1. Префиксная архивация

1 секунда — 30 очков

Префиксная архивация — это двухшаговый алгоритм для архивации совокупности слов:

1. Сортируем слова в алфавитном порядке.
2. В отсортированном списке, для каждой пары идущих подряд слов $a_1a_2\ldots a_{n_a}$, $b_1b_2\ldots b_{n_b}$ находим наибольший такой k, что для всех $1 \leqslant i \leqslant k$ выполняется $a_i = b_i$. Если $k > 0$, преобразуем второе слово к виду $kb_{k+1}b_{k+2}\ldots b_{n_b}$.

   Другими словами: в каждом слове, если у него есть префикс, равный префиксу предыдущего слова, заменяем этот префикс на его длину (в десятичной системе счисления). При разархивации мы можем это слово легко восстановить, так как у нас есть предыдущее слово и длина префикса, который нужно из него взять.

Написать программу, которая заархивирует с помощью описанного алгоритма заданую совокупность слов.

Входные данные. В первой строке текстового файла `pref.sis` дано количество слов N ($1 \leqslant N \leqslant 10\,000$), и в каждой из следующих N строк дано одно состоящее из маленьких латинских букв слово длиной $1\ldots30$ символов.

Выходные данные. В текстовой файл `pref.val` вывести ровно N строк: заархивированные слова, в каждой строке по одному слову. При сортировке символы сравнить по их ASCII-кодам.

Пример.

pref.sis	pref.val
4	kala
post	2ss
kass	3t
kala	post
kast	

Слова `post`, `kass`, `kala`, `kast`, отсортированные в алфавитном порядке — `kala`, `kass`, `kast`, `post`, из чего видим, что у слов `kala` и `kass` есть общий префикс `ka` длиной 2, у слов `kass` и `kast` общий префикс `kas` длиной 3 и у слов `kast` и `post` общего префикса нет. Найденные общие префиксы заменяем их длинами и получаем результат: `kala`, `2ss`, `3t`, `post`.

2. Числа-палиндромы

1 секунда — 30 очков

Целое число называют палиндромом, если оно читается одинаково как слева направо, так и справа налево. Например, число 121 — палиндром, а 123 — нет.

Написать программу, которая для заданного числа находит ближайшие палиндромы.

Входные данные. В единственной строке текстового файла `pal.sis` дано число N ($0 \leqslant N \leqslant 9\,223\,372\,036\,302\,733\,229$).

Выходные данные. В первую строку текстового файла `pal.val` вывести число N_1: максимальный палиндром, не больше N. Во вторую строку вывести N_2: минимальный палиндром, не меньше N.

Пример.

pal.sis	pal.val
123	121
	131

Оценивание. Вывод N_1 и N_2 оценивается отдельно, каждое из них даёт 50% от стоимости теста. Если программа не смогла найти один из этих палиндромов, вывести вместо него число -1.

3. Цирк

1 секунда 40 очков

В настольной игре "Цирк" игральная доска состоит из N клеток, которые пронумерованы $1 \ldots N$. Клетка номер 1 начальная, и клетка номер N — конечная. В начале игры фишку игрока устанавливают на начальную клетку. Цель игрока — попасть в конечную клетку.

На каждом ходу игрок кидает обычный 6-гранный кубик и двигает фишку вперёд на столько клеток, сколько точек выпало на кубике ($1 \ldots 6$) (при выпадении на кубике числа s фишка двигается из клетки r в клетку $r + s$).

Некоторые клетки соединены между собой односторонними туннелями. Когда фишка после броска кубика останавливается на клетке, являющейся началом туннеля, она сразу передвигается к концу туннеля. Если фишка останавливается в конце туннеля, ничего не происходит.

Написать программу, которая находит минимальное возможное количество бросков кубика, которым можно попасть в конечную клетку, если все броски дадут подходящие результаты.

Входные данные. В первой строке текстового файла `ts.sis` дано количество клеток игрового стола N ($1 \leqslant N \leqslant 1000$) и количество туннелей M ($0 \leqslant 2 \cdot M \leqslant N$). В каждой из следующих M строк дано два разных целых числа: номера входной и выходной клеток одного туннеля. Известно, что ни одна клетка не соединена с более чем одном туннелем. Начальная и конечная клетки не являются началом туннеля.

Выходные данные. В первую строку текстового файла `ts.val` вывести искомое количество бросков кубика K, и в каждую из следующих K строк вывести одно целое число: числа, выпавшее на кубике в порядке бросков кубика. На последнем ходу фишка должна остановиться точно на последней клетке, т.к. перепрыгивать её нельзя. Известно, что попасть в конечную клетку возможно. Если оптимальных решений несколько, вывести любое из них.

Пример.

`ts.sis`	`ts.val`
`16 2`	`2`
`2 10`	`1`
`12 6`	`6`

Игру можно закончить за два броска, если

1. на кубике выпадет 1, перейти из клетки 1 в клетку 2 и оттуда по туннелю в клетку 10;
2. на кубике выпадет 6, перейти из клетки 10 в клетку 16.

Естественно, одним броском в конечную клетку попасть невозможно.