1. Завтрашний день

1 секунда — 20 очков

Написать программу, которая находит дату, следующую за данной (по Грегорианскому календарю).

Входные данные. На единственной строке текстового файла `homme.sis` дана дата в формате *ДД.ММ.ГГГГ* (причем такая дата существует, и $1000 \leqslant ГГГГ \leqslant 9000$).

Выходные данные. На единственной строке текстового файла `homme.val` вывести дату, следующую за данной во входном файле в том же формате.

Пример.

homme.sis	homme.val
01.01.2004	02.01.2004

Пример.

homme.sis	homme.val
31.12.2004	01.01.2005

Примечание. Високосными считаются года делящиеся на четыре, кроме тех, которые делятся на сто (это невисокосные года), кроме тех, которые делятся на четыреста (эти — опять високосные).

Примечание. Для решения задания НЕ разрешается менять системную дату компьютера. Решения, нарушившие это требование, будут дисквалифицированы.

2. Ход конём

1 секунда — 40 очков

Даны координаты двух полей на шахматной доске. Требуется написать программу, которая сможет найти способ шахматному коню добраться из одного поля в другое за наименьшее число ходов.

Входные данные. На первой строке текстового файла `ratsu.sis` даны координаты начального поля в формате VR, где V указывает вертикаль $a \ldots h$, а R — горизонталь $1 \ldots 8$. На второй строке даны координаты конечного поля в том же формате.

Выходные данные. На первой строке текстового файла `ratsu.val` вывести минимальное число ходов K, которое необходимо чтобы добраться из начального поля в конечное. На следующих $K+1$ строках вывести путь коня начиная с исходного поля. Если кратчайших путей несколько, вывести любой из них.

Пример.

ratsu.sis	ratsu.val
a1	3
b1	a1
	c2
	a3
	b1

Примечание. За один ход шахматный конь передвигается на 2 поля по-горизонтали и 1 поле по-вертикали или же на 1 поле по-горизонтали и 2 — по-вертикали.

Оценивание. В этом задании 50% очков получат решения, которые смогут найти только лишь длину кратчайшего пути, но не сам путь.

3. XOR-шифр

открытые тесты — 40 очков

Операция “исключающее или” (которую также называют XOR и обозначают как $\oplus$) является логичеслой операцией на двух логических величинах, результат которой “верно”, если ровно один из операндов равен “верно”, и “неверно” во всех остальных случаях.

При “сложении” двоичных чисел операцией $\oplus$, их биты рассматривают как логические величины (1 = “верно”, 0 = “неверно”) и каждый бит результата считается как XOR соответствующих битов аргументов (младший бит результата — это XOR младших битов аргументов, второй бит результата — XOR вторых битов аргументов, и.т.д.). Например, $0101_2 \oplus 1100_2$ считается следующим образом: младший (самый правый) бит результата равен $1 \oplus 0 = 1$; следующий бит равен $0 \oplus 0 = 0$; далее $1 \oplus 1 = 0$; и наконец $0 \oplus 1 = 1$. В итоге получаем $0101_2 \oplus 1100_2 = 1001_2$. Так как $\oplus$-сумма двух битов всегда один бит, переводов в следующий разряд не бывает.

Рассмотрим классический алгоритм шифровки данных с помощью операции $\oplus$. Для того, чтобы зашифровать l-битовое двоичное число t используется l-битовый ключ k и результатом шифровки (криптограммой) является исключающее или исходного числа и ключа, т.е. число $s = t \oplus k$. Тот, кто знает криптограмму s и ключ k сможет восстановить изначальное сообщение t (т.е. расшифровать криптограмму). Для этого ему всего лишь нужно вычислить $t = s \oplus k$. Это так, потому что операция $\oplus$ обратна сама себе: для любых t и k выполняется $s \oplus k = t \oplus k \oplus k = t$.

Случилось так, что одно сообщение T зашифровали несколькими разными ключами, и после этого оригинал сообщения потерялся. Остались однако все N ключей K_1, K_2, ..., K_N, использованных для шифровки, и все полученные N криптограмм E_1, E_2, ..., E_N, только неизвестно каким ключом была получена какая криптограмма. Необходимо по этим данным восстановить изначальный текст T.

Входные данные. На первой строке входного файла `xor.sis` даны числа N и L, разделённые пробелом. На каждой из следующих N строк даны L разделённых пробелами 8-битовых двоичных чисел: ключи $K_{1...N}$, которые были использованы для шифровки сообщения T. На следующих N строках также по L разделённых пробелами 8-битовых двоичных чисел: полученные криптограммы $E_{1...N}$ в случайном порядке.

Выходные данные. На единственную строку текстового файла `xor.val` вывести L разделённых пробелами 8-битовых двоичных чисел: один из возможных изначальных текстов T. Все числа вывести в точности 8-битовыми (т.е. числа могут начинаться с нулей).

Пример.

`xor.sis`
```
3 3
01010101 01010101 01010101
10101010 10101010 10101010
11111111 11111111 11111111
10111010 10110110 10110000
00010000 00011100 00011010
11101111 11100011 11100101
```

`xor.val`
```
01000101 01001001 01001111
```

Проверка правильности ответа: Зашифруем $T = 01000101_2 01001001_2 01001111_2$ ...

- ... ключом $K_1 = 01010101_2 01010101_2 01010101_2$
  получим криптограмму $E_1 = 00010000_2 00011100_2 00011010_2$,
  которая находится на строке 6 входного файла.
- ... ключом $K_2 = 10101010_2 10101010_2 10101010_2$
  получим криптограмму $E_2 = 11101111_2 11100011_2 11100101_2$,
  которая находится на строке 7 входного файла.
- ... ключом $K_3 = 11111111_2 11111111_2 11111111_2$
  получим криптограмму $E_3 = 10111010_2 10110110_2 10110000_2$,
  которая находится на строке 5 входного файла.

Оценивание. В этом задании предоставлены входные файлы `xortest.01.sis` до `xortest.10.sis` и в качестве решения необходимо предоставить соответствующие выходные файлы `xortest.01.val` до `xortest.10.val`. Программу предоставлять не надо, это не оценивается.