Problem 4. Fix the heap 8-bit

Input file:	`input.bin`
Output file:	`output.bin`
Time limit:	1 second
	3 seconds (for Java)
Memory limit:	**64** megabytes

Petya wrote a heap for dynamic memory allocation in his own programming language Tse. Here is how it works.

A correct heap is a continuous segment of memory consisting of N cells. Each cell contains an unsigned **8-bit** integer, which can have any value **from 0 to 255** inclusive. The heap is split into B sub-segments called blocks. Each of the memory cells belongs to strictly one block.

Each block consists of two or more cells. The first and the last cells of a block contain the effective size of the block, which is the number of cells in it, excluding the first and last cells. The rest of the block's cells can contain arbitrary data regardless of whether the block is free or occupied.

Tse is a very low-level language, and its users often mess everything up and corrupt the heap by inadvertently writing their data into wrong cell. Users are asking Petya to come up with a feature of recovering heap integrity after such incidents. The recovery code must analyze the contents of N cells of the memory segment where the heap is supposed to be located, and change the contents of several memory cells in such a manner that the segment becomes a correct heap. The number of modified cells must be minimal.

Input

The contents of N memory cells comprising the analyzed memory segment are provided in a binary file of size N bytes. Each byte of the file represents a single memory cell. $2 \le N \le 2^{24}$ holds true, meaning that the size of the file is not less than two bytes and not greater than 16 megabytes.

Output

Write the found set of K cell changes into a binary file of size $4K$ bytes.

Each change is described by four bytes. The first three bytes are treated as an unsigned 24-bit integer and define the address (index) of the modified cell. Naturally, this number must be smaller than N — the size of the input file. The last, fourth byte defines the new contents of the cell.

24-bit addresses must be written in little-endian byte order, with the first byte being the least significant one and the last byte being the most significant one. If there are several answers with the same number of changes, you can write any of them. The order of describing cell changes in the output file is irrelevant.

Examples

For convenience, the contents of the input and the output files are shown below in hex format. In the testing system, the files will be in binary! You can download samples in binary format in the «News» tab near the problem statements.

For fast file reading it is recommended to use one call of: `Files.readAllBytes` on Java and `fread` on C++. Don't forget that files should be opened in binary mode on C++.

input.bin	output.bin
02 AA BB 02 00 00 03 AB AC AD 03	
02 AA BB CC 00 DD 03 AB AC AD 03	03 00 00 02 05 00 00 00

Example explanation

In both examples $N = 11$. In the first example the heap is already correct, containing three blocks with effective sizes 2, 0, and 3, respectively. Since no changes are necessary, the output file must be empty. In the second example, the heap is incorrect, but it can be fixed with two changes, by turning it exactly into the heap from the first example. For this purpose, it is suggested that the cell with the address 3 be changed from `CC` to `02`, and the cell with the address 5 be changed from `DD` to `00`.

Illustration for the second example (integers displayed in decimal format):