1. Светофоры

10 секунд 100 очков

Вдоль дороги стоят N светофоров. В каждом светофоре попеременно горит то K секунд зелёный, то K секунд красный свет. Необходимо проехать из одного конца дороги в другой с некой заданной постоянной скоростью так, чтобы в момент проезда у каждого светофора в нём горел бы зелёный свет, либо зелёный свет зажигался или гас бы в тот самый момент. Известно время, которое необходимо для того, чтобы с этой скоростью проехать отрезок пути от каждого светофора до следующего.

Написать программу, которая найдёт для каждого светофора момент времени, когда в нём необходимо включить зелёный свет так, чтобы дорогу можно было бы проехать описанным образом в обе стороны (т. е. для каждого конца дороги найдётся момент времени такой, что машина, начавшая движение в этот момент у первого светофора, проезжает все светофоры на зелёный свет).

Входные данные. На первой строке текстового файла `foor.sis` даны разделённые пробелом числа N и K ($1 \leqslant N \leqslant 500\,000$, $1 \leqslant K \leqslant 1\,000\,000\,000$). На следующих $N-1$ строках даны числа t_1, t_2, …, t_{N-1}, где t_i ($1 \leqslant t_i \leqslant 1\,000\,000\,000$) означает время, которое необходимо чтобы проехать отрезок дороги между светофорами i и $(i+1)$ с заданной скоростью.

Выходные данные. На первых N строках текстового файла `foor.val` вывести целые числа s_1, s_2, …, s_N, где s_i ($0 \leqslant s_i < 2 \cdot K$) означает момент времени, в который в светофоре i должен зажечься зелёный свет. На последней строке вывести два разделённых пробелом числа: u_1 и u_2 ($0 \leqslant u_1, u_2 < 2 \cdot K$) — моменты времени, когда необходимо начинать движение у первого и последнего светофоров соответственно. Если найдётся несколько решений, вывести любое из них. Известно, что решение найдётся.

Пример.

`foor.sis`
```
4 20
10
15
20
```

`foor.val`
```
0
10
25
5
0 15
```

2. Вычитание слов

1 секунда 100 очков

Определим «вычитание» слов следующим образом: разница $A - B$ слов A и B это слово, которое получается удалением из слова A всех букв, которые также были в слове B.

При таком вычитании удаляется каждая буква из слова A столько раз, сколько она присутствует в слове B, например `KALAKALA` − `AAA` = `KLKLA`, `ELEVANT` − `TALV` = `EEN`, `AABBCC` − `CBA` = `ABC`. Вычитание возможно лишь в том случае, если каждая буква присутствует в слове A не меньшее количество раз, чем в слове B, например вычитать `ABC` − `AA` нельзя.

Дано одно «длинное» и несколько «коротких» слов. Написать программу, которая вычитает из «длинного» слова некое количество «коротких» так, чтобы результат был бы минимальной возможной длины.

Входные данные. На первой строке текстового файла `sona.sis` дано «длинное» слово A, на второй — число «коротких» слов N ($1 \leqslant N \leqslant 21$) и на каждой из следующих N строк по одному «короткому» слову B_i. Слова состоят только из заглавных латинских букв и каждое слово не длиннее 250 букв.

Выходные данные. На единственной строке текстового файла `sona.val` вывести длину искомого минимального слова C которое можно получить последовательно вычитая из слова A некое количество слов B_i. Каждое из слов B_i можно вычитать не более одного раза.

Пример.

`sona.sis`
```
KALAKAJAKAS
4
KAJAKAS
KLAAS
AJA
KA
```

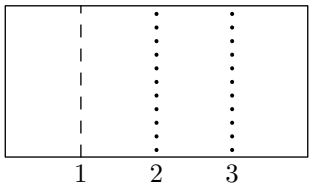
`sona.val`
```
1
```

Минимальное слово получается следующим образом: `KALAKAJAKAS` − `KLAAS` = `KAJAKA`, `KAJAKA` − `AJA` = `KKA`, `KKA` − `KA` = `K`.

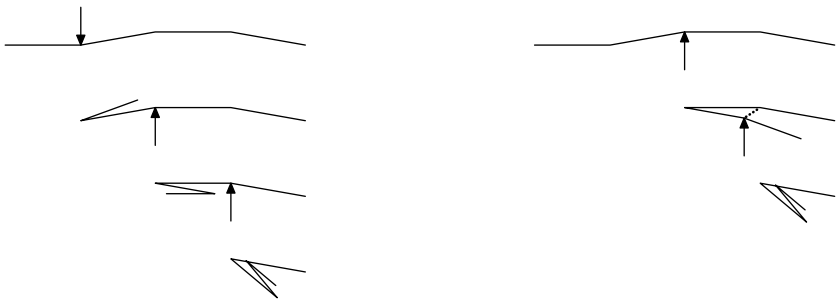
3. Брошюра

2 секунды 100 очков

Инструкция по использованию карманной USB-памяти представляет из себя брошюру: сложенный несколько раз вдоль прямоугольный листок бумаги. В расправленном виде на листе видны параллельные линии сгиба, находящиеся на равных расстояниях друг от друга. Некоторые линии сгиба выгнуты «вниз», а некоторые — «вверх» (на приведённом рисунке линия 1 выгнута вниз, а линии 2 и 3 — вверх).



Развёрнутую брошюру можно сложить обратно сгибая по линиям подряд, как показано на следующем рисунке слева: сначала сгибаем вниз по линии 1, затем вверх по линии 2 и ешё раз вверх по линии 3. Однако сгибать можно и несколько слоёв бумаги одновременно, поэтому брошюру можно сложить и за две операции как показано на рисунке справа: сначала по линии 2 вверх, затем сразу по двум оказавшимся друг над другом линиям 1 и 3 вверх.



Написать программу, которая найдёт минимальное количество операций, за которое можно свернуть брошюру с заданной конфигурацией линий сгиба. Можно предполагать, что сгибать одновременно можно сколь угодно толстую пачку.

Входные данные. На первой строке текстового файла `voldik.sis` дано количество линий сгиба на листе N ($0 \leqslant N \leqslant 500$) и на второй строке описание линий сгиба слева направо: N разделённых пробелами чисел, где 0 означает выгиб вверх, а 1 — выгиб вниз.

Выходные данные. На единственной строке текстового файла `voldik.val` вывести минимальное необходимое количество операций сгибания K.

Пример.

`voldik.sis`	`voldik.val`
`3` `0 1 1`	`2`