Codeforces Beta Round 10

A. Power Consumption Calculation

1 second, 256 megabytes

Tom is interested in power consumption of his favourite laptop. His laptop has three modes. In normal mode laptop consumes P_1 watt per minute. T_1 minutes after Tom moved the mouse or touched the keyboard for the last time, a screensaver starts and power consumption changes to P_2 watt per minute. Finally, after T_2 minutes from the start of the screensaver, laptop switches to the "sleep" mode and consumes P_3 watt per minute. If Tom moves the mouse or touches the keyboard when the laptop is in the second or in the third mode, it switches to the first (normal) mode. Tom's work with the laptop can be divided into n time periods $[l_1, r_1], [l_2, r_2], ..., [l_n, r_n]$. During each interval Tom continuously moves the mouse and presses buttons on the keyboard. Between the periods Tom stays away from the laptop. Find out the total amount of power consumed by the laptop during the period $[l_1, r_n]$.

Input

The first line contains 6 integer numbers $n, P_1, P_2, P_3, T_1, T_2$ ($1 \le n \le 100, 0 \le P_1, P_2, P_3 \le 100, 1 \le T_1, T_2 \le 60$). The following n lines contain description of Tom's work. Each i-th of these lines contains two space-separated integers l_i and r_i ($0 \le l_i < r_i \le 1440, r_i < l_{i+1}$ for $i < n$), which stand for the start and the end of the i-th period of work.

Output

Output the answer to the problem.

input
1 3 2 1 5 10 0 10
output
30

input
2 8 4 2 5 10 20 30 50 100
output
570

B. Cinema Cashier

1 second, 256 megabytes

All cinema halls in Berland are rectangles with K rows of K seats each, and K is an odd number. Rows and seats are numbered from 1 to K. For safety reasons people, who come to the box office to buy tickets, are not allowed to choose seats themselves. Formerly the choice was made by a cashier, but now this is the responsibility of a special seating program. It was found out that the large majority of Berland's inhabitants go to the cinema in order to watch a movie, that's why they want to sit as close to the hall center as possible. Moreover, a company of M people, who come to watch a movie, want necessarily to occupy M successive seats in one row. Let's formulate the algorithm, according to which the program chooses seats and sells tickets. As the request for M seats comes, the program should determine the row number x and the segment $[y_l, y_r]$ of the seats numbers in this row, where $y_r - y_l + 1 = M$. From all such possible variants as a final result the program should choose the one with the minimum function value of total seats remoteness from the center. Say, $x_c = \lceil \frac{k}{2} \rceil, y_c = \lceil \frac{k}{2} \rceil$ — the row and the seat numbers of the most "central" seat. Then the function value of seats remoteness from the hall center is $\sum_{y=y_l}^{y_r} |x - x_c| + |y - y_c|$. If the amount of minimum function values is more than one, the program should choose the one that is closer to the screen (i.e. the row number x is lower). If the variants are still multiple, it should choose the one with the minimum y_l. If you did not get yet, your task is to simulate the work of this program.

Input

The first line contains two integers N and K ($1 \le N \le 1000, 1 \le K \le 99$) — the amount of requests and the hall size respectively. The second line contains N space-separated integers M_i from the range $[1, K]$ — requests to the program.

Output

Output N lines. In the i-th line output «-1» (without quotes), if it is impossible to find M_i successive seats in one row, otherwise output three numbers x, y_l, y_r. Separate the numbers with a space.

input
2 1 1 1
output
1 1 1 -1

input
4 3 1 2 3 1
output
2 2 2 1 1 2 3 1 3 2 1 1

C. Digital Root

2 seconds, 256 megabytes

Not long ago Billy came across such a problem, where there were given three natural numbers A, B and C from the range $[1, N]$, and it was asked to check whether the equation $AB = C$ is correct. Recently Billy studied the concept of a digital root of a number. We should remind you that a digital root $d(x)$ of the number x is the sum $s(x)$ of all the digits of this number, if $s(x) \le 9$, otherwise it is $d(s(x))$. For example, a digital root of the number 6543 is calculated as follows: $d(6543) = d(6 + 5 + 4 + 3) = d(18) = 9$. Billy has counted that the digital root of a product of numbers is equal to the digital root of the product of the factors' digital roots, i.e. $d(xy) = d(d(x)d(y))$. And the following solution to the problem came to his mind: to calculate the digital roots and check if this condition is met. However, Billy has doubts that this condition is sufficient. That's why he asks you to find out the amount of test examples for the given problem such that the algorithm proposed by Billy makes mistakes.

Input
The first line contains the only number N ($1 \le N \le 10^6$).

Output
Output one number — the amount of required A, B and C from the range $[1, N]$.

input
4
output
2

input
5
output
6

For the first sample the required triples are $(3, 4, 3)$ and $(4, 3, 3)$.

D. LCIS

1 second, 256 megabytes

This problem differs from one which was on the online contest.

The sequence $a_1, a_2, ..., a_n$ is called increasing, if $a_i < a_{i+1}$ for $i < n$.

The sequence $s_1, s_2, ..., s_k$ is called the subsequence of the sequence $a_1, a_2, ..., a_n$, if there exist such a set of indexes $1 \le i_1 < i_2 < ... < i_k \le n$ that $a_{i_j} = s_j$. In other words, the sequence s can be derived from the sequence a by crossing out some elements.

You are given two sequences of integer numbers. You are to find their longest common increasing subsequence, i.e. an increasing sequence of maximum length that is the subsequence of both sequences.

Input
The first line contains an integer n ($1 \le n \le 500$) — the length of the first sequence. The second line contains n space-separated integers from the range $[0, 10^9]$ — elements of the first sequence. The third line contains an integer m ($1 \le m \le 500$) — the length of the second sequence. The fourth line contains m space-separated integers from the range $[0, 10^9]$ — elements of the second sequence.

Output
In the first line output k — the length of the longest common increasing subsequence. In the second line output the subsequence itself. Separate the elements with a space. If there are several solutions, output any.

input
7 2 3 1 6 5 4 6 4 1 3 5 6
output
3 3 5 6

input
5 1 2 0 2 1 3 1 0 1
output
2 0 1

E. Greedy Change

2 seconds, 256 megabytes

Billy investigates the question of applying greedy algorithm to different spheres of life. At the moment he is studying the application of greedy algorithm to the problem about change. There is an amount of n coins of different face values, and the coins of each value are not limited in number. The task is to collect the sum x with the minimum amount of coins. Greedy algorithm with each its step takes the coin of the highest face value, not exceeding x. Obviously, if among the coins' face values exists the face value 1, any sum x can be collected with the help of greedy algorithm. However, greedy algorithm does not always give the optimal representation of the sum, i.e. the representation with the minimum amount of coins. For example, if there are face values $\{1, 3, 4\}$ and it is asked to collect the sum 6, greedy algorithm will represent the sum as $4 + 1 + 1$, while the optimal representation is $3 + 3$, containing one coin less. By the given set of face values find out if there exist such a sum x that greedy algorithm will collect in a non-optimal way. If such a sum exists, find out the smallest of these sums.

Input
The first line contains an integer n ($1 \le n \le 400$) — the amount of the coins' face values. The second line contains n integers a_i ($1 \le a_i \le 10^9$), describing the face values. It is guaranteed that $a_1 > a_2 > ... > a_n$ and $a_n = 1$.

Output
If greedy algorithm collects any sum in an optimal way, output -1. Otherwise output the smallest sum that greedy algorithm collects in a non-optimal way.

input
5 25 10 5 2 1
output
-1

input
3 4 3 1
output
6