через пробел m байтов — содержимое памяти Клары.

Второй запуск

При втором запуске решение получает запись в памяти Клары, а затем отвечает на вопросы Карла. В первой строке записано слово «`parts`». Во второй строке записано целое число m — размер записи в памяти Клары ($0 \leqslant m \leqslant 409\,600$). В третьей строке записаны через пробел m байтов — содержимое её памяти. Эти две строки повторяют то, что ваше решение вывело при первом запуске.

В следующей строке записано целое число q — количество вопросов Карла ($1 \leqslant q \leqslant 10\,000$). Далее следуют сами вопросы. Каждый вопрос начинается со строки, содержащей два целых числа h и w — количество строк и столбцов в очередном прямоугольнике ($10 \leqslant h, w \leqslant 20$). Следующие h строк содержат по w байтов, записанных через пробел — содержимое прямоугольника. Гарантируется, что каждый заданный прямоугольник встречается в исходном сообщении ровно один раз. Вопросы зафиксированы заранее и не зависят от результатов первого запуска.

В ответ на каждый вопрос выведите в отдельной строке два целых числа — строку и столбец исходного сообщения, соответствующие левому верхнему углу прямоугольника, заданного в вопросе. Строки нумеруются от 1 до r сверху вниз, а столбцы — от 1 до c слева направо.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Для краткости память показана не полностью. Полную версию примера можно найти в `samples.zip`.

стандартный ввод

```
message
20 24
33 39 73 4A 5A AA E0 86 96 4B 0B 83 A0 FA 82 9B B0 6E DC 03 1C B9 5B 81
86 3E 23 7B C9 38 77 82 7D 62 EA CE A8 DE 85 6C 36 B3 10 EE 85 6A D5 92
14 BD 58 74 20 7B 36 E1 89 B8 6F 4A F4 8F 17 2E 2F 0F 79 DD AA 9F 6F AD
85 21 B6 2F 58 37 87 7B 3F EE D9 7D 9A E6 AA 12 E0 B6 BB 3D 72 BD 34 A5
E5 8A 73 EE 69 BF E0 0D 5C 57 EF 42 7B 91 07 B8 7D A9 40 0D 4B 52 2D BC
25 F7 4F A7 18 4D 76 EB EB 3E AA 3D C2 19 D3 EE 77 BF C1 38 FF C4 07 C0
CD 2B 79 C3 27 A6 C6 DB D3 17 EA CD 74 BC E5 42 36 F8 D2 86 F9 E9 86 AA
F8 37 39 BF 0C B6 2C 9A F5 04 40 BB D8 FD B4 97 2A 9A A6 D1 9E 2A 60 23
F7 CF 3F 25 CB C1 25 08 0F 1F D2 34 C4 61 27 2E 7B E9 00 FD 86 77 E9 AF
7B 44 57 2E 47 F9 CC A0 03 E3 60 C2 DF C1 F5 6C 59 0E 99 64 3D 7D E7 75
EC C9 BE 91 3B DF 1C DC 61 5C 66 1C B3 26 1C 2E 11 0D 19 BD DC 08 1A 90
BF 93 A0 B9 CD 02 DD E6 49 6F 53 E2 2C 34 10 EA 1A 44 B4 49 7E D5 B6 CB
4A E9 C7 3F F1 FF 24 33 5D 8F D4 26 2E C4 FD 81 FB 96 36 51 F1 38 BE 1E
5A C9 B2 3D 06 99 4F 99 3F 45 DB DA 14 BE 53 D7 B2 2D 64 7B 10 74 0E 70
B6 07 1A B4 F3 25 4D EB 3F 68 72 10 3B 56 F2 A7 C4 A4 28 AE 16 D0 13 CC
91 C4 4D 51 04 39 A8 13 3C 1F 00 57 24 2A FD EA FC EB 77 B8 E1 7D DF 0D
92 51 DA 2A CD A1 F3 97 1A 7A EF 41 DF BD 16 4D 05 4B 78 20 B7 68 38 1C
10 D5 DE 39 58 8F F6 22 8B E8 E8 D0 FB 37 31 33 9E C8 FC 79 62 4F BB 96
5F 04 CB 93 16 9F 15 07 96 27 35 09 AB 79 92 37 44 15 14 A1 4E 04 67 5D
C1 C4 8B 1A 77 E1 D2 4D 06 42 07 A3 1A 67 EC F1 B2 08 96 F6 C3 4E 79 E9
```

стандартный вывод

```
484
14 00 18 00 33 39 73 4A 5A AA E0 86 <...> C3 4E 79 E9
```

стандартный ввод

```
parts
484
14 00 18 00 33 39 73 4A 5A AA E0 86 <...> C3 4E 79 E9
2
10 10
39 73 4A 5A AA E0 86 96 4B 0B
3E 23 7B C9 38 77 82 7D 62 EA
BD 58 74 20 7B 36 E1 89 B8 6F
21 B6 2F 58 37 87 7B 3F EE D9
8A 73 EE 69 BF E0 0D 5C 57 EF
F7 4F A7 18 4D 76 EB EB 3E AA
2B 79 C3 27 A6 C6 DB D3 17 EA
37 39 BF 0C B6 2C 9A F5 04 40
CF 3F 25 CB C1 25 08 0F 1F D2
44 57 2E 47 F9 CC A0 03 E3 60
11 20
18 4D 76 EB EB 3E AA 3D C2 19 D3 EE 77 BF C1 38 FF C4 07 C0
27 A6 C6 DB D3 17 EA CD 74 BC E5 42 36 F8 D2 86 F9 E9 86 AA
0C B6 2C 9A F5 04 40 BB D8 FD B4 97 2A 9A A6 D1 9E 2A 60 23
CB C1 25 08 0F 1F D2 34 C4 61 27 2E 7B E9 00 FD 86 77 E9 AF
47 F9 CC A0 03 E3 60 C2 DF C1 F5 6C 59 0E 99 64 3D 7D E7 75
3B DF 1C DC 61 5C 66 1C B3 26 1C 2E 11 0D 19 BD DC 08 1A 90
CD 02 DD E6 49 6F 53 E2 2C 34 10 EA 1A 44 B4 49 7E D5 B6 CB
F1 FF 24 33 5D 8F D4 26 2E C4 FD 81 FB 96 36 51 F1 38 BE 1E
06 99 4F 99 3F 45 DB DA 14 BE 53 D7 B2 2D 64 7B 10 74 0E 70
F3 25 4D EB 3F 68 72 10 3B 56 F2 A7 C4 A4 28 AE 16 D0 13 CC
04 39 A8 13 3C 1F 00 57 24 2A FD EA FC EB 77 B8 E1 7D DF 0D
```

стандартный вывод

```
1 2
6 5
```

Задача D. Сообщение из шума

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Алиса хочет передать Еве по *числовому проводу* сообщение — одно английское слово.

К сожалению, сейчас числовой провод передаёт только шум: случайные целые числа от 0 до $10^9 - 1$ включительно. Алисе известна последовательность из следующих 10 000 чисел, которые будут переданы.

К счастью, у Алисы есть суперспособность: стереть из этой последовательности любое количество элементов на любых позициях. Порядок оставшихся чисел не поменяется.

К сожалению, после этого примерно половина чисел потеряется при передаче: каждое передаваемое число с вероятностью 1/2 исчезнет. Порядок оставшихся чисел опять не поменяется.

Как должны действовать Алиса и Ева, чтобы передать заданное слово?

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. При вводе и выводе числа в одной строке отделяются друг от друга пробелами. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение действует за Алису. В первой строке записано имя «`Alisa`». Вторая строка содержит одно слово из английского словаря, имеющее длину от 2 до 15 букв и состоящее из маленьких букв. В третьей строке записано целое число n — размер последовательности (в этой задаче n всегда равно 10 000). Четвёртая строка содержит n чисел от 0 до $10^9 - 1$ включительно — исходную последовательность. Числа выбраны заранее генератором псевдослучайных чисел, все целые числа из диапазона равновероятны.

Вывести следует те числа, которые Алиса решила **оставить** в последовательности. В первой строке выведите целое число m — количество оставшихся чисел. Во второй строке выведите сами эти числа в том же порядке, в котором они следуют в исходной последовательности.

Второй запуск

При втором запуске решение действует за Еву. В первой строке записано имя «`Eva`». Во второй строке записано целое число k — количество чисел, оставшихся в последовательности. Третья строка содержит k чисел от 0 до $10^9 - 1$ включительно — то, что осталось от последовательности: каждое число, которое Алиса решила оставить в первом запуске, с вероятностью 1/2 оказалось здесь и с вероятностью 1/2 исчезло. Как именно исчезают числа из последовательности — зафиксировано заранее в каждом тесте, то есть, если решения ведут себя одинаково при первом запуске, то они получат одинаковые последовательности при втором запуске.

Выведите одно английское слово — то, которое Алиса должна была передать Еве.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Для краткости последовательности показаны не полностью. Полную версию примера можно найти в `samples.zip`.

стандартный ввод
Alisa spark 10000 833080662 16249270 933346436 811379468 <...> 13286897 459644281
стандартный вывод
3900 933346436 811379468 877083772 408973036 <...> 583178591 13286897

стандартный ввод
Eva 1955 811379468 408973036 585189166 111199534 <...> 226510051 829146141
стандартный вывод
spark

Задача E. Четыре плюс четыре

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

У Королевы Марианны три дочери-принцессы. Марианна хранит свою королевскую печать в сейфе. Сейф защищён *паролем* — это восьмибуквенное английское слово из королевского словаря, которое меняется каждые несколько дней.

Королева регулярно уезжает в отпуск, а принцессы в это время учатся править королевством. Никто из них не знает пароль — но Марианна хочет, чтобы любые две принцессы, придя к согласию, могли открыть сейф. Для этого она пишет на трёх карточках *ключи* — четырёхбуквенные английские слова из королевского словаря. Каждый ключ составлен **из букв пароля**: из восьми букв выбраны четыре и, возможно, переставлены, чтобы получилось слово из словаря. Затем Королева кладёт карточки в шляпу, после чего три принцессы по очереди достают наугад одну из карточек и забирают себе, никому не показывая.

Имея под рукой королевский словарь, придумайте, как договориться Королеве и принцессам, чтобы после того, как Марианна выдала принцессам ключи, любые две из них могли восстановить по своим двум ключам пароль.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки. Словарь один и тот же во всех тестах задачи и в примере: слова взяты из свободно распространяемого списка под названием ENABLE2K. Все слова во вводе и выводе состоят из маленьких английских букв.

Первый запуск

При первом запуске решение действует за Королеву Марианну. В первой строке записано слово «`password`». Во второй строке записано целое число n — количество паролей, для которых нужно выбрать ключи ($1 \leqslant n \leqslant 10\,000$). В каждой из следующих n строк записан один пароль — восьмибуквенное слово из королевского словаря. В качестве пароля могут быть даны только слова, из букв которых можно составить **не менее трёх** различных ключей.

Далее задан королевский словарь, его описание занимает четыре строки и одинаково во всех тестах. В первой из этих строк записано целое число n_8 — количество восьмибуквенных слов в словаре ($n_8 = 28\,558$). Во второй через пробел даны сами восьмибуквенные слова в алфавитном порядке. В третьей строке записано целое число n_4 — количество четырёхбуквенных слов в словаре ($n_4 = 3919$). В четвёртой через пробел даны сами четырёхбуквенные слова в алфавитном порядке.

Выведите n строк: для каждого пароля выведите в любом порядке три ключа, которые напишет на карточках Королева Марианна, разделяя их пробелами. Все ключи должны быть четырёхбуквенными словами из королевского словаря, составленными из букв пароля: из восьми букв выбираются четыре и, возможно, переставляются, чтобы получилось слово из словаря. Никаких других ограничений на выбор ключей нет: например, какому-то паролю можно сопоставить три одинаковых ключа, а другому — тот же самый ключ и ещё два других.

Второй запуск

При втором запуске решение действует за принцесс. В первой строке записано слово «`keys`». Во второй строке записано целое число m — количество пар ключей ($1 \leqslant m \leqslant 60\,000$). В каждой из следующих m строк записана через пробел пара ключей — каждую такую пару программа жюри выбирает из какой-то тройки ключей, выведенной при первом запуске. Пара выбирается так: сначала один из ключей удаляется, а потом оставшиеся два ключа, возможно, меняются местами. Выбор происходит детерминированно: если два решения при первом запуске выдали одинаковые ответы, то и входные данные для них при втором запуске будут одинаковыми.

Далее задан королевский словарь, его описание занимает четыре строки и полностью совпадает с описанием при первом запуске.

Выведите m строк: для каждой пары ключей — правильный пароль.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Для краткости словарь показан не полностью. Полную версию примера можно найти в `samples.zip`.

стандартный ввод	*стандартный вывод*
password 2 password couthier 28558 aardvark aardwolf <...> zyzzyvas 3919 aahs aals abas <...> zori zyme	swap road saws thou thou thou

стандартный ввод	*стандартный вывод*
keys 4 swap road thou thou saws swap road saws 28558 aardvark aardwolf <...> zyzzyvas 3919 aahs aals abas <...> zori zyme	password couthier password password

Отметим, что последнее восьмибуквенное слово в словаре, «`zyzzyvas`» — пример слова, которое не может встретиться в качестве пароля: из его букв можно составить только один ключ, «`yays`». Напомним, что паролями могут быть только слова, из букв которых можно составить хотя бы три различных ключа. В словаре есть 70 восьмибуквенных слов, для которых это условие неверно.

Задача F. Отметка на графе

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Дан граф из n вершин и m рёбер — неориентированный, без петель и кратных рёбер. Известно, что этот граф получен одним из двух способов.

- Граф сгенерирован случайно: процесс начинается с графа без рёбер, а далее в него m раз добавляется случайное ребро, равновероятно выбранное из тех, которых ещё нет в графе.

  В этом случае следует оставить на графе отметку. Для этого можно от 0 до 5 раз выбрать пару вершин и поменять состояние ребра между ними: добавить ребро, если его не было, или удалить, если оно было.

- Граф содержит отметку, то есть получен из случайного графа процедурой, описанной выше — но после этого вершины случайным образом перенумерованы, и рёбра тоже даны в случайном порядке. Вершины каждого ребра также могут быть даны в любом порядке.

  В этом случае больше ничего не нужно делать.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. При вводе и выводе числа в одной строке отделяются друг от друга пробелами. В конце каждой строки входных данных следует символ перевода строки.

При каждом запуске решение получает граф. В первой строке записаны два целых числа n и m — количество вершин и рёбер в графе. Каждая из следующих m строк содержит два целых числа u и v, означающих, что в графе есть ребро между вершинами u и v ($1 \leqslant u, v \leqslant n$, $u \neq v$, все неориентированные рёбра различны).

Первый запуск

При первом запуске граф сгенерирован заранее случайно, как указано в условии ($n = 1000$, $2000 \leqslant m \leqslant 5000$). В первой строке следует вывести слово «`mark`», а во второй строке — число k, обозначающее количество операций с рёбрами ($0 \leqslant k \leqslant 5$). Каждая из следующих k строк должна содержать два целых числа u и v, означающих, что следует поменять состояние ребра между вершинами u и v ($1 \leqslant u, v \leqslant n$, $u \neq v$).

Второй запуск

При втором запуске дан тот граф, который получился после всех действий из первого запуска — но вершины случайным образом перенумерованы, рёбра даны в случайном порядке, и вершины каждого ребра также даны в случайном порядке. Как именно происходят все перемешивания — зафиксировано заранее в каждом тесте, то есть, если решения ведут себя одинаково при первом запуске, то они получат одинаковые входные данные при втором запуске. В этом случае в первой строке следует вывести слово «`ok`».

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Для краткости графы показаны не полностью. Полную версию примера можно найти в `samples.zip`.

стандартный ввод	*стандартный вывод*
1000 3560 603 151 415 20 102 569 895 552 <...> 224 267 651 506	mark 3 763 968 572 286 453 139

стандартный ввод	*стандартный вывод*
1000 3561 192 768 693 994 786 238 351 329 <...> 100 66 54 819	ok

Задача G. Передача дежурства

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Аня сегодня дежурит в лаборатории. Там у дежурного на пульте есть — без преувеличения — целый миллион переключателей! Переключатели пронумерованы целыми числами от 1 до 10^6, и каждый переключатель отвечает за прибор с таким же номером. По переключателям непонятно, включены их приборы или выключены, но известно, что нажатие на переключатель меняет состояние со включённого на выключенное и наоборот.

Утром, когда Аня приходит на дежурство, все приборы выключены. Затем приходят другие сотрудники и время от времени нажимают на переключатели.

Чтобы оптимизировать потребление энергии, после каждого переключения дежурный должен различать следующие классы состояний:

- все приборы выключены,
- включён ровно один прибор — нужно знать, какой именно,
- включено два или больше приборов.

Аня, конечно, справится с этим заданием. Но после неё будет дежурить её друг Андрей. А при передаче дежурства Аня может оставить Андрею лишь короткую записку. После прочтения Андрей будет дежурить точно так же, как Аня: другие сотрудники будут нажимать на переключатели, а ему нужно будет знать, в каком из классов состояний находится лаборатория.

Помогите ребятам придумать, что написать в записке, чтобы не только Аня, но и Андрей после каждого переключения обладал всей нужной информацией.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В каждом тесте все нажатия при обоих запусках зафиксированы заранее. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение действует за Аню. В первой строке записано слово «`start`». Во второй строке записано целое число n — количество переключений ($1 \leqslant n \leqslant 100\,000$). Каждая из следующих n строк содержит одно целое число — номер прибора (от 1 до 10^6), на переключатель которого нажал сотрудник.

В ответ на каждое переключение выведите строку с одним числом:

- `0`, если все приборы выключены,
- номер включённого прибора, если включён ровно один прибор,
- `-1`, если включено два или больше приборов.

После всех ответов выведите в отдельной строке записку, которую Аня оставит Андрею. Записка должна иметь длину от 0 до 1000 символов и состоять из символов с ASCII-кодами от 32 до 126. Никаких других ограничений на содержимое записки нет.

Второй запуск

При втором запуске решение действует за Андрея. В первой строке записано слово «`resume`». Во второй строке дана записка — ровно то, что решение вывело при первом запуске. В третьей строке записано целое число m — количество переключений ($1 \leqslant m \leqslant 100\,000$). Каждая из следующих m строк содержит одно число — номер прибора (от 1 до 10^6), на переключатель которого нажал сотрудник.

В ответ на каждое переключение выведите строку с одним числом — по тем же правилам, что и при первом запуске.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*
start 5 10 14 10 12 10	10 -1 14 -1 -1 3 10 12 14

стандартный ввод	*стандартный вывод*
resume 3 10 12 14 6 14 277 12 10 277 12	-1 -1 -1 277 0 12

Задача H. Жадная сортировка

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Петя — робот-сортировщик. В его памяти хранится массив длины n (от 1 до 100), элементы которого — попарно различные целые числа. Позиции в массиве пронумерованы слева направо целыми числами от 1 до n.

Нина — оператор робота. Нина хочет отсортировать этот массив: сделать так, чтобы для любых двух различных позиций элемент на левой был меньше элемента на правой. Единственная доступная для этого команда — сравнить элементы на двух позициях, i и j. Если элемент на левой из этих позиций (это может быть как i, так и j) больше элемента на правой, то Петя меняет их местами, после чего показывает на экране число `1`. В противном случае Петя ничего не меняет в массиве и просто показывает на экране число `0`.

К сожалению, блок питания у Пети барахлит, поэтому иногда робот выключается. Выглядит это так: в ответ на очередную команду Петя вместо того, чтобы выполнить её, просто показывает на экране число `-1`, и после этого не реагирует на следующие команды.

Чтобы Петя снова заработал, Нина разбирает его и собирает заново. Массив от этого не меняется. К сожалению, за время ремонта Нина успевает забыть, какие команды она уже дала роботу. Далее Петя работает, пока не кончится заряд его батареи, а после этого выключается опять.

Заряда Петиной батареи хватит на выполнение 1500 команд. Петя выключается ровно два раза: после выполнения x-й команды и после выполнения 1500-й команды ($0 < x < 1500$, число x не известно Нине). Зная всё это, помогите Нине сделать так, чтобы после второго выключения массив в Петиной памяти был отсортирован.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. Ваше решение действует за Нину, а программа жюри — за робота Петю. В конце каждой строки входных данных следует символ перевода строки.

Это интерактивная задача. Не забывайте очищать буфер вывода сразу после печати каждой команды!

Далее описано взаимодействие вашего решения и программы жюри — оно одинаковое в обоих запусках.

В первой строке входных данных записано целое число n — размер массива ($1 \leqslant n \leqslant 100$). Далее решение даёт команды, а программа жюри их выполняет и выводит ответы.

Чтобы дать роботу команду «сравнить элементы на позициях i и j», выведите строку вида «i j», где $1 \leqslant i, j \leqslant n$. В ответ вы получите строку с одним числом на ней:

- `1` означает, что Петя поменял местами элементы массива на позициях i и j, потому что левый элемент был больше правого;
- `0` означает, что Петя ничего не поменял, потому что левый элемент не был больше правого (то есть либо левый был меньше правого, либо $i = j$);
- `-1` означает, что Петя вместо выполнения этой команды выключился.

В последнем случае решение должно завершить работу. В остальных случаях можно дать следующую команду.

Если вы хотите закончить взаимодействие до того, как Петя выключится, вместо очередной команды выведите строку «`-1 -1`». После этого решение должно завершить работу.

В каждом тесте массив перед первым запуском зафиксирован заранее, а перед вторым запуском он в том состоянии, в котором оказался в конце первого запуска. Также в каждом тесте зафиксировано число команд x ($0 < x < 1500$), после которого Петя выключается в первый раз. Во втором запуске робот выключится после $1500 - x$ команд, даже если взаимодействие в первом запуске закончилось до того, как он выключился в первый раз.

Пример

Далее показаны два запуска какого-то решения на первом тесте. Пустые строки добавлены лишь для удобства чтения — при реальном запуске их не будет.

стандартный ввод	стандартный вывод	массив
5 1 1 0 1 1 0 1 0 -1	 1 5 1 2 2 3 3 4 4 5 2 1 3 2 4 3 1 2	4 2 5 1 3 3 2 5 1 4 2 3 5 1 4 2 3 5 1 4 2 3 1 5 4 2 3 1 4 5 2 3 1 4 5 2 1 3 4 5 2 1 3 4 5

стандартный ввод	стандартный вывод	массив
5 1 0 0	 1 2 2 3 1 2 -1 -1	2 1 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

Задача I. Телепатия

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Братья Флим и Флэм показывают фокус, который они называют «телепатия».

Сначала Дискорд, который выступает ведущим, генерирует две случайных двоичных строки a и b. Каждая строка состоит из $n = 10^6$ цифр, и каждая цифра равна нулю или единице равновероятно и независимо от остальных цифр. Строка a достаётся Флиму, а b — Флэму. Каждый видит только свою строку, и не знает строку брата.

После этого каждый из братьев выбирает $k = 10^5$ различных позиций в строке — только не в своей, а в строке брата, которую он не знает!

Наконец, Дискорд выписывает слева направо цифры из a на позициях, выбранных Флэмом, а под ними, также слева направо — цифры из b на позициях, выбранных Флимом. После этого зрители считают, сколько раз цифра из a совпадает с выписанной под ней цифрой из b. Для «доказательства» того, что телепатия работает, совпадений должно быть **больше, чем две трети**, то есть хотя бы 66 667.

Помогите Флиму и Флэму договориться о том, как выбирать позиции в строках друг у друга, зная только свою строку, так, чтобы «доказать», что телепатия работает.

Рассмотрим небольшой пример.

- Пусть для краткости $n = 20$ и $k = 5$.
- Пусть строка a равна `00101011011110111001`.
- Пусть строка b равна `11000111101000011010`.
- Флим видит строку a и выбирает позиции $2, 3, 5, 7, 11$ в строке b.
- Флэм видит строку b и выбирает позиции $1, 4, 9, 16, 20$ в строке a.
- Дискорд выписывает $a_1 = 0$, $a_4 = 0$, $a_9 = 0$, $a_{16} = 1$, $a_{20} = 1$.
- А под ними пишет $b_2 = 1$, $b_3 = 0$, $b_5 = 0$, $b_7 = 1$, $b_{11} = 1$.
- Из пяти пар цифры совпадают во всех, кроме первой ($a_1 = 0$, но $b_2 = 1$).
- Значит, Флим и Флэм добились $4/5$ совпадений.
- Доля совпадений больше, чем $2/3$, то есть братья «доказали», что телепатия работает!

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза: в первом запуске за Флима, а во втором — за Флэма. Программа жюри действует за Дискорда. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске решение действует за Флима. В первой строке записано имя «`Flim`». Во второй строке записаны через пробел два целых числа n и k — длина строки и количество позиций, которые нужно выбрать (во всех тестах задачи $n = 10^6$ и $k = 10^5$). В третьей строке даны n двоичных цифр без пробелов — строка Флима a.

Выведите через пробел k целых чисел $1 \leqslant p_1 < p_2 < \ldots < p_k \leqslant n$ — выбранные позиции в строке b.

Второй запуск

При втором запуске решение действует за Флэма. В первой строке записано имя «`Flam`». Во второй строке записаны через пробел два целых числа n и k — длина строки и количество позиций, которые нужно выбрать (здесь снова $n = 10^6$ и $k = 10^5$). В третьей строке даны n двоичных цифр без пробелов — строка Флэма b.

Выведите через пробел k целых чисел $1 \leqslant q_1 < q_2 < \ldots < q_k \leqslant n$ — выбранные позиции в строке a.

После второго запуска программа жюри сравнивает цифры в k парах: сначала a_{q_1} с b_{p_1}, затем a_{q_2} с b_{p_2}, и так далее. Если равенство получилось более чем в $\frac{2}{3}k$ парах, решение получает вердикт `OK` на тесте, иначе — `Wrong Answer`.

В задаче один пример, доступный для скачивания, и ещё 50 секретных тестов. Все двоичные строки сгенерированы заранее генератором псевдослучайных чисел: каждая цифра каждой строки равновероятно и независимо от других цифр оказалась либо нулём, либо единицей.

Пример

Далее показаны два запуска какого-то решения на первом тесте. Для краткости строки и ответы показаны не полностью. Полную версию примера можно найти в `samples.zip`. В примере получилось 66 859 совпадений.

стандартный ввод	*стандартный вывод*
Flim 1000000 100000 110111111110<...>11010111	3 14 25 <...> 999979 999990

стандартный ввод	*стандартный вывод*
Flam 1000000 100000 000000110100<...>10011111	7 16 21 <...> 999977 999992

Задача J. Тетра-паззл

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Тетра-паззл — походовая игра для одного игрока, напоминающая Тетрис. Игра происходит на квадратном поле размера 5×5 клеток. Изначально поле пустое.

Каждый ход состоит в том, чтобы поставить на поле тетрамино заданного вида — фигурку из четырёх клеток, связную по стороне. Каждую фигурку можно поворачивать, переворачивать и перемещать, и в итоге следует положить так, чтобы она заняла четыре свободные клетки поля. После этого все строки и все столбцы поля, в которых все клетки заняты, одновременно очищаются: все их клетки становятся свободными. Игрок проигрывает, если не может сделать ход.

Всего есть пять видов тетрамино. Каждый вид задаётся своим названием — заглавной английской буквой, напоминающей его форму:

```
.....   .....   .....   .....   .....
.*...   .*...   .**..   .***.   .**..
.*...   .*...   .**..   ..*..   ..**.
.*...   .**..   .....   .....   .....
.*...   .....   .....   .....   .....

  I       L       O       T       Z
```

В игре есть два режима: базовый режим и режим подготовки. В базовом режиме игрок получает n фигурок по одной, и должен положить указанную фигурку, прежде чем узнает фигурку для следующего хода. Цель — успешно сделать все n ходов.

В режиме подготовки игрок планирует свою будущую партию в базовом режиме. При подготовке игрок получает сразу n случайно сгенерированных пар фигурок: по две на выбор для первого, второго, ..., n-го хода. Из каждой пары игрок должен выбрать одну фигурку, которая и встретится на этом ходу при игре в базовом режиме.

Ваша задача — успешно сыграть в эту игру: сначала в режиме подготовки, а затем в базовом режиме с выбранными фигурками.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

При втором запуске эта задача — интерактивная. Не забывайте очищать буфер вывода сразу после каждого выведенного хода!

Первый запуск

При первом запуске игра идёт в режиме подготовки. В первой строке записано слово «`prepare`». Вторая строка содержит целое число n — количество ходов ($1 \leqslant n \leqslant 1000$). В третьей строке записаны через пробел n пар фигурок — по два варианта для первого, второго, ..., n-го хода. Каждая пара задаётся двумя заглавными буквами из набора «`ILOTZ`», соответствующими видам фигурок в паре. Буквы в паре различны и могут следовать в любом порядке. В каждом тесте каждая пара выбрана заранее случайно, равновероятно из всех возможных вариантов и независимо от других пар.

Выведите строку из n букв без пробелов: для каждого хода — какой из двух вариантов вы выбрали.

Второй запуск

При втором запуске игра идёт в базовом режиме. В первой строке записано слово «`play`». Вторая строка содержит целое число n — количество ходов, такое же, как при первом запуске ($1 \leqslant n \leqslant 1000$). Далее следуют n ходов.

Каждый ход начинается с того, что решение получает в отдельной строке букву — вид фигурки, выбранный для этого хода при первом запуске. В ответ следует вывести поле с поставленной фигуркой, но до удаления заполненных строк и столбцов. Поле занимает 5 строк, в каждой из которых

5 символов. Символ «.» (точка, ASCII-код 46) соответствует пустой клетке, символ «#» (решётка, ASCII-код 35) — клетке, занятой на более раннем ходу, а символ «*» (звёздочка, ASCII-код 42) — клетке только что поставленной фигурки.

Если указанный выше формат соблюдён, но ход некорректен, решение получит вердикт «Wrong Answer» на тесте.

Если ход корректен, то все заполненные строки и столбцы очищаются, а оставшиеся от фигурок клетки в дальнейшем следует обозначать символом «#». Далее происходит следующий ход.

Если все n ходов уже произошли, решение получит вердикт «OK» на тесте.

Примеры

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Пустые строки добавлены лишь для удобства чтения — при реальном запуске их не будет.

стандартный ввод	*стандартный вывод*
prepare 6 TO LO ZI LI LT LT	OLZITT

стандартный ввод	*стандартный вывод*
play 6 O L Z I T T	..... **... **... ....*** ##..* ##... ....### ##**# ##.** ....### ..*.. ##*## ..*.. ..*.. ***## .*... .... *#... **... *.... .....

Задача K. Триекция

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Существует много типов комбинаторных объектов, для которых количество различных объектов заданного размера n — это число Каталана $C_n = \frac{(2n)!}{n!(n+1)!}$. Первые несколько чисел Каталана таковы: $C_0 = 1$, $C_1 = 1$, $C_2 = 2$, $C_3 = 5$, $C_4 = 14$, $C_5 = 42$, ... Рассмотрим три таких типа объектов:

- Косые полиомино с периметром $2n + 2$. Это фигуры из клеток на клетчатом поле размера $h \times w$, где $h + w = n + 1$, связные по стороне. Левая нижняя и правая верхняя клетки заняты — они принадлежат полиомино. Остальные клетки могут быть как заняты, так и свободны, но выполняются такие условия:
  - в каждой строке и в каждом столбце занятые клетки образуют непрерывный отрезок;
  - каждый отрезок в столбце начинается выше или там же, где отрезок в соседнем слева столбце;
  - каждый отрезок в столбце заканчивается выше или там же, где отрезок в соседнем слева столбце;
  - каждый отрезок в строке начинается правее или там же, где отрезок в соседней снизу строке.
  - каждый отрезок в строке заканчивается правее или там же, где отрезок в соседней снизу строке.

  Вот все косые полиомино размера $n = 4$:

```
poly      poly      poly      poly      poly      poly      poly
4 1       3 2       3 2       3 2       3 2       3 2       3 2
#         .#        .#        ##        ##        ##        .#
#         .#        ##        #.        ##        ##        ##
#         ##        #.        #.        #.        ##        ##
#
```

```
          poly      poly      poly      poly      poly      poly
poly      2 3       2 3       2 3       2 3       2 3       2 3
1 4       ..#       .##       ###       ###       ###       .##
####      ###       ##.       ##.       #..       ###       ###
```

- 321-избегающие перестановки длины n. Это перестановки $p_1, p_2, \ldots, p_n$ из элементов $1, 2, \ldots, n$, в которых нет такой тройки позиций $i < j < k$, на которых $p_i > p_j > p_k$.

  Вот все 321-избегающие перестановки размера $n = 4$:

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

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

- Триангуляции выпуклого $(n + 2)$-угольника. Пронумеруем вершины многоугольника целыми числами от 1 до $n + 2$ в порядке обхода. В каждом из n треугольников упорядочим номера вершин по возрастанию. Сами треугольники упорядочим по возрастанию как тройки чисел.

  Вот все триангуляции размера $n = 4$:

```
triang      triang      triang      triang      triang      triang      triang
1 2 5       1 2 3       1 2 3       1 2 6       1 2 4       1 2 3       1 2 4
1 5 6       1 3 4       1 3 6       2 3 4       1 4 5       1 3 6       1 4 6
2 3 5       1 4 6       3 4 5       2 4 5       1 5 6       3 4 6       2 3 4
3 4 5       4 5 6       3 5 6       2 5 6       2 3 4       4 5 6       4 5 6

triang      triang      triang      triang      triang      triang      triang
1 2 3       1 2 6       1 2 3       1 2 6       1 2 6       1 2 6       1 2 5
1 3 5       2 3 6       1 3 4       2 3 4       2 3 6       2 3 5       1 5 6
1 5 6       3 4 5       1 4 5       2 4 6       3 4 6       2 5 6       2 3 4
3 4 5       3 5 6       1 5 6       4 5 6       4 5 6       3 4 5       2 4 5
```

Зафиксируем число n и рассмотрим три множества:

- A_n — множество косых полиомино размера n,
- B_n — множество 321-избегающих перестановок размера n,
- C_n — множество триангуляций размера n.

Постройте *триекцию* между этими множествами. Триекция — это почти как биекция, только множества не два, а три. А именно, придумайте функцию $f_n : A_n \cup B_n \cup C_n \to A_n \cup B_n \cup C_n$, для которой значение от любого объекта одного из трёх типов — это обязательно объект **другого** типа, и при этом $f_n(f_n(x)) = x$ для любого x (то есть f_n — функция, обратная самой себе).

После этого по данным объектам трёх типов выведите результат триекции.

Дополнительное условие: заданное число n **не может** иметь вид $2^k - 1$.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

Первый запуск

При первом запуске в первой строке записаны через пробел два целых числа n и t — размер, одинаковый для всех объектов, и количество объектов ($2 \leqslant n \leqslant 35$, $1 \leqslant t \leqslant 1000$, число n не может иметь вид $2^k - 1$). Далее даны t объектов. Описание объекта начинается со строки с названием его типа, в следующих строках следует описание объекта, зависящее от типа. Подробное описание типов и формат записи объектов показаны в условии выше и в примере ниже.

В первой строке выведите через пробел числа n и t (эта строка попала в формат для удобства — чтобы вывод программы можно было дать в качестве ввода без изменений). Далее выведите t объектов — результаты триекции от t заданных объектов. Формат вывода объектов — такой же, как во вводе.

Второй запуск

При втором запуске формат ввода и вывода точно такой же, как при первом запуске. Но даны не изначальные t объектов, а те, которые решение вывело при первом запуске, причём **переставленные случайным образом**.

Изначальные t объектов зафиксированы заранее в каждом тесте. Случайная перестановка, которая применяется между первым и вторым запуском, также зафиксирована заранее.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте. Заметим, что вывод во втором запуске — это переставленный ввод в первом запуске.

стандартный ввод	*стандартный вывод*
5 4 poly 4 2 .# ## ## #. perm 4 1 5 2 3 triang 1 2 4 1 4 5 1 5 7 2 3 4 5 6 7 perm 2 1 3 5 4	5 4 perm 3 1 4 2 5 poly 4 2 ## ## ## #. poly 3 3 .## ### ##. triang 1 2 3 1 3 7 3 4 7 4 5 7 5 6 7

стандартный ввод	*стандартный вывод*
5 4 poly 4 2 ## ## ## #. triang 1 2 3 1 3 7 3 4 7 4 5 7 5 6 7 poly 3 3 .## ### ##. perm 3 1 4 2 5	5 4 perm 4 1 5 2 3 perm 2 1 3 5 4 triang 1 2 4 1 4 5 1 5 7 2 3 4 5 6 7 poly 4 2 .# ## ## #.