Task 1: Mountains (`mountains`)

Pengu the Penguin lives in Antartica with his penguin friends. Despite being flightless birds, they wish to experience the joy of soaring through the air. Pengu decides to fulfill their wishes through the power of technology - with gliders.

Luckily for Pengu, there are n mountains in the Transantartic Mountain Range. The mountains are labelled 1 to n and are arranged in a row from left to right. The i^{th} mountain has height H_i.

Pengu decides to pick 3 mountains x, y and z. He plans to build a base station on mountain y and receiving stations on mountains x and z. Penguins will glide from mountain y to either mountain x or z. To acommodate more penguins while avoiding midair collisions, mountains x and z are to the left and right of mountain y respectively. Furthermore, mountains x and z must be strictly shorter than mountain y.

Pengu is very meticulous and wants to consider all possible choices. Find the number of possible choices for (x, y, z) such that $1 \leq x < y < z \leq n$, $H_x < H_y$ and $H_y > H_z$.

Input

Your program must read from standard input.

The first line contains an integer n, the number of mountains.

The second line contains n integers, where the i^{th} integer represents the height of the i^{th} mountain H_i.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the total number of possible choices for (x, y, z).

Implementation Note

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `Mountains.java` and place your `main` function inside `class Mountains`.

Subtasks

The maximum execution time on each instance is 2.0s, and the maximum memory usage on each instance is 256MiB. For all testcases, the input will satisfy the following bounds:

- $3 \leq n \leq 3 * 10^5$
- $0 \leq H_i \leq 10^{18}$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	2	H_i is non-decreasing ($H_i \leq H_j$ for $i < j$)
2	4	$0 \leq H_i \leq 1$
3	9	$0 \leq H_i \leq 99$
4	36	$n \leq 500$
5	28	$n \leq 10^4$
6	9	$0 \leq H_i \leq 10^5$
7	12	-

Sample Testcase 1

This testcase is valid for all subtasks except subtask 1.

Input	Output
5 0 1 1 0 1	2

Sample Testcase 1 Explanation

There are 2 valid triplets: (1, 2, 4) and (1, 3, 4).

Sample Testcase 2

This testcase is valid for subtasks 4, 5, 6 and 7 only.

Input	Output
6 500 20 900 0 900 70	7

Sample Testcase 2 Explanation

There are 7 valid triplets: (1, 3, 4), (1, 3, 6), (1, 5, 6), (2, 3, 4), (2, 3, 6), (2, 5, 6) and (4, 5, 6).

Task 2: Visiting Singapore (`visitingsingapore`)

Singapore organizes many events. Suppose one event is organized in Singapore per day, where $\Sigma = \{1, 2, \ldots, K\}$ represents the set of possible events. Attending event i will increase your happiness by $V[i]$. Let $S[1], \ldots, S[n]$ be the list of events organized in n days in order. (Note that the same events may appear multiple times in the sequence.)

You want to attend m events $T[1], \ldots, T[m]$ in this order. (Note that the same events may appear multiple times in T.) So, you decide to fly to Singapore on the ith day and leave Singapore on the jth day. It is also possible that you do not fly to Singapore at all.

During your visit, you try to attend events $T[1], \ldots, T[m]$ in order.

When you attend the event $T[i]$, your happiness is increased by $V[T[i]]$. When you skip events $T[p], T[p+1], \ldots, T[q]$, your happiness is reduced by $A + (q - p + 1)B$ where A and B are some given parameters.

In addition, if during your stay you do not attend events for d consecutive days, your happiness is reduced by $A + dB$. More formally, if you attend the events $S[p], S[q]$ where $p + 2 \leq q$ but none of the events in between, your happiness is reduced by $A + (q - p - 1)B$.

You want to maximize your happiness. Can you compute the maximum happiness?

Input

Your program must read from standard input.

The input starts with a line with five integers K, n, m, A and B, where K, n, and m are positive integers and A and B are negative integers.

The second line contains K positive integers where the ith integer represents $V[i]$, the happiness of the ith event.

The third line contains n integers, where the ith integer represents $S[i]$, which is between 1 and K.

The fourth line contains m integers, where the ith integer represents $T[i]$, which is between 1 and K.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the total happiness in an optimal schedule.

Implementation Note

If you are implementing your solution in Java, please name your file `VisitingSingapore.java` and place your `main` function inside `class VisitingSingapore`.

Subtasks

The maximum execution time on each instance is 2.0s, and the maximum memory usage on each instance is 256MiB. For all testcases, the input will satisfy the following bounds:

- $1 \leq K \leq 1000$
- $1 \leq n, m \leq 5000$
- $-100 \leq A, B \leq 0$
- $1 \leq V[i] \leq 100$ for all i.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	4	$K = 1, m \leq n \leq 10^3$
2	6	$K = 1, n < m \leq 10^3$
3	12	$A = B = 0$
4	7	$A = 0$
5	8	$B = 0$
6	13	$n, m < 100$
7	50	-

Sample Testcase 1

This testcase is valid for subtasks 1, 6 and 7.

Input	Output

`1 5 3 -5 -4`
`10`
`1 1 1 1 1`
`1 1 1`
	`30`

Sample Testcase 1 Explanation

In this example, $K = 1$, $n = 5$, $m = 3$, $A = -5$ and $B = -4$. Since there is only one type of event and $m \leq n$, one possible optimal solution is to go to Singapore on the first day and leave Singapore on the mth day.

Since the happiness for the task is 10 and $m = 3$, the optimal happiness is 30.

Sample Testcase 2

This testcase is valid for subtasks 2, 6 and 7.

Input	Output
1 3 5 -10 -5 10 1 1 1 1 1 1 1 1	10

Sample Testcase 2 Explanation

Since there is only one type of event and $n > m$, A possible optimal solution is to go to Singapore on the first day and leave Singapore on the nth day. Also, we need to skip events $T[m - n + 1], \ldots, T[n]$.

Since the happiness for the task is 10, $n = 3$ and $m = 5$, the plan is to try $T[1], T[2], T[3]$ for three days; then skip $T[4]$ and $T[5]$. The gain in happiness for the first three events is $10*3 = 30$. The reduction in happiness for the last two events is $A + 2B = -10 + 2(-5) = -20$. In total, the optimal happiness is 10.

Sample Testcase 3

This testcase is valid for subtasks 3, 4, 5, 6 and 7.

Input	Output
4 7 4 0 0 1 2 3 4 3 1 2 1 4 1 1 1 2 3 4	7

Sample Testcase 3 Explanation

The optimal solution is to try $S[2] = 1$, $S[3] = 2$ and $S[5] = 4$. The score is $1 + 2 + 4 = 7$.

Sample Testcase 4

This testcase is valid for subtasks 4, 6 and 7.

Input	Output
4 8 4 0 -3 1 2 3 4 3 1 2 1 1 4 1 1 1 2 3 4	-1

Sample Testcase 4 Explanation

The optimal solution is to try $S[5] = 1$ and $S[6] = 4$. The score is $1 + 4 - (2 * 3) = -1$.

Sample Testcase 5

This testcase is valid for subtasks 5, 6 and 7.

Input	Output
4 8 4 -3 0 1 2 3 4 3 1 2 1 1 4 1 1 1 2 3 4	2

Sample Testcase 5 Explanation

The optimal solution is to try $S[5] = 1$ and $S[6] = 4$. The entries $T[2]$ and $T[3]$ are deleted, which costs -3. The score is $1 + 4 - 3 = 2$.

Sample Testcase 6

This testcase is valid for subtasks 6 and 7.

Input	Output
6 10 6 -2 -1 1 2 3 4 5 6 3 1 5 2 6 1 5 1 1 4 1 2 3 4 5 6	4

Sample Testcase 6 Explanation

The optimal solution arrives at Singapore on day 2 and leave Singapore on day 5. The solution tries $S[2] = 1$, $S[3] = 5$ and $S[5] = 6$. We skip $T[2]$ to $T[4]$. So, the reduction of happiness is $-2 + 3 * (-1) = -5$. We skip day 4. So, the reduction of happiness is $-2 + (-1) = -3$. The score is $1 + 5 + 6 - 5 - 3 = 4$.

Task 3: Solar Storm (`solarstorm`)

Squeaky the Mouse is the captain of a spaceship, on a mission to explore the outer reaches of the solar system. His spaceship is shaped like a rod – there is a long, straight pathway along the length of the spaceship, and N modules are connected at various locations along the pathway. The modules are numbered from 1 to N, starting from the left side of the spaceship. The distances between adjacent modules need not be equal.

Here is an example of a possible configuration of the spaceship:

Each module i (where i ranges from 1 to N) contains equipment to support the operation of the spaceship, and hence has an associated positive integer value v_i that measures how important it is for the operation of the spaceship. Electrical wiring is run between every pair of adjacent modules, which allows the equipment to be controlled remotely from another module.

Alas, there is a impending solar storm quickly approaching Squeaky's spaceship! The solar storm contains a burst of electromagnetically charged particles which will damage the equipment on Squeaky's spaceship if they are not protected properly.

Luckily, Squeaky's competent crew has brought along S shields on this mission. Each shield may be deployed in any module, or left undeployed. When deployed, each shield generates a magnetic field that will deflect the electromagnetically charged particles away from the spaceship, and hence protect all equipment located in modules no more than K metres away from the shield. (Equipment located in the same module as a shield will of course be protected.)

Note that the pathway does not need to be shielded as the electrical wiring along the pathway is not susceptible to damage from the charged particles.

To ensure that as much of the spaceship remains operational after the solar storm as possible, Squeaky wants to deploy the shields optimally, i.e. to maximise the total value of protected modules. Furthermore, from any protected module, he wants it to be possible to control equipment in all protected modules, to minimise the amount of walking that the crew will need to do after the solar storm. However, due to the design of the electrical wiring on the spaceship, a module A can control equipment at some module B only if all modules between A and B (inclusive of both A and B) are operational. As Squeaky's chief engineering officer on this mission, help him to figure out the optimal placement of shields satisfying this constraint.

Input

Your program must read from standard input.

The first line contains three integers, N, S, and K, which represent the number of modules, number of shields, and the protection radius of each shield (in metres) respectively.

The second line contains $N-1$ integers. The i^{th} integer is d_i, the distance between module i and module $i+1$, in metres.

The third line contains N integers. The i^{th} integer is v_i, the value of module i.

Output

Your program must print to standard output.

The output should contain exactly two lines.

The first line should contain a single integer, T, the number of shields to deploy.

The second line should contain exactly T integers, specifying the modules in which shields should be deployed. This list may be printed in any order. If there are multiple possible ways to maximise the total value of protected modules, all of them will be accepted.

Note: You will get the "Output is invalid" response if your output does not adhere to the format above, or the integer T on the first line is not between 0 and S inclusive, or the T integers that you specify on the second line are not between 1 and N inclusive. You will get the "Damaged module blocking communication between protected modules" response if there is a damaged module that is between two protected modules, which is not allowed as per the task description above.

Implementation Note

As the input lengths for subtasks 2, 3, 4, 6, and 7 may be very large, you are recommended to use C++ with fast input routines to solve this problem. The scientific committee does not have a solution written in Java or Python that can fully solve this problem.

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `SolarStorm.java` and place your `main` function inside `class SolarStorm`.

Subtasks

The maximum execution time on each instance is 2.0s, and the maximum memory usage on each instance is 256MiB. For all testcases, the input will satisfy the following bounds:

- $1 \leq S \leq N \leq 10^6$
- $1 \leq K \leq 10^{12}$
- $1 \leq d_i \leq 10^6$ for all $1 \leq i \leq N-1$

- $1 \leq v_i \leq 10^6$ for all $1 \leq i \leq N$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	10	$S = 1$, $N \leq 10^4$, $K \leq 10^9$, $d_i \leq 10^5$ for all $1 \leq i \leq N-1$, $v_i \leq 10^5$ for all $1 \leq i \leq N$
2	7	$S = 1$, $d_i = 1$ for all $1 \leq i \leq N-1$
3	11	$S = 1$
4	8	$K = 1$, $d_i = 2$ for all $1 \leq i \leq N-1$
5	18	$N \leq 10^4$
6	16	$S \leq 50$
7	30	-

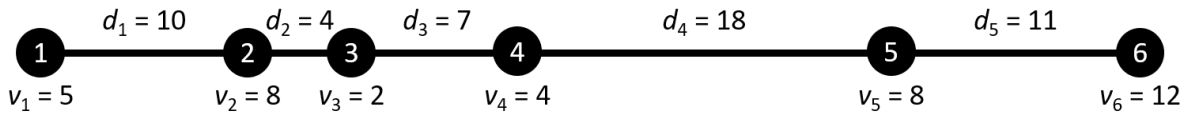
Sample Testcase 1

This testcase is valid for subtasks 5, 6, and 7.

Input	Output
6 2 7 10 4 7 18 11 5 8 2 4 8 12	2 3 5

Sample Testcase 1 Explanation

The spaceship looks like this:



The optimal locations to deploy the two shields are at modules 3 and 5. The shield at module 3 will protect modules 2, 3, and 4; the shield at module 5 will protect module 5. Notice that the pathway between modules 4 and 5 is not fully protected; this is acceptable as the electrical wiring along the pathway is not susceptible to damage.

Note that deploying the two shields at modules 3 and 6 instead is not a valid solution, because module 5 will be damaged by the solar storm, making it impossible for crew at module 6 to control equipment at modules 2, 3, or 4.

Sample Testcase 2

This testcase is valid for subtasks 5, 6, and 7.

Input	Output
6 2 38 10 4 7 18 11 5 8 2 4 8 12	1 4

Sample Testcase 2 Explanation

The spaceship has the same configuration as in sample testcase 1, but the shield radius is much larger. Deploying a single shield at module 4 is sufficient to protect all modules.

Note that there are many other acceptable solutions. Some alternative solutions could be:

- Deploy a single shield at module 3
- Deploy two shields, one at module 3 and the other at module 5
- Deploy two shields at module 3

All valid solutions that maximise the total value of protected modules will be accepted.

Sample Testcase 3

This testcase is valid for subtasks 1, 3, 5, 6, and 7.

Input	Output
6 1 12 10 4 7 18 11 5 8 2 4 8 12	1 5

Sample Testcase 3 Explanation

The spaceship has the same configuration as in sample testcase 1. Deploying the shield at module 5 will protect modules 5 and 6, which is optimal.

Note that it is also optimal to deploy the shield at module 6.

Sample Testcase 4

This testcase is valid for subtasks 1, 2, 3, 5, 6, and 7.

Input	Output
12 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 4 5 6 6 5 4 3 2 1	1 6

Sample Testcase 4 Explanation

The optimal location is to deploy the shield at module 6, which will protect modules 4, 5, 6, 7, and 8.

Note that it is also optimal to deploy the shield at module 7.

Sample Testcase 5

This testcase is valid for subtasks 4, 5, 6, and 7.

Input	Output
10 3 1 2 2 2 2 2 2 2 2 2 2 3 7 5 6 8 4 3 2 2 9	3 3 4 5

Sample Testcase 5 Explanation

The optimal locations to deploy the three shields are at modules 3, 4, and 5.