TASK 1.1. COINS

You are given M positive integer values and one of these values is 1. You are also given an unlimited number of coins of each of these values. Consider the following problem: A certain amount of money S should be paid by a minimal number of coins with the given values.

It is known that this problem is solvable in some cases by the following greedy algorithm: Find the greatest value of a coin that is less than or equal to *S*, and then subtract it from *S*. Continue doing the same, until the value of S becomes zero. The number of coins, which is used by the algorithm to reduce *S* to zero seems to be the minimal number of coins needed at all.

In many cases the above assertion is true, but on some sets of values and for some *S* the greedy algorithm described above does not compute the optimal solution. For example, on a set of values {1, 2, 5, 7, 10} and for $S = 14$, the greedy algorithm gives a solution with 3 coins (14=10+2+2), while the obvious minimal solution is with 2 coins (14 = 7 + 7).

A question arises – for which sets of values the greedy algorithm does not produce an optimal solution. Write a program **COINS,** that for a given set of coins' values should examine if there exists an amount *S*, which is representable by a smaller number of coins than the greedy algorithm says.

The program has to read the data from the **standard input**. On the first line of the input, the number *M* of different coins' values is given ($1 < M < 100$). On the second line of the input, the values $a_1, a_2, \ldots, a_M$ are given ($1=a_1 < a_2 < \ldots < a_M \leq 7000000$), separated by a single space. On the third line of the input two integers *x* and *y* are given ($0 < x < y \leq 7000000$), separated by a space.

The program should print on the first line of the **standard output** a value *S*, $x \leq S \leq y$, for which the greedy algorithm fails to give the optimal solution. On the second line, the program should print the numbers $b_1, b_2, \ldots, b_M$ of coins (separated by a space), corresponding to the different values (in the same order as given in the input) to represent the amount of *S*, i.e. $S = a_1b_1 + a_2b_2 + \ldots + a_M b_M$. The total number of used coins should be less than the number of coins obtained by the greedy algorithm. If there exists more than one solution, your program should print any of them.

The input data always guarantee the presence of at least one S for which the described greedy algorithm fails.

EXAMPLE

Input	Output
5 1 2 5 7 10 1 100	14 0 0 0 2 0