Labyrinth

Write a programme that searches for a path in a multi-dimensional maze.

Definitions

A maze is contained in a non-empty k-dimensional compartment (k-dimensional cuboid) consisting of k-dimensional unit cubes. Each such cube can be filled, forming the walls of the maze, or empty, forming a space in which to move. The maze can be navigated by passing between empty cubes touching each other with a (k-1)-dimensional wall.

The position of each cube (empty or filled) is determined by its coordinates, which are positive integers.

The path in the maze is a sequence of transitions between empty cubes from the start position to the end position. The start and end positions are defined by giving the coordinates of the two empty cubes. The path length is the number of transitions. If the end position is also the start position, the path has length zero.

Input data

The program reads data from standard input. Valid data consists of four lines of text. The three initial lines each contain k integers between 1 and SIZE_MAX, where k is a positive integer specifying the dimension of the maze. These are respectively:

n_1, n_2, n_3, ..., n_k,
x_1, x_2, x_3, ..., x_k,
y_1, y_2, y_3, ..., y_k,
where the numbers n_i specify the size of the maze in each dimension, and the numbers x_i and y_i specify the coordinates of the start and end positions, respectively.

The fourth line contains an integer describing the positions of the maze walls. In the binary expansion of this number, the bit (z_1-1)+(z_2-1)n_1+(z_3-1)n_1n_2+...+(z_k-1)n1_n2...n_k-1 specifies the cube with coordinates (z_1,z_2,z_3,...,z_k). If this bit is 0, the cube is empty, and if it is 1, the cube is filled.

The number on the fourth line can be specified in two ways:

a hexadecimal number starting with the character combination 0x, written using the digits 0, 1, ..., 9, a, b, ..., f, A, B, ... F.
character R, followed by five integers between 0 and UINT32_MAX written at base 10: a, b, m, r, s_0, used as follows, with the additional condition that the number m is not zero.
We calculate the numbers s_1, s_2, s_3, ..., s_r from the formula s_i=(as_i-1+b)mod m. We calculate the remainders w_i=s_i mod n_1n_2...n_k. A number describing the position of the walls of the maze has the bit number j set in the binary expansion, if there is such an index i that
j mod 2^(32)=w_i.

Numbers may be preceded by leading zeros. Numbers occurring on one line are separated by any number of whitespace characters. There may be no space between the R character and the first number, or there may be any number of whitespace characters. There may be any number of whitespace characters at the beginning and end of each line.

Description of programme operation

The program reads the data and then, if correct, determines the length of the shortest path from the start position to the end position. The programme prints one line to the standard output (terminated with a transition to a new line, i.e. in the C language with the ASCII code 10) containing the length of the found path (without leading zeros) or the message NO WAY, if there is no path. The program terminates with the code 0. The program should explicitly release the allocated memory.

Error handling
If the input data is invalid, the program prints a single line (terminated by a transition character to a new line) to the standard diagnostic output containing the message ERROR L, where L is an integer between 1 and 5. This is the number of the first line associated with the error. If an unrecoverable error has occurred, e.g. a library function has failed, memory has run out, the result cannot be determined, the program prints one line (terminated by a transition to a new line character) to the standard diagnostic output containing the message ERROR 0. The program terminates with the code 1. The program should explicitly release the allocated memory before terminating, unless a memory allocation error has occurred.