Problem H. Spaceship

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are piloting a spaceship in the three-dimensional space.

Your spaceship is currently located in the point (x_1, y_1, z_1), flying in the direction (vx_1, vy_1, vz_1). Your goal is to get to the point (x_2, y_2, z_2), flying in the direction (vx_2, vy_2, vz_2). Your spaceship can only fly in straight lines or in circular arcs with radius at least r_0. When changing between two segments of the flight (a line and an arc, an arc and a line, two different arcs, or two different lines, however pointless that last option is), the direction in the inflection point must be the same (all direction vectors are given up to a positive multiplier, so directions $(1, 1, 1)$ and $(2, 2, 2)$ are the same, but $(1, 1, 1)$ and $(-1, -1, -1)$ are not the same).

You need to find any trajectory to do so. Note that the trajectory does not need to be the shortest, it only needs to fit the relatively loose bounds described in the output format section.

Input

The first line of the input contains three integers x_1, y_1 and z_1 ($-10 \le x_1, y_1, z_1 \le 10$). The second line of the input contains three integers vx_1, vy_1 and vz_1 ($-10 \le vx_1, vy_1, vz_1 \le 10$, at least one of them is not zero). The third line of the input contains three floating-point numbers x_2, y_2 and z_2 ($-10 \le x_2, y_2, z_2 \le 10$). The fourth line of the input contains three floating-point numbers vx_2, vy_2 and vz_2 ($-10 \le vx_2, vy_2, vz_2 \le 10$, at least one of them is not zero). The fifth line of the input contains an integer r_0 ($1 \le r_0 \le 10$).

Output

In the first line of output, print the number n of segments in your trajectory ($0 \le n \le 100$). In the next n lines, print the descriptions of the segments.

For each segment, first print 1 if it's a line segment, or 2 if it's a circle segment, followed by the floating-point coordinates of the ending point of the segment (the starting point will be taken from the ending point of the previous segment, or from the overall starting point for the first segment). For a circle segment, additionally print the floating-point coordinates of the center of the circle.

All coordinates you print must not exceed 1000 by absolute value. The judging program will use at least 'double' floating-point precision for all calculations. For each inflection point, the judging program will check that the angle between the directions of the two segments does not exceed 10^{-6}. The judging program will also check that the ending point is not farther than 10^{-6} from the required ending point, that the angle between the starting direction and the required starting direction does not exceed 10^{-6}, and that the angle between the ending direction and the required ending direction does not exceed 10^{-6}. For each circular segment, the judging program will check that the distances from the center of the circle to each endpoint do not differ by more than 10^{-6}, and that those distances are not less than $r_0 - 10^{-6}$.

Note that it is dangerous to output very short line segments, as rounding errors can contribute a lot to determining their direction, and thus make them fail the test above. In particular, a zero-length line segment will always result in a wrong answer.

Examples

standard input	standard output
1 2 3 0 1 0 3 7 3 1 0 0 2	2 1 1.0 5.0 3.0 2 3.0 7.0 3.0 3.0 5.0 3.0
1 2 3 4 5 6 1 2 3 4 5 6 1	0