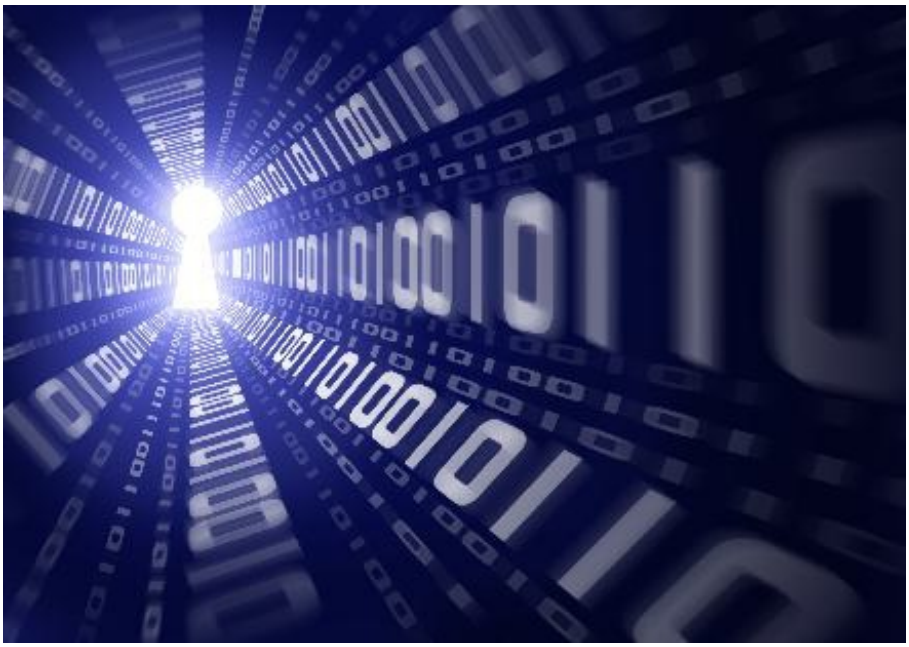
Telecommunications

From beacons over telegraphs and telephones towards radio and television and ultimately computer networks and internet – telecommunication systems knew a tremendous evolution within the last century! This was also clear to Richard Hamming, who was bothered by erroneous data transmission.

As the transmission channel is often very noisy, the received information is not necessarily the information that was sent out. Looking at digital data transmission, a straight-forward way to check for errors would be a supplementary bit of information, a so-called parity bit. Each package of data on *n-1* bits is completed by 1 parity bit, whose value is chosen such that the number of ones in the *n*-bit package is even. Although there is little overhead, parity checking is not very robust. If an even number of bits is flipped during transmission, the check bit remains valid and the error will not be detected. Moreover, although parity can detect errors, it provides no indication on the position of the flipped bit. The data must be discarded entirely and re-transmitted.



If more error-correcting bits are included with a message, and if those bits can be arranged such that different incorrect bits produce different error results, then bad bits could be identified and corrected! Among others, Hamming invented the following algorithm that generates a single-error correcting code for any number of bits.

Hamming single-error correcting code is the same as described in this Wikipedia article.

Suppose the original message is described by an unsigned 64-bit integer X. Transform it into a 64-bit 01 string S (Least significant digit first), then use the above procedure to encode it into a 01 string S2. Transform it into a (not necessarily 64-bit) integer Y. Now you are given the integer Y in decimal notation, with at most 1-bit error. Restore the original integer X in decimal notation.

Input

Input contains several lines, each containing a single non-negative integer (in decimal notation). Input terminates with -1.

Output

For each input line output one line with a single integer - the decoded message (in decimal notation).

Example

Input:
0
81
-1

Output:
0
11

Explanation

11 → (1101) (least significant digit first) → encode (1010101) → error in bit 2 (1000101) → 81 (decimal)