1. Цены на бензин

1 секунда 20 очков

Экономический аналитик Томас каждый день записывает цену на бензин, которая горит на бензозаправке напротив его дома. Теперь он хочет на основании этих данных найти самые длинные периоды подорожания и удешевения бензина. Написать программу, которая помогла бы ему это сделать.

Входные данные. В первой строке текстового файла `bh.sis` дано целое число N ($1 \leqslant N \leqslant 10\,000$): длина последовательности цен, записанной Томасом. Во второй строке файла дано N разделённых пробелами целых чисел H_i ($0 < H_i \leqslant 10\,000$, $1 \leqslant i \leqslant N$): цена одного литра бензина в соответствующий день.

Выходные данные. В первую строку текствого файла `bh.val` вывести два разделённых пробелам целых числа A и L, которые обозначают, что самый длинный период, в течение которого цена на бензин ни разу не понизилась, начался в день номер A и закончился в день номер L. Если периодов с максимальной длительностью несколько, вывести самый последний из них. Во вторую строку файла вывести в таком же виде самый длинный период, в течение которого цена на бензин ни разу не повысилась.

Пример.

bh.sis	bh.val
7	1 3
1 2 3 2 3 2 1	5 7

2. Курсор мыши

1 секунда 20 очков

Программист Прийт для передвижения находящегося на экране компьютера окна должен передвинуть курсор мыши к краю этого окна. Окно прямоугольной формы, его края параллельны краям экрана, и для перетаскивания его нужно захватить мышкой за любой край или угол. Как типичный программист, Прийт стремится минимизировать физическую нагрузку. Написать программу, которая подсчитает для Прийта, за какую точку ему нужно перетаскивать окно.

Входные данные. В первой строке текстового файла `hk.sis` дано два разделённых пробелами целых числа X и Y ($0 \leqslant X \leqslant 10\,000$, $0 \leqslant Y \leqslant 10\,000$): координаты начального положения курсора мыши. Во второй и третьей строках файла дано также по два разделённых пробелами целых числа: координаты соответственно левого верхнего угла X_1 и Y_1 и нижнего правого угла окна X_2 и Y_2 ($0 \leqslant X_1 \leqslant X_2 \leqslant 10\,000$, $0 \leqslant Y_1 \leqslant Y_2 \leqslant 10\,000$).

Выходные данные. В единственную строку текстового файла `hk.val` вывести два разделённых пробелами целых числа: координаты точки на краю окна, ближайшей к начальному положению мыши. Если ближайших точек несколько, вывести любую из них.

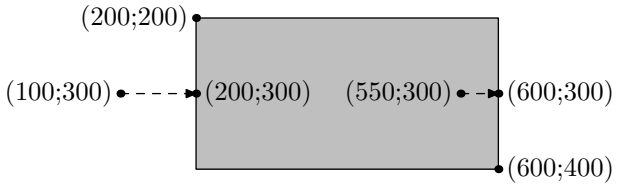
Пример.

hk.sis	hk.val
100 300	200 300
200 200	
600 400	

Пример.

hk.sis	hk.val
550 300	600 300
200 200	
600 400	

Примеры иллюстрирует следующий рисунок:



3. Временные интервалы

1 секунда 20 очков

Веб-сервер логирует дату и время всех запросов с точностью до секунды. Для подсчёта статистики необходимо сгруппировать записи логов по разным временным интервалам, например, по дням или по часам.

Проще всего все временные интервалы одной длительности представлять через начальный момент соответствующего интервала, округляя в нижнюю сторону дату и время. Например, при округлении момента времени 18:22:55 в нижнюю сторону до 5-минутного интервала, получаем 18:20:00.

Написать программу, которая по заданному моменту времени вычисляет начальную точку соответствуюшего временного интервала с заданной длительностью.

Входные данные. В первой строке текстового файла `ai.sis` задан изучаемый момент времени в виде $YYYY$-MM-DD hh:mm:ss, где $YYYY$ обозначает год, MM — месяц, DD — день, hh — час, mm минуту, и ss — секунду. Номер года всегда задаётся четырёхзначный, все остальные числа двухзначные. Можно предполагать, что данные дата и время всегда корректные. Во второй строке файла задана длительность временного интервала: `MIN` (1 минута), `5MIN` (5 минут), `HOUR` (час), `DAY` (сутки), `MON` (месяц) или `YEAR` (год).

Выходные данные. В единственную строку текстового файла `ai.val` вывести начальный момент времени искомого временного интервала в том же формате, что и во входных данных.

Пример.

`ai.sis`
```
2008-11-10 09:08:07
MIN
```
`ai.val`
```
2008-11-10 09:08:00
```

Пример.

`ai.sis`
```
2008-11-10 09:08:00
5MIN
```
`ai.val`
```
2008-11-10 09:05:00
```

Пример.

`ai.sis`
```
2008-11-10 00:00:00
MON
```
`ai.val`
```
2008-11-01 00:00:00
```

4. Грузоперевозки

1 секунд 40 очков

Правительство Эльбонии решило инвестировать в железнодорожные грузоперевозки и приобрести новые локомотивы. Увы, сами железные дороги тоже не в лучшем состоянии, поэтому для каждого железнодорожного пути установлен лимит на максимальную нагрузку.

Написать программу, которая находит максимальную нагрузку, с которой локомотив может проехать из любого города Эльбонии в любой другой (хотя, возможно, и не самым коротким путём).

Входные данные. В первой строке текстового файла `kv.sis` дано два разделённых пробелами целых числа: количество городов N ($2 \leqslant N \leqslant 100$) и количество железнодорожных путей M ($1 \leqslant M \leqslant 1000$). Города пронумерованы числами $1 \ldots N$. В каждой из следующих M строк дано три разделённых пробелами целых числа: для i-го железнодорожного пути номера городов A_i и B_i, между которыми он проходит ($1 \leqslant A_i \leqslant N$, $1 \leqslant B_i \leqslant N$, $A_i \neq B_i$), и максимальная нагрузка C_i, допустимая на этом пути ($0 < C_i \leqslant 10\ 000$). Между двумя городами может быть и несколько железнодорожных путей.

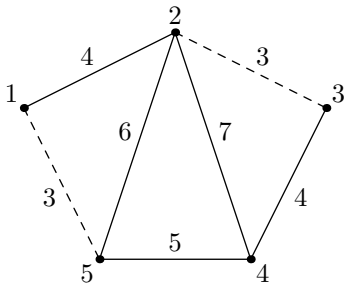
Выходные данные. В единственную строку текстового файла `kv.val` вывести одно целое число: максимальную грузоподъёмность, с которой можно перевозить товар из любого города Эльбонии в любой другой.

Пример.

`kv.sis`
```
5 7
1 2 4
1 5 3
2 3 3
2 4 7
2 5 6
3 4 4
4 5 5
```

`kv.val`
```
4
```

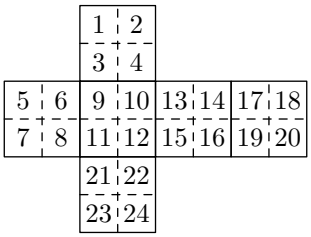
На приведённом ниже рисунке все железнодорожные пути, помеченные сплошной линией, выдерживают нагрузку как минимум 4 единицы, и через них можно проехать из любого города в любой другой. Для большей массы груза это было бы уже невозможно.



5. Кубик Рубика

80 очков

Малый кубик Рубика состоит из $2 \times 2 \times 2$ соединённых между собой единичных кубов. Пронумеруем наружние грани этих кубов, как показано на следующей развёртке кубика:



В начальном положении каждая грань кубика окрашена в один цвет, отличный от цветов других граней. Если обозначим эти цвета буквам `A`...`F`, то сможем каждое состояние кубика описать 24-буквенным словом, в котором буква на i-ой позиции показывает, в какой цвет окрашен квадрат, находящийся на i-ом месте на развёртке кубика. В начальном состоянии

квадраты 1...4 окрашены в цвет `A`, квадраты 5...8 — в цвет `B`, и т.д. Обозначение этого состояния `AAAABBBBCCCCDDDDEEEEFFFF`.

Далее, на каждом шаге слои кубика можно повернуть в трёх разных плоскостях относительно друг друга на 90°, 180° или 270°. Итого в каждом состоянии можно сделать девять разных ходов. Поворачивая грань, содержащую квадраты 1...4, на 90° по часовой стрелке, получим состояние `AAAACCBBDDCCEEDDBBEEFFFF`. Поворачивая затем грань, содержащую квадраты 9...12, на 90° потив часовой стрелки, получим состояние `AAEDCABADCDCFEFDBBEECBFF`.

Найти способ, как перейти из заданного состояния обратно в начальное состояние за минимальное число шагов. Перед тем, как начать делать ходы из данного положения, кубик можно перевернуть в любое положение (то есть любой гранью наверх и любой гранью к себе). Этот разворот не считается за шаг.

Входные данные. В единственной строке текстового файла `rk.sis` дано слово длиной 24, состоящее из букв `A`...`F`.

Выходные данные. Если заданное слово — корректное описание состояния кубика Рубика, вывести в первую строку текстового файла `rk.val` число шагов N, необходимых для возвращения кубика в начальное состояние. Во вторую строку вывести состояние кубика перед тем, как начать делать шаги (оно может отличаться от заданного состояния поворотом целого кубика). В следующие N строк вывести состояния кубика после каждого шага.

Если заданное слово не является корректным описанием состояния кубика Рубика, вывести в первую строку текстового файла `rk.val` слово `VIGA` и начиная со второй строки — обоснование. Обоснование вывести в свободной текстовой форме (его оценивает человек) на эстонском, русском или английском языке.

Пример.

`rk.sis`
```
ADAEBBEECABADCDCFEFDFCFB
```

`rk.val`
```
2
AAEDCABADCDCFEFDBBEECBFF
AAAACCBBDDCCEEDDBBEEFFFF
AAAABBBBCCCCDDDDEEEEFFFF
```

Оценивание. В этом задании будет предоставлено 12 комплектов входных данных в файлах `rktest.01.sis` до `rktest.12.sis`, и в качестве решения нужно предоставить соответствующие им комплекты выходных данных в файлах `rktest.01.val` до `rktest.12.val`. Представлять вашу программу не обязательно, она не оценивается.

Выполнение разных частей задачи приносит очки следующим образом:

- Классифицирование входных данных на корректные и некорректные: 10% от стоимости задания.
- Обоснование причины некорректных данных: 30% от стоимости задания.
- Решение корректных входных данных: 30% от стоимости задания.
- Решение корректных входных данных за минимальное число шагов: 30% от стоимости задания.