Задания первого тура

**Олимпиада по веб-программированию 2018 г. «1С-Битрикс»**

### А. Казино “три острых топора”

**Имя входного файла:** input.txt или стандартный поток ввода  
**Имя выходного файла:** output.txt или стандартный поток вывода

Во время просмотра любимого сериала, фрилансер Костя увидел рекламу казино. Интересная задача сама пришла в голову программисту. А что если я разработаю своё казино с играми и яркими фоновыми картинками. Интересной задачей будет подсчет выигрыша моих посетителей, в зависимости от их ставок и результатов игр.

**Входные данные**

В первой строке целое положительное число n (1 ≤ n ≤ 104) — количество ставок.

В следующих n-строках содержатся ставки, по одной в строке. Каждая ставка задается тремя значениями ai, si, ri (1 ≤ ai ≤ 105, 1 ≤ si ≤ 1000), где ai — это идентификатор игры, si — сумма ставки на исход игры, ri — результат игры, может принимать значения: L - левая команда, R - правая команда, D - ничья.

Далее идет целое положительное число m (n ≤ m ≤ 105) — количество игр.

В следующих m-строках содержатся игры, по одной в строке. Каждая игра задается пятью значениями bj, cj, dj, kj, tj (1 ≤ bj ≤ 105, 1 ≤ cj, dj, kj≤ 100), где bi — это идентификатор игры, cj — вещественное число, коэффициент на победу левой команды, dj — вещественное число, коэффициент на победу правой команды, kj — вещественное число, коэффициент на ничью, tj — результат игры, может принимать значения: L - левая команда, R - правая команда, D - ничья.

**Выходные данные**

Одно число - итоговый баланс игрока (насколько больше стало денег, чем до того, как он поставил). Например, если единственная ставка на 100 рублей не выиграла, в ответе требуется вывести -100. Все коэффициенты больше 1.

**Пример**

|  |  |
| --- | --- |
| Входные данные | Результат работы |
| 3 1 100 L 2 100 R 3 100 D 3 1 1.2 1.3 1.4 L 2 1.2 1.3 1.4 L 3 1.2 1.3 1.4 L | -180 |

### B. Размер имеет значение

**Имя входного файла:** input.txt или стандартный поток ввода  
**Имя выходного файла:** output.txt или стандартный поток вывода

Фрилансер Костя взялся за интересный заказ на бирже: разработка системы анализа логов веб-сервера, который работает в сети с адресацией IPv6. Основная загвоздка в том, что в логах веб-сервера все ip-адреса хранятся в сокращенном виде, а заказчик хочет получить отчет с полными ip-адресами.

Для сокращения применяется два правила:

1. В каждом блоке удаляются ведущие нули. Однако каждый из блоков должен содержать хотя бы один символ.

«2001:0db0:0000:123a:0000:0000:0000:0030»  →  «2001:db0:0:123a:0:0:0:30».

1. Самая длинная последовательность нулевых блоков сокращаются до ::. Замена производится только один раз; группа из одного нулевого блока не заменяется.

Необходимо запрограммировать преобразователь IPv6 адресов из сокращенного в полный вид.

**Входные данные**

В строках указываются сокращенные IPv6-адреса. Количество адресов не превышает 1000.

**Выходные данные**

Для каждого сокращенного ip выведите его полный вид, по одному адресу в строке.

**Пример**

|  |  |
| --- | --- |
| Входные данные | Результат работы |
| c32g:t2:0:124:1:: c32g::124:1:0:1234:ff0 c32g::1:0:1234:ff0 c32g:t2:0:124:1:: :: | c32g:00t2:0000:0124:0001:0000:0000:0000 c32g:0000:0000:0124:0001:0000:1234:0ff0 c32g:0000:0000:0000:0001:0000:1234:0ff0 c32g:00t2:0000:0124:0001:0000:0000:0000 0000:0000:0000:0000:0000:0000:0000:0000 |

### C. Семь раз отмерь, один раз отрежь

**Имя входного файла:** input.txt или стандартный поток ввода  
**Имя выходного файла:** output.txt или стандартный поток вывода

Костя одобрял проверку данных перед их использованием и положительно отзывался о коллегах, которые не забывали о валидации. Тех кто забывал проверять данные он тоже любил, т.к. за деньги исправлял их ошибки.

В этот день он проснулся в 11, работы не было, а руки чесались. Костя хотел сделать свой универсальный механизм валидации, который можно будет использовать в разных проектах. Время которое предстоит потратить на эту разработку, скрипя сердцем, он посчитал инвестицией в своё светлое будущее.

Костя мучительно вспоминал сайты, формы и поля, которые он делал. Спустя час он составил список необходимых правил валидации:

1. Строка длиной b (n ≤ b ≤ m), тип валидации S. n и m параметры валидации.
2. Целое число a (n ≤ a ≤ m), тип валидации N. n и m параметры валидации.
3. Номер телефона по маске +7 (999) 999-99-99, тип валидации P.
4. Дата и время по формату d.m.Y H:i, тип валидации D.
5. Адрес электронной почты, тип валидации E. Имя от 4 до 30 символов, можно использовать заглавные и строчные латинские буквы, цифры и нижнее подчеркивание. Адрес не может начинаться с нижнего подчеркивания. Дальше следует символ @, за ним домен - от 2 до 30 заглавных или строчных латинских букв, точка и домен верхнего уровня - от 2 до 10 строчных латинских букв.

**Входные данные**

В строках указываются значения и правила для валидации. Количество значений для валидации не превышает 1000. Каждая строка содержит в себе значение в треугольных скобках, правило для валидации (указывается символом типа валидации) и параметры если они есть. Гарантируется, что во входной строке отсутствуют треугольные скобки.

**Выходные данные**

По порядку в соответствующей строке вывести OK если валидация пройдена и FAIL в противном случае.

**Пример**

|  |  |
| --- | --- |
| Входные данные | Результат работы |
| <asd ns> S 1 10 <90> N 11 83 <+7 (999) 999-99-99> P <17.01.2018 13:38> D <test@test.com> E | OK FAIL OK OK OK |

### D. Маршрут построен

**Имя входного файла:** input.txt или стандартный поток ввода  
**Имя выходного файла:** output.txt или стандартный поток вывода

В свободное от заказов и работы время, Костя изучал алгоритмы маршрутизации запросов в интернете. Он уже был знаком с не адаптивными алгоритмами и пришло время переходить к адаптивным. В качестве задачи он выбрал для себя программирование алгоритма для определения быстрейшего маршрута в сети.

**Входные данные**

В первой строке содержатся целые числа n и m (2 ≤ n ≤ 1000, 0 ≤ m ≤ 10000), где n — количество узлов в сети, а m — количество соединений между узлами.

В следующих m строках содержатся описания соединений между узлами, по одному в строке. Каждое соединение задается тремя числами ai, bi, wi (0 ≤ ai, bi < n, 1 ≤ wi ≤ 106) где ai, bi — это узлы сети, wi — время передачи сигнала. Узлы нумеруются с нуля.

Далее идет целое положительное число k (n ≤ k ≤ 1000) — количество запросов.

В следующих k строках содержатся описания запросов, по одному в строке. Каждый запрос задается тремя числами cj, dj, rj (1 ≤ cj, dj ≤ n), где cj, dj — это узлы сети, rj — запрос, который делится на 3 типа:

1. ? — запрос на получение времени передачи сигнала между узлами сети cj и dj;
2. -1 — удаление соединения между узлами сети cj и dj;
3. целое положительное число — время передачи сигнала между узлами cj и dj изменилось на число из запроса.

**Выходные данные**

На каждый запрос "?" в порядке поступления вывести время передачи сигнала между указанными узлами на тот момент. По одному ответу в строке. Если пакет не сможет дойти от одного узла до другого, вывести -1.

**Пример**

|  |  |
| --- | --- |
| Входные данные | Результат работы |
| 3 3 0 1 100 0 2 100 1 2 100 4 0 1 ? 0 1 -1 0 2 300 0 1 ? | 100 400 |

## 