Problem K: Key Knocking

Time limit: 1 s
Memory limit: 512 MiB

Goran is recovering from his knee surgery and is experimenting with a smart card used for storing cryptographic keys. In this problem, a *key* is a bitstring of size $3n$ where n is a positive integer. Particular bits of the key are indexed with integers 1 through $3n$ left to right. *Weight* of a key is the number of pairs of different neighbouring bits increased by one. For example, the weight of the key "000" is 1, while the weight of the key "011010100" is 7.

He has discovered that he can tamper with the key by sending small jolts of electricity through the circuits of the smart card. In particular, he can reliably perform the following operation: pick two arbitrary neighbouring bits of the key and flip both of them. For example, one operation can change the key "000" to "110".

Given a key of size $3n$ find any sequence of at most n operations that transforms the given key to a key of weight at least $2n$. You may assume that a solution always exists.

Input

The first line contains a string consisting of digits "0" and "1" — the initial key. The length of the key is $3n$ where n is a positive integer such that $1 \leq n \leq 100\,000$.

Output

The first line should contain an integer m where $0 \leq m \leq n$ — the number of operations in your solution. The following line should contain m integers $a_1, a_2, \ldots, a_m$ describing your solution. The number a_k is the index of the left one of two bits being flipped in the k-th step.

If the initial key already has weight at least $2n$ you may output only the integer 0.

Example

input	input	input
000000000	111001000111	010101
output	**output**	**output**
3 2 5 6	2 3 9	0