2018 Team Formation Test

Task Overview

ID	Name	Time Limit	Memory Limit	Subtasks
T181	Pig's CD	2.000 s	256 MB	10 + 10 + 23 + 35 + 15 + 7
T182	Cave Exploration	1.000 s	256 MB	13 + 14 + 17 + 11 + 8 + 22 + 15
T183	Exam Anti-Cheat	Output-Only		10 + 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10
T184	Hackerland's Got Talent	1.000 s	256 MB	7 + 9 + 18 + 11 + 27 + 14 + 14

Notice:
Unless otherwise specified, inputs and outputs shall follow the format below:

- One space between a number and another number or character in the same line.
- No space between characters in the same line.
- Each string shall be placed in its own separate line.
- Outputs will be automatically fixed as follows: Trailing spaces in each line will be removed and an end-of-line character will be added to the end of the output if not present. All other format errors will not be fixed.

C++ programmers should be aware that using C++ streams (`cin` / `cout`) may lead to I/O bottlenecks and substantially lower performance.

For some problems 64-bit integers may be required. In Pascal it is `int64`. In C/C++ it is `long long` and its token for `scanf` /`printf` is `%lld`.

All tasks are divided into subtasks. You need to pass all test cases in a subtask to get points.

T181 - PIG'S CD

Time Limit: 2.000 s / Memory Limit: 256 MB

Pepper Pig loves music, and she loves listening to CDs more than listening to songs on the Internet. She has a long CD cabinet at home, with N CDs placed in a row. She numbered the CDs from 1 to N (left to right). The i-th CD has a stunning index S_i and the playing time T_i units (where T_i are positive integers).

Pepper Pig has $\frac{N(N+1)}{2}$ unique playlists. Each playlist can be represented by a pair of integers (L, R), with the property that $1 \leq L \leq R \leq N$, meaning the playlist plays the CDs from the L-th one to the R-th one, with total playing time of $T_L + T_{L+1} + \cdots + T_{R-1} + T_R$ units.

Pepper Pig defines the stunning power of a playlist (L, R) as the greatest common divisor of its S_i. In other words, if a playlist plays the L-th CD to the R-th CD, the stunning power is $\gcd(S_{L..R})$. Pepper Pig reminds you that, $\gcd(S_{L..R})$ is the largest positive integer that divides each of the $S_L, S_{L+1}, \ldots, S_{R-1}, S_R$. For example, $\gcd(6, 10, 15) = 1$, while $\gcd(6, 12, 15) = 3$.

Pepper Pig gets bored after finishing all the exams. She has a holiday of Q days. On the k-th day, she will choose a positive integer x_k, and play all the playlists that have their stunning powers equal to x_k once. She wants to know, for each day, what is the total playing time of the playlists she plays.

INPUT

The first line contains an integer N.
The second line contains N integers: $S_1, S_2, \ldots, S_N$
The third line contains N integers: $T_1, T_2, \ldots, T_N$
The forth line contains an integer Q.
Each of the following Q lines contains one positive integer: x_k

OUTPUT

Output Q lines, the k-th line contains an integer indicating the total playing time of the playlists she plays on the k-th day of holiday.

SAMPLE TESTS

	Input	Output
1	4 10 6 12 15 10 6 12 15 8 1 2 3 4 5 6 7 8	43 44 60 0 0 24 0 0

On the third day of Pepper Pig's holiday, $x_3 = 3$. She will play the playlists: $(2, 4)$ and $(3, 4)$.
Playlist $(2, 4)$ has total playing time of $6 + 12 + 15 = 33$.
Playlist $(3, 4)$ has total playing time of $12 + 15 = 27$.
Therefore, on the third day, Pepper Pig plays the CDs for total time $33 + 27 = 60$.

2	10 1 1 1 1 1 1 1 1 1 1 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1 1	220000

SUBTASKS

For all cases:
$1 \le N, Q \le 3 \times 10^5$
$1 \le S_i, x_k \le 10^{18}$
$1 \le T_i \le 10^3$

	Points	Constraints
1	10	$1 \le N, Q \le 100$ $1 \le S_i \le 1000$
2	10	$1 \le N, Q \le 5000$ $1 \le S_i \le 10^6$
3	23	$1 \le N, Q \le 10^5$ $1 \le S_i \le 2$
4	35	$1 \le N, Q \le 10^5$ $1 \le S_i \le 1000$
5	15	$1 \le S_i \le 10^6$
6	7	No additional constraints

T182 - CAVE EXPLORATION

Time Limit: 1.000 s / Memory Limit: 256 MB

A cave in Hackerland, believed to have more than 2000 years of history, is discovered recently. Archaeologists are really interested in exploring the structure of the cave.

According to recorded history, the cave has N rooms with $N-1$ corridors, while the exact value of N is unknown. The rooms have the shape of circle. The corridors each form a connection between two distinct rooms, such that it is possible to visit between any rooms. In terms of graph theory, the rooms and corridors form a tree.

As the cave is full of danger, for safety reasons, it is suggested to use the robot, Robo, for exploring the structure of the cave. Unfortunately, Robo cannot distinguish different rooms and corridors because Robo has only a sensor on its left arm that detects corridors. It cannot tell how many corridors are connected to the room. As this characteristic increases the difficulty of the task, Robo is designed in a way that allows it to place and sense flags in the rooms. However, the flags are identical and Robo has a limited number of flags. It is known that there are no flags in the cave initially.

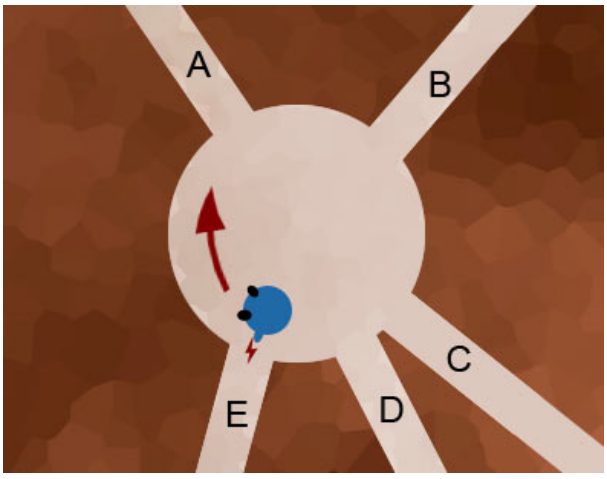
YOUR TASK

Your task is to implement a subprogram `explore` to control Robo, and find out the structure of the cave. To be more specific, you should find out:

- the number of rooms, i.e. the value of N
- the details of the corridors, that is, which two rooms each corridor connects.

Robo is initially placed at one of the rooms, with the sensor on its left arm pointing at a corridor of the room. To explore the cave, Robo only supports the following five exploration procedures / functions:

- procedure `chooseCorridor(x)`
  - x should be an integer between 1 and 10^9 (inclusive).
  - Robo will walk along the wall of the room, in clockwise direction. Robo does not record how long it walks, it only records the number of corridors its left arm has sensed. After sensing x corridors (including the one initially pointing at), Robo will enter the next corridor it senses, and then walk through the corridor until it reaches another room. When Robo reaches another room, its left arm will point at the corridor that it just comes from.
  - For instance, if Robo is at a room with five corridors (Robo does not know the room has five corridors) as shown:



    - He will enter corridor A if $x = 1, 6, 11, \ldots$.
    - He will enter corridor B if $x = 2, 7, 12, \ldots$.
    - He will enter corridor C if $x = 3, 8, 13, \ldots$.
    - He will enter corridor D if $x = 4, 9, 14, \ldots$.
    - He will enter corridor E if $x = 5, 10, 15, \ldots$.
- procedure `placeFlag()`
  - Places a flag in the current room. Nothing happens if Robo had no flags left. Robo may place 0 or more flags in a room.

- procedure `removeFlag()`
  - Removes one flag from the current room. Robo can reuse the flag later. Nothing happens if there were no flags in the room.
- function `countFlags()`
  - Robo will detect the number of flags in a room. The function returns an integer: the number of flags detected.
- function `detectDeadEnd()`
  - Only in some subtasks, Robo is also equipped with a sensor that detects whether the room is a dead end. A room is a dead end if and only if there is only one corridor connecting the room. If the room is a dead end, the function returns `true`, and `false` otherwise.

Once you are confident about the structure of the cave, you are allowed to number the rooms in any way you want, with the following restrictions:

- The rooms should be numbered as integers between 1 and N (inclusive).
- No two rooms share the same number.

Then, you can report the structure of the cave by calling the following procedures:

- procedure `reportRooms(N)`
  - You should call this procedure **EXACTLY ONCE**, where N should be the number of rooms in the cave.
  - Once called, Robo must not make any more calls to the exploration procedures / functions.
- procedure `reportCorridor(u, v)`
  - You should call this procedure **EXACTLY** $N-1$ **TIMES** after calling `reportRooms(N)`. Each call reports that there is a corridor connecting room u and room v, where $1 \leq u, v \leq N$ and $u \neq v$.

Your subprogram `explore` should return after reporting.

Your answer (i.e. the reported structure of the cave) is considered to be correct if and only if the structure is identical to the actual cave, i.e. we can get the same set of corridors by the permutating the room numbers.

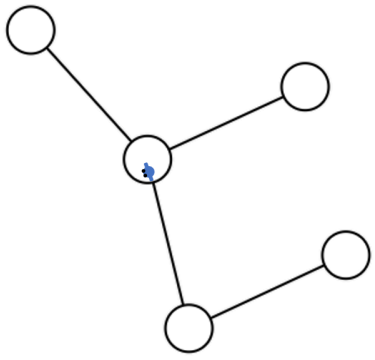
IMPLEMENTATION

You should implement a procedure `explore(maxN, L, F, D)` that makes calls to the seven grader procedures and functions mentioned in the section "Your Task". The meaning of the parameters to the `explore` procedure are:

- integer $maxN$: The number of rooms in the cave will be between 2 and $maxN$ (inclusive).
- integer L: The maximum number of calls you may make to **each** of the 5 exploration procedures / functions.
- integer F: The number of flags Robo has initially.
- boolean D: True iff Robo is allowed to use the `detectDeadEnd` function.

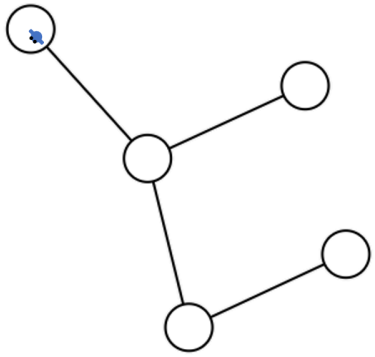
SAMPLE RUN

Suppose `explore(100, 60000, 10000, true)` is called, and Robo is placed in a cave as shown: (black lines and circles represent the corridors and the rooms respectively)



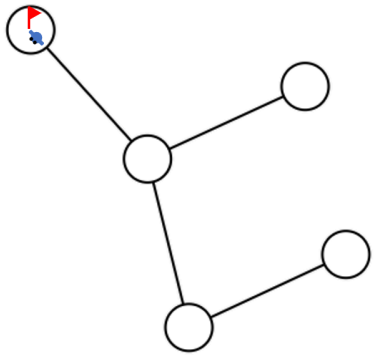
Function Call	Returns	Explanation
`detectDeadEnd()`	`false`	The room Robo currently at has more than one corridor.
`chooseCorridor(10)`		

After the function call:



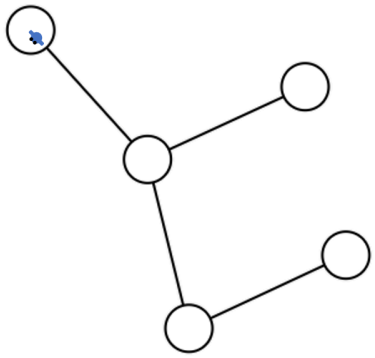
`countFlags()`	`0`	There are no flags in the room Robo currently at.
`detectDeadEnd()`	`true`	The room Robo current at has only one corridor.
`placeFlag()`		

After the function call:



`countFlags()`	`1`	There is a flag in the room Robo currently at.
`removeFlag()`		

After the function call:

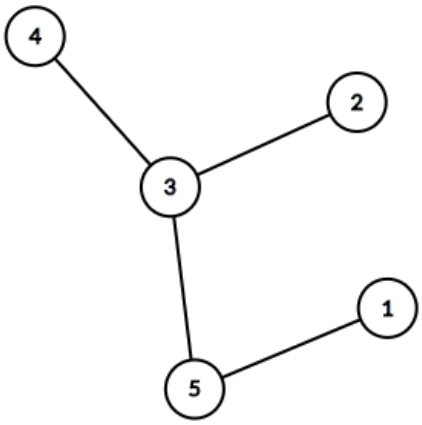


`countFlags()`	`0`	There are no flags in the room Robo currently at.
`removeFlag()`		As no flags are in the room Robo currently at, nothing happens.

...
(some function calls were omitted)
...

`reportRooms(5)`	
`reportCorridor(2, 3)`	
`reportCorridor(5, 3)`	
`reportCorridor(3, 4)`	
`reportCorridor(1, 5)`	
`explore() returns`	

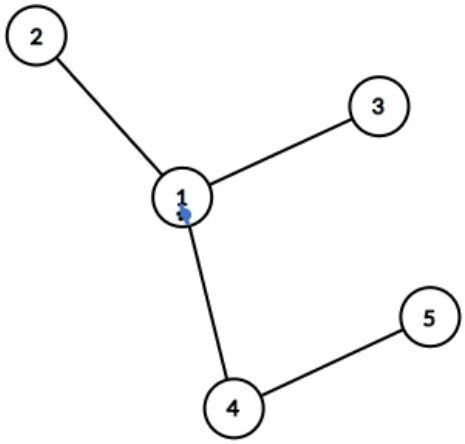
With these five calls, Robo reports that the cave structure is as:



SAMPLE TESTS

	Input	Output
1	5 1 2 1 3 1 4 4 5 1 4 100 60000 10000 1	Correct

The input represents the following initial situation:



SUBTASKS

	Points	Constraints
1	13	$maxN = 20$, $L = 60000$, $F = 10000$, $D = \text{true}$ Also, the structure of the cave is in a shape of "star". In other words, there exists exactly $N - 1$ rooms each with one corridor connected, and the remaining room has $N - 1$ corridors connected.
2	14	$maxN = 20$, $L = 60000$, $F = 10000$, $D = \text{true}$ Also, the structure of the cave is either in a shape of "star" or in a shape of "chain". In other words, at least one of the followings is true: • there exists exactly $N - 1$ rooms each with one corridor connected, and the remaining room has $N - 1$ corridors connected. • there exists exactly two rooms each with one corridor connected, and $N - 2$ rooms each with two corridors connected.
3	17	$maxN = 100$, $L = 60000$, $F = 10000$, $D = \text{true}$
4	11	$maxN = 100$, $L = 60000$, $F = 1000$, $D = \text{true}$
5	8	$maxN = 100$, $L = 60000$, $F = 100$, $D = \text{true}$
6	22	$maxN = 100$, $L = 60000$, $F = 2$, $D = \text{true}$
7	15	$maxN = 100$, $L = 30000$, $F = 2$, $D = \text{false}$

IMPORTANT: Each subtask includes all previous subtasks. In other words, for each submission, it cannot pass subtask $i + 1$ if it fails any of the subtasks $1, 2, \ldots, i$

SAMPLE GRADER

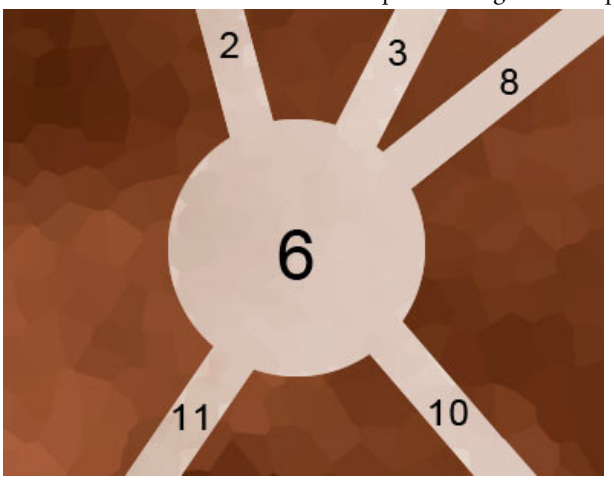
In order to test your program, you may download the sample grader files. To use the sample grader, you should implement your program under the folder corresponding to the programming language, and follow the instructions below:

Language	Source Code Filename	Compilation Command	Execution Command
Pascal	`cave.pas`	`./compile_pas.sh`	`./cave`
C	`cave.c`	`./compile_c.sh`	`./cave`
C++11	`cave.cpp`	`./compile_cpp.sh`	`./cave`

When testing your programs with the sample grader, your input should match the format and constraints from the task statement, otherwise, unspecified behaviors may occur. The sample grader reads the input in the following format:

- line 1: N
- line $1+i$ (for all $1 \le i < N$): a_i b_i
- line $N+1$: S T
- line $N+2$: $maxN$ L F D

Here, a_i and b_i state that the i-th corridor connects room a_i and b_i. Robo will be initially placed at room S, with its left arm pointing at the corridor connecting room T. For each room, the corridors are ordered by the room numbers they connect, in clockwise direction. Here is an example showing how sample grader arranges the corridors:



Line $N+2$ indicates the parameters when calling your procedure `explore`. Please note that you should enter `1` or `0` for D (representing `true` and `false` respectively).

The sample grader outputs `Correct`, if your program correctly find out the structure of the cave. Otherwise, it outputs `Wrong Answer`, followed by a message suggesting what might be done incorrectly. The sample grader also prints the function calls in the standard error stream.

To read from file, you may use: `./cave < input.txt`
To print the function calls to file, you may use: `./cave 2> calls.txt`
To read from file and print the function calls to file, you may use: `./cave < input.txt 2> calls.txt`

T183 - EXAM ANTI-CHEAT

Output-Only

Under the education system in Hackerland, student hackers will take a public exam when they complete their informatics courses.

Alice, as one of the examination center supervisors, finds that the student hackers love cheating by peeking at others' exam paper. Although there are different versions of exam paper, student hackers may still be able to cheat if there are someone having the same version of exam paper nearby. Therefore, she wants to distribute the exam papers in a way that can increase the difficulty of cheating: maximizing the minimum distance between two students having the same version of exam paper.

Formally, there are N student hackers attending the examination. The seat of each examinee corresponds to a point in a 2D Cartesian coordinate plane. The seats have **integral** coordinates (x_i, y_i) and no two seats have the same coordinates.

There are V versions of exam paper, each with unlimited supply. Each examinee must receive **exactly one** version of exam paper.

Alice wants to distribute the exam papers to make the minimum distance between two students having the same version of exam paper as large as possible. Can you do this for her?

Note: The "distance" in this problem refers to Euclidean distance. In other words, the distance between two points is $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$

This is an output-only task consisting of 10 independent test cases in separate input files. You are only required to submit the outputs that correspond to the given input files.

INPUT

The first line consists two integers: N -- the number of examinee, and V -- the number of versions of exam paper.

Each of the next N lines contains two integers x_i and y_i, the coordinates of the i-th student's seat. $(0 \leq x_i, y_i \leq 1000)$

No two seats have the same coordinates. For your convenience, the coordinates have been sorted by x, then by y.

OUTPUT

Output a string of length N. The i-th character indicates the version of exam paper the i-th student will receive.

The versions of exam paper should be represented by the first V uppercase letters: `A`, `B`, ...

To read from file and write to file, you may either use file I/O or input redirection: `./programname < examX.in > examX.out`

During contest, it is also possible to submit (upload) a single output file instead of a zip file. The filename must match one of the output filenames (case-sensitive).

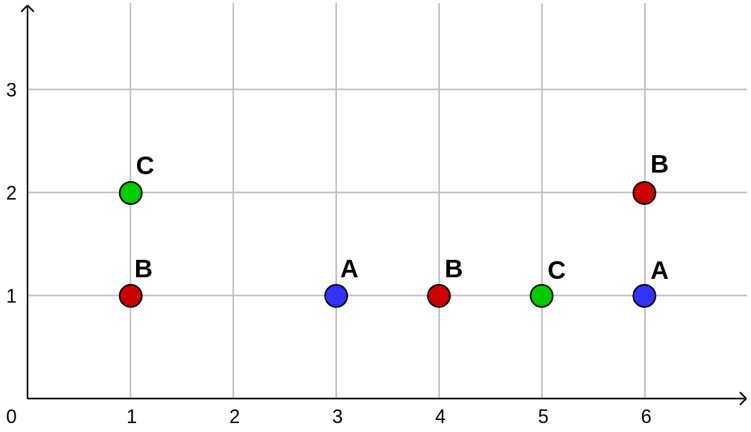
Output-Only Task Information

This is an output-only task. Please place all the output files in the root of a zip file. Do not place them in a subfolder. Windows/*nix line endings will be corrected automatically. Maximum size of the zip file is 2^{24} bytes (16MB).

Output filename requirements: `exam1.out` `exam2.out` `exam3.out` `exam4.out` `exam5.out` `exam6.out` `exam7.out` `exam8.out` `exam9.out` `exam10.out`

SAMPLE TESTS

	Input	Output
1	7 3 1 1 1 2 3 1 4 1 5 1 6 1 6 2	BCABCAB



In this sample input/output, the minimum distance between two students having the same version of exam paper $M = \sqrt{2^2 + 1^2} = \sqrt{5} \approx 2.236$ (students sitting at $(4, 1)$ and $(6, 2)$).

SCORING

Each of the 10 test cases carries 10 points.

If your arrangement does not satisfy the conditions in the task statement, you score 0% for the case.

If your arrangement satisfies the conditions in the task statement, your score for the case is calculated according to the values of D and T as follows. Let M be the minimum distance between two students having the same version of exam paper in your arrangement.

- If $M \geq T$, you score 100%.
- If $M < T$, you score $10 \times 10^{(M-D)/(T-D)}$ percent. Therefore, if $M = D$, you score $10 \times 10^0 = 10$ percent. It is guaranteed that $T - D > 0$.

Here, the values of D and T are defined as follows:

- D is the minimum distance between two students in the seating plan (therefore $D = 1$ in the sample).
- T is the target distance between two students having the same version of exam paper (as shown in the table below). It is guaranteed that there exists an arrangement that achieves $M \geq T$.

TEST CASE OVERVIEW

Case	Input	Output	N	V	T
1	exam1.in	exam1.out	62	2	53.7
2	exam2.in	exam2.out	100	2	124.0
3	exam3.in	exam3.out	123	2	32.0
4	exam4.in	exam4.out	689	2	5.6
5	exam5.in	exam5.out	777	3	32.7
6	exam6.in	exam6.out	256	3	33.3
7	exam7.in	exam7.out	512	4	77.0
8	exam8.in	exam8.out	947	4	17.1
9	exam9.in	exam9.out	999	5	60.8
10	exam10.in	exam10.out	1000	5	26.4

T184 - HACKERLAND'S GOT TALENT

Time Limit: 1.000 s / Memory Limit: 256 MB

Just two weeks ago, your team of programmers lost the Hackerland Gymnastics Contest by an embarrassing score margin (most teams scored above 10^{18} and your team was not even close). You decided to go solo and participate in the inaugural Hackerland's Got Talent.

You have passed the audition easily; what comes next is N rounds of performances. The first round is for contestants to get acquainted with the audience; contestants will be scored only in the next $N-1$ rounds, i.e. rounds $2, 3, \ldots, N$.

You have prepared exactly N acts. The quality of the i-th act is given by Q_i. The higher Q_i is, the more the audience and the judges will love the i-th act.

For $2 \le i \le n$, there is an integer W_i measuring the weight of the i-th round. Weight of a round depends on many factors, like the number of audience, whether the show is live, the reputation of the judges, and random rules to make the show more entertaining.

Suppose you perform act x in round $i-1$ and act y in round i. Then you score $W_i \times \max(0, Q_y - Q_x)$ in round i, and your final score will be the sum of scores you get in rounds $2, 3, \ldots, N$.

Obviously, one cannot perform the same act twice in the show, so each of your N acts is to be performed **exactly once**.

You don't know if you are good enough to win Hackerland's Got Talent, but at least you would like to maximize your final score. Help yourself out!

INPUT

The first line of input consists of an integer N, the number of rounds (also the number of acts you have prepared).

The second line of input consists of N integers $Q_1, Q_2, \ldots, Q_N$, the quality of your acts.

The third line of input consists of $N-1$ integers $W_2, W_3, \ldots, W_N$, the weights of the rounds.

OUTPUT

Output two lines.

In the first line, output one integer, the maximum total score you can achieve.

In the second line, output a permutation $P_1, P_2, \ldots, P_N$ of $\{1, 2, \ldots, N\}$ as space-separated integers, meaning that you will perform act P_i in round i.

If there are multiple solutions, output any one of them.

SAMPLE TESTS

	Input	Output
1	3 1 10 100 2 1	198 1 3 2

You get a score of $W_2 \times \max(0, Q_3 - Q_1) = 2 \times \max(0, 100 - 1) = 198$ in round 2, then a score of $W_3 \times \max(0, Q_2 - Q_3) = 1 \times \max(0, 10 - 100) = 0$ in round 3, for a total score of 198.

2	4 6 9 3 1 10 1 10	110 3 2 4 1

Total score is $10 \times (9 - 3) + 1 \times 0 + 10 \times (6 - 1) = 110$.

3

4 6 9 3 1 1 10 1	80 3 4 2 1

Total score is $1 \times 0 + 10 \times (9 - 1) + 1 \times 0 = 80$.

SUBTASKS

For all cases: $2 \leq N \leq 60, 1 \leq Q_i, W_i \leq 1000$.

	Points	Constraints
1	7	$N = 3$
2	9	$N \leq 10$
3	18	$N \leq 18$
4	11	$W_i = 1$ for $2 \leq i \leq N$
5	27	$N \leq 36$
6	14	$N \leq 50$
7	14	No additional constraints