1. Анализ журнала

1 секунда 30 очков

В журнале компьютерной системы записи вида "*время событие ID-сессии ID-пользователя*", где *время* — время события, а *событие* — одно из `S` (пользователь вошёл в систему), `V` (пользователь вышёл из системы) или `R` (пользователь запустил перезагрузку системы, в результате чего все пользователи одновременно вышли из системы).

Идентификаторы сессий могут повторяться в журнале, но ни в один момент времени не бывает несколько активных сессий с одинаковым идентификатором. У одного пользователя может быть несколько параллельных сессий, но даже в таком случае у каждой сессии свой уникальный идентификатор.

Написать программу, которая на основании этого журнала найдёт общую долготу периодов, во время которых пользователь использовал систему от начала журнала до времени последней записи в журнале, учитывая время параллельных сессий лишь один раз. Можно считать, что журнал начинается в момент, когда ни один пользователь не находится в системе.

Входные данные. На первой строке текстового файла `loge.sis` стоит количество записей журнала N ($1 \leqslant N \leqslant 10\,000$), а на каждой из следующих N строк — одна запись в описанном выше формате. Время, а также идентификаторы сессий и пользователей записаны как целые числа $0 \ldots 10^9$. Записи идут в хронологическом порядке.

Выходные данные. В текстовый файл `loge.val` вывести ровно одну строку для каждого пользователя, представленного во входных данных. На каждую строку вывести два разделённых пробелом целых числа: идентификатор пользователя и общую долготу периодов, во время которых этот пользователь использовал систему. Порядок строк в файле не важен.

Пример.

`loge.sis`
```
3
10 S 1 1
20 S 2 2
40 V 1 1
```

`loge.val`
```
1 30
2 20
```

Пример.

`loge.sis`
```
4
10 S 1 1
20 S 2 1
30 V 2 1
40 V 1 1
```

`loge.val`
```
1 30
```

2. Инициалы

1 секунда 30 очков

Друзья решили участвовать в лыжном марафоне. Для поднятия мотивации к тренировке они хотят нарисовать на доске большую таблицу, куда каждый сможет записать свои тренировочные километры. В таблице будет один столбец для каждого человека. Так как на доске места мало, а друзей много, нужно будет как-то сокращать имена, чтобы все поместились. Есть идея использовать для каждого первые буквы имени и фамилии, и ещё как можно меньше букв из фамилии так, чтобы результат всё-таки был уникальным.

Написать программу, которая прочитает имена друзей и выведет подходящие заглавия для столбцов таблицы.

Входные данные. На первой строке текстового файла `init.sis` стоит количество друзей N ($1 \leqslant N \leqslant 10\,000$), и на каждой из следующих N строк стоят имя и фамилия одного из друзей. Можно считать, что все имена и фамилии состоят только из латинских букв, и длина ни одного имени или фамилии не превышает 30 букв. Также можно считать, что первая буква имени и полная фамилия у каждого друга уникальны.

Выходные данные. В текстовый файл `init.val` вывести ровно N строк. На каждую строку вывести сокращение одного имени в порядке их появления во входном файле. Проверка уникальности сокращений не различает большие и маленькие буквы.

Пример.

`init.sis`	`init.val`
`3`	`MMa`
`Mari Maasikas`	`MMu`
`Mati Murakas`	`MK`
`Mart Kivikas`	

3. Объединение телекоммуникаций

1 секунда 40 очков

В одной стране есть несколько компаний, предлагающих мобильную связь. У каждой компании есть свои кабельные линии, которые объединяют все базовые станции этой компании в одну сеть. В соответствии с решением о слиянии этих компаний, они хотят объединить свои сети в одну. Для этого они уже получили от кабельных фирм набор предложений о прокладывании новых кабельных линий между различными базовыми станциями.

Написать программу, которая поможет мобильным компаниям выбрать комплект предложений с минимальной стоимостью, с помощью которого возможно объединить их сети в одну.

Входные данные. На первой строке текстового файла `side.sis` стоят общее число базовых станций всех компаний N ($1 \leqslant N \leqslant 10\,000$), число уже имеющихся кабельных линий M ($0 \leqslant M \leqslant 10\,000$) и число предложений о новых линиях K ($0 \leqslant K \leqslant 10\,000$). Базовые станции пронумерованы $1 \ldots N$.

На каждой из следующих M строк стоят два целых числа A_i и B_i ($1 \leqslant A_i \leqslant N$, $1 \leqslant B_i \leqslant N$, $A_i \neq B_i$), которые показывают, что между базовыми станциями A_i и B_i уже есть кабельная линия. На каждой из следующих K строк стоят три целых числа A_j, B_j и C_j ($1 \leqslant A_j \leqslant N$, $1 \leqslant B_j \leqslant N$, $A_j \neq B_j$, $0 < C_j \leqslant 10\,000$), которые показывают, что между базовыми станциями A_j и B_j можно построить новую кабельную линию по цене C_j. Предложения пронумерованы $1 \ldots K$ в порядке их появления во входных данных.

Выходные данные. На первую строку текстового файла `side.val` вывести минимальную общую стоимость необходимых работ H и число новых построенных линий L. На вторую строку файла вывести L разделённых пробелами целых числа: номера тех предложений, которые нужно принять. Порядок предложений в строке не важен. Если решений с минимальной стоимостью несколько, можно вывести любое из них. Можно считать, что в каждом тесте объединение сети возможно.

Пример.

`side.sis`
```
5 3 2
1 2
2 3
4 5
1 4 5
2 5 7
```

`side.val`
```
5 1
1
```

1. Анализ журнала

1 секунда 30 очков

В журнале компьютерной системы записи вида "*время событие ID-сессии ID-пользователя*", где *время* — время события, а *событие* — одно из `S` (пользователь вошёл в систему), `V` (пользователь вышёл из системы) или `R` (пользователь запустил перезагрузку системы, в результате чего все пользователи одновременно вышли из системы).

Идентификаторы сессий могут повторяться в журнале, но ни в один момент времени не бывает несколько активных сессий с одинаковым идентификатором. У одного пользователя может быть несколько параллельных сессий, но даже в таком случае у каждой сессии свой уникальный идентификатор.

Написать программу, которая на основании этого журнала найдёт сумму длин всех сессий каждого пользователя от начала журнала до времени последней записи в журнале и выведет эти суммы для всех пользователей. Можно считать, что журнал начинается в момент, когда ни один пользователь не находится в системе.

Входные данные. На первой строке текстового файла `logg.sis` стоит количество записей журнала N ($1 \leqslant N \leqslant 10\,000$), а на каждой из следующих N строк — одна запись в описанном выше формате. Время, а также идентификаторы сессий и пользователей записаны как целые числа $0 \ldots 10^9$. Записи идут в хронологическом порядке.

Выходные данные. В текстовый файл `logg.val` вывести ровно одну строку для каждого пользователя, представленного во входных данных. На каждую строку вывести два разделённых пробелом целых числа: идентификатор пользователя и сумму длин всех сессий данного пользователя. Порядок строк в файле не важен.

Пример.

`logg.sis`
```
4
10 S 1 1
20 S 2 2
30 V 2 2
40 V 1 1
```

`logg.val`
```
1 30
2 10
```

Пример.

`logg.sis`
```
3
10 S 1 1
20 S 2 2
40 V 1 1
```

`logg.val`
```
1 30
2 20
```

2. DNS-кодирование

1 секунда 30 очков

Система доменных имён интернета DNS не чувствительна к регистру символов, например имена `lists.ut.ee` и `LiStS.Ut.Ee` равнозначные. Следовательно, используя подходящим образом большие и маленькие буквы, можно закодировать в имени дополнительную информацию.

В этом задании нужно написать две программы, одна из которых закодирует в данное имя данную дополнительную информацию, а вторая декодирует из имени, полученного от первой программы, скрытую в нём информацию.

Кодер

Входные данные. На первой строке текстового файла `dnsx.sis` стоит количество комплектов входных данных N ($1 \leqslant N \leqslant 1000$). На следующих $2 \cdot N$ строках находится N комплектов входных данных, каждый на двух строках: на первой строке — DNS-имя S_i и на второй строке — неотрицательное целое число X_i. Имя — это последовательность маленьких латинских букв, цифр, минусов и точек длиной до 63 знаков. Можно считать, что каждое X_i такое, что при оптимальном способе кодирования в имя S_i возможно закодировать по крайней мере все целые числа $0 \ldots X_i$.

Выходные данные. В текстовый файл `dnsx.val` вывести ровно N строк, на каждую строку вывести одно имя S_i, в котором некоторые маленькие буквы могут быть заменены большими. Другие изменения в имени не разрешены.

Пример.

`dnsx.sis`	`dnsx.val`
`1`	`LiStS.Ut.Ee`
`lists.ut.ee`	
`181`	

Декодер

Входные данные. На первой строке текстового файла `dnsx.sis` стоит количество входных данных N ($1 \leqslant N \leqslant 1000$). На каждой из следующих N строк стоит одна выходная строка кодера S_i.

Выходные данные. В текстовый файл `dnsx.val` вывести ровно N строк, на каждую строку вывести одно раскодированное X_i.

Пример.

`dnsx.sis`	`dnsx.val`
`1`	`181`
`LiStS.Ut.Ee`	

Оценивание. В этом задании каждое решение получит очки только за те тесты, в которых кодер не сделает в имени других изменений кроме замены маленьких букв на большие, и декодер правильно восстановит изначальное число X по изменённому имени.

3. Делительная головка

1 секунда 40 очков

При фрезеровании шестерней используется делительная головка, у которой есть патрон для закрепления заготовки и рукоятка для вращения патрона. За N оборотов рукоятки патрон делает ровно один оборот.

Под рукояткой есть делительный диск или лимб, имеющий M круговых рядов концентрично расположенных отверстий, в i-том ряду через равные промежутки сделаны K_i отверстий. Рукоятку можно фиксировать в этих отверстиях, таким образом помимо полного оборота рукоятку можно поворачивать и на какое-либо количество отверстий.

В комплекте с головкой была таблица, где для каждого возможного числа зубцов H было указано, сколько оборотов нужно сделать рукояткой, какой ряд отверстий нужно использовать, и на сколько отверстий после полного оборота нужно ещё повернуть рукоятку, чтобы правильно поставить заготовку для фрезерования следующего зубца.

К несчастью, эта таблица потерялась, и теперь нужна программа, которая смогла бы её заменить.

Входные данные. На первой строке текстового файла `jaga.sis` стоит N ($1 \leqslant N \leqslant 100$) — число оборотов рукоятки необходимое для одного оборота патрона. На второй строке число рядов отверстий M ($1 \leqslant M \leqslant 10$). На третьей строке — M целых чисел K_i ($1 < K_1 < K_2 < \ldots < K_M \leqslant 100$). На четвёртой строке желаемое число зубцов шестерни H ($2 \leqslant H \leqslant 10\,000$).

Выходные данные. Если фрезерование желаемой шестерни невозможно, на единственную строку текстового файла `jaga.val` вывести слово `EI`. Если фрезерование возможно, на первую строку вывести слово `JAH`, и на вторую строку вывести рецепт перемещения от одного зубца к другому. Если для этого не нужно использовать лимб, вывести одно целое число — количество оборотов рукоятки. Если же нужно использовать лимб, вывести три целых числа — количество оборотов рукоятки, ряд с каким количеством отверстий нужно выбрать, и на сколько отверстий нужно повернуть рукоятку после полного оборота. Если возможных ответов несколько, вывести любой из них.

Пример.

jaga.sis	jaga.val
8 2 3 4 7	EI

Пример.

jaga.sis	jaga.val
8 2 3 4 4	JAH 2

Пример.

jaga.sis	jaga.val
8 2 3 4 12	JAH 0 3 2

Оценивание. В этом задании очки за тесты с ответом `EI` получат только те программы, которые правильно решат хотя бы один тест с ответом `JAH`.

1. Подобные треугольники

1 секунда 30 очков

Два треугольника называют подобными, если между их сторонами можно установить такое соответствие, что

- каждой стороне первого треугольника соответствует ровно одна сторона второго;
- каждой стороне второго треугольника соответствует ровно одна сторона первого;
- существует такая константа k, что отношение длин каждой стороны первого треугольника и соответствующей ей стороны второго треугольника равно k.

Написать программу, которая проверит, являются ли подобными два треугольника, заданные длинами своих сторон.

Входные данные. На первой строке текстового файла `kolm.sis` стоят три разделённых пробелами положительных целых числа a_1, b_1, c_1: длины сторон первого треугольника. На второй строке файла также стоят три разделённых пробелами положительных целых числа a_2, b_2, c_2: длины сторон второго треугольника. Можно считать, что длина ни одной из сторон не превышает 10 000.

Выходные данные. Если заданные входными данными треугольники не подобны, то на единственную строку текстового файла `kolm.val` вывести слово `EI`. Если треугольники подобны, то на первую строку текстового файла `kolm.val` вывести слово `JAH`, и на вторую строку вывести соответствие сторон треугольника в виде равенства трёх дробей аналогично приведённому ниже примеру, где в числителях стоят стороны первого треугольника, а в знаменателях — второго, каждая ровно один раз. Если возможных соответствий несколько, вывести любое из них.

Пример.

kolm.sis	kolm.val
2 3 4 3 3 3	EI

Пример.

kolm.sis	kolm.val
3 4 5 10 6 8	JAH a1/b2=b1/c2=c1/a2

Оценивание. В этом задании за тесты с ответом `EI` очки получат только те программы, которые верно решат хотя бы один тест с ответом `JAH`.

2. Сумма цифр

1 секунда 30 очков

Написать программу, которая найдёт сумму цифр данного целого числа.

Входные данные. На единственной строке текстового файла `rist.sis` стоит целое число, абсолютное значение которого не превышает 1 000 000 000.

Выходные данные. На единственную строку текстового файла `rist.val` вывести сумму цифр заданного входными данными числа.

Пример.

rist.sis	rist.val
123	6

$1 + 2 + 3 = 6$.

Пример.

rist.sis	rist.val
-1234	10

$1 + 2 + 3 + 4 = 10$.

3. Классификация слов

1 секунда 40 очков

Известно, что на эстонском имена всегда пишутся с большой буквы, а прочие слова пишутся с большой буквы только в начале предложения.

Написать программу, которая по этому правилу классифицирует слова, представленные в данном тексте, на имена, прочие слова и такие слова, классификация которых на основании данного текста невозможна.

Входные данные. В текстовом файле `liik.sis` находится до 30 строк, каждая длиной до 80 знаков. В файле могут присутствовать большие и маленькие латинские буквы, пробелы, запятые, точки, вопросительные и восклицательные знаки. Точки, а также вопросительные и восклицательные знаки заканчивают предложение, в то время как запятые, пробелы и переводы строк лишь разделяют слова. На последней строке файла стоит лишь одна точка, и ничего больше.

Выходные данные. В текстовый файл `liik.val` вывести ровно три строки. На первую строку файла вывести все представленные во входных данных слова, которые на основании вышеприведённого правила можно отнести к именам. На вторую строку файла вывести те слова, о которых можно утверждать, что они не имена. На третью строку файла вывести те слова, о которых нельзя с уверенностью сказать ни того, ни другого. Порядок слов на строках и их представление большими или маленькими буквами не имеет значения. Однако каждое слово нужно вывести ровно один раз, и слова должны быть разделены пробелами.

Пример.

`liik.sis`
```
Kati on Mati sober.
Mati on Kati sober ka.
Mari on nende molema sober.
.
```

`liik.val`
```
Mati Kati
on sober ka nende molema
Mari
```

Пример.

`liik.sis`
```
Mari ja Marju
leidsid palju marju.
.
```

`liik.val`
```

ja leidsid palju
Mari Marju
```