Maratona de Programação da SBC 2022

This problem set is used in simultaneous contests:
Maratona SBC de Programação SBC 2022
Tercera Fecha Gran Premio de México 2022
Gran Premio de Centroamérica 2022
Torneo Argentino de Programación 2022
The 2022 ICPC Bolivia Preliminary Contest

October 8th, 2022

Problems book

General Information

This problem set contains 14 problems; pages are numbered from 1 to 19, without considering this page. Please, verify your book is complete.

A) Program name
1) Solutions written in C/C++ and Python, the filename of the source code is not significant, can be any name.
2) Solutions written in Java, filename should be: *problem_code*`.java` where *problem_code* is the uppercase letter that identifies the problem. Remember in Java the main class name and the filename must be the same.
3) Solutions written in Kotlin, filename should be: *problem_code*`.kt` where *problem_code* is the uppercase letter that identifies the problem. Remember in Kotlin the main class name and the filename must be the same.

B) Input
1) The input must be read from *standard input*.
2) The input is described using a number of lines that depends on the problem. No extra data appear in the input.
3) When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input.
4) Every line, including the last one, ends with an end-of-line mark.
5) The end of the input matches the end of file.

C) Output
1) The output must be written to *standard output*.
2) When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output.
3) Every line, including the last one, must end with an end-of-line.

Promo:

Sociedade Brasileira de Computação

v1.0

Problem A
Finding Maximal Non-Trivial Monotones

In this problem we will be dealing with character sequences, often called *strings*. A sequence is *non-trivial* if it contains at least two elements.

Given a sequence s, we say that a chunk $s_i, \ldots, s_j$ is *monotone* if all its characters are equal, and we say that it is *maximal* if this chunk cannot be extended to left or right without losing the monotonicity.

Given a sequence composed only of characters "`a`" and "`b`", determine how many characters "`a`" occur in non-trivial maximal monotone chunks.

Input

The input consists of two lines. The first line contains a single integer N, where $1 \leq N \leq 10^5$. The second line contains a string with exactly N characters, composed only of the characters "`a`" and "`b`".

Output

Print a single line containing an integer representing the total number of times the character "`a`" occurs in non-trivial maximal monotone chunks.

Input example 1	**Output example 1**
7 abababa	0

Input example 2	**Output example 2**
7 bababab	0

Input example 3	**Output example 3**
10 aababaaabb	5

Input example 4	**Output example 4**
10 bbaababaaa	5

Problem B
Fun with Stones

Alice and Bob will play a game with 3 piles of stones. They take turns and, on each turn, a player must choose a pile that still has stones and remove a positive number of stones from it. Whoever removes the last stone from the last pile that still had stones wins. Alice makes the first move.

The i-th pile will have a random and uniformly distributed number of stones in the range $[L_i, R_i]$. What is the probability that Alice wins given that they both play optimally?

Input

The input consists of a line with 6 integers, respectively, $L_1, R_1, L_2, R_2, L_3, R_3$. For each i, $1 \leq L_i \leq R_i \leq 10^9$.

Output

Print an integer representing the probability that Alice wins modulo $10^9 + 7$.

It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$*, where* p *and* q *are integers and* $q \not\equiv 0 \ (mod\ 10^9 + 7)$*, that is, we are interested in the integer* $p \times q^{-1} \ (mod\ 10^9 + 7)$*.*

Input example 1	**Output example 1**
`3 3 4 4 5 5`	`1`

Input example 2	**Output example 2**
`4 4 8 8 12 12`	`0`

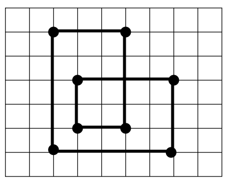
Input example 3	**Output example 3**
`1 10 1 10 1 10`	`580000005`

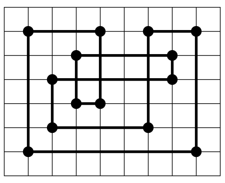
Input example 4	**Output example 4**
`5 15 2 9 35 42`	`1`

Problem C
Cutting with Lasers

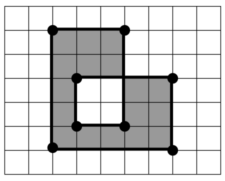
A laser cutting machine for wood sheets has a laser head that can move in only two directions, horizontal and vertical. You have been hired to be part of the testing team for the machine.

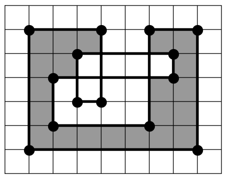
One of the tests consists of programming the machine to perform a non-empty sequence of consecutive cuts that starts and ends at the same point. Each cut in the sequence, except the first, starts at the point at which the previous cut ended. No cuts touch the edge of the sheet to be cut. Figures (a) and (b) below show two examples of cutting sequences, with respectively 8 and 14 cuts.


(a)

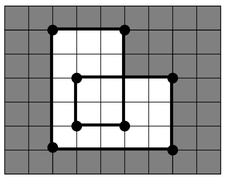

(b)

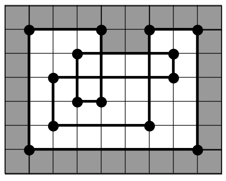
Your boss asked you to determine the area of the largest piece produced by the sequence of cuts, disregarding the piece attached to the edges of the cut sheet. That is, only the pieces contained in the polygon formed by the cut lines should be considered. Figures (c) and (d) below show respectively the largest pieces produced by the cuts of figures (a) and (b).


(c)


(d)

To illustrate, figures (e) and (f) below show the discarded piece (which contains the edges of the wood sheet) of the cut sequences of figures (a) and (b), respectively.


(e)


(f)

Input

The first line contains an integer N, the number of cuts in the sequence ($4 \leq N \leq 10^4$). The second line contains two integers X_0 and Y_0, the initial position of the laser head in the sequence of cuts ($1 \leq X_0 \leq 10^3$ and $1 \leq Y_0 \leq 10^3$). Each of the next N lines contains two integers X_i and Y_i, the final position of the cut i ($1 \leq X_i \leq 10^3$ and $1 \leq Y_i \leq 10^3$, for $1 \leq i \leq N$, and $(X_N, Y_N) = (X_0, Y_0)$). All positions given in the input are distinct, except the first and the last positions.

Output

Your program must output a single line, containing a single integer, the area of the largest piece produced by the sequence of cuts.

Input example 1	**Output example 1**

```
8
2 1
7 1
7 4
3 4
3 2
5 2
5 6
2 6
2 1
```
	`17`

Input example 2	**Output example 2**

```
14
1 1
8 1
8 6
6 6
6 2
2 2
2 4
7 4
7 5
3 5
3 3
4 3
4 6
1 6
1 1
```
	`21`

Problem D
Displacing Particles

A square has its vertices at the coordinates $(0,0), (0,2^N), (2^N,2^N), (2^N,0)$. Each vertex has an attractor. A particle is placed initially at position $(2^{N-1}, 2^{N-1})$. Each attractor can be activated individually, any number of times. When an attractor at position (i,j) is activated, if a particle is at position (p,q), it will be moved to the midpoint between (i,j) and (p,q).

Given N and a point (x,y), calculate the least number of times you have to activate the attractors so that the particle ends up at position (x,y).

Input

The input consists of a single line containing three integers N, x and y, such that $1 \leq N \leq 20$ and $0 < x, y < 2^N$.

Output

Print a single line, containing the least number of times you have to active the attractors.

Input example 1	**Output example 1**
`1 1 1`	`0`

Input example 2	**Output example 2**
`4 12 4`	`1`

Input example 3	**Output example 3**
`4 3 1`	`3`

Problem E
Eliminating Ballons

An enourmous number of balloons are floating about in the Convention Hall after the Awarding Ceremony of the ICPC Subregional Contest. The manager of the Convention Hall is angry, because he will host another event tomorrow and the ballons must be removed. Fortunately this year Carlinhos came prepared with his bow and arrows to pop the balloons.

Also, luckily, due to the air conditioning flow, the balloons are all in the same vertical plane (that is, a plane parallel to a wall), although in distinct heights and positions.

Carlinhos will shoot from the left side of the convention hall, at a chosen height, in the direction of the right side of the Convention Hall. Each arrow moves from left to right, at the height it was shot, in the same vertical plane of the baloons. When an arrow touches a balloon, the baloon pops and the arrow continues its movement to the right, at a height decreased by 1. In other words, if the arrow was at height H, after popping a balloon it continues at height $H - 1$.

Carlinhos wants to pop all balloons shooting as few arrows as possible. Can you help him?

Input

The first line of input contains an integer N, the number of balloons ($1 \leq N \leq 5 \times 10^5$). Since all balloons are in the same vertical plane, lets define that the *height* of a ballon is given in relation to the y-axis and the *position* of a balloon is given in relation to the x-axis. Balloons are numbered from 1 to N. Balloon numbers indicate theis positions, from leftmost (balloon number 1) to rightmost (balloon number N), independent of their heights. The position of balloon number i is different from the position of balloon number $i + 1$, for all i. The second line contains N integers H_i, where H_i indicates the height of balloon number i ($1 \leq H_i \leq 10^6$ for $1 \leq i \leq N$).

Output

Your program must output a single line, containing a single integer, the miminum number of arrows that Carlinhos needs to shoot to pop all balloons.

Input example 1	**Output example 1**

`5`
`3 2 1 5 4`
	`2`

Input example 2	**Output example 2**

`4`
`1 2 3 4`
	`4`

Input example 3	**Output example 3**

`6`
`5 3 2 4 6 1`
	`3`

Problem F
Multidimensional Hangman

The Multidimensional Hangman Game has very peculiar rules. In a way, it is like you are playing several games of the traditional Hangman game at the same time, with the difference that the words don't have to exist in the dictionary. If you've never played Hangman, don't worry: all the information you need is below.

In the multidimensional version of the game, there are several words on a board, initially unknown, all of the same length. At each turn in the game, you discover some characters from certain word positions (how these characters were discovered is not important for this problem). At a certain point, when only one unknown character remains in each word on the board, the game goes into the *all or nothing* phase. At this point, you must choose a word that maximizes the number of compatibilities with the words on the board. For a choosen word P, we say it is compatible with a word T on the board if all known letters in T occur in exactly the same positions in P.

Given the known information about the words on the board, you must determine which word to choose for *all or nothing* phase, which maximizes the number of compatibilities. If there is more than one solution, print the lexicographically smallest. We say that a word P is lexicographically smaller than a word Q if $P_i < Q_i$ where P_i is the i-th character of P, Q_i is the i-th character of Q and i is the smallest index such that $P_i \neq Q_i$.

Input

The first line of the input contains two integers N and C satisfying $1 \leq N \leq 10^4$ and $1 \leq C \leq 12$, indicating the number of words on the board and the length of the words it contains. Each of the next N lines contains a word of length C composed only of characters from "`a`" to "`z`" except for one of its positions, which will contain a character "`*`" , indicating that the character at that position is still unknown.

Output

Print a single line, containing, in order, a word T, of length C, and an integer M, such that M is the greatest number of compatibilities a word might have with the input words and T is the lexicographically smallest amongst the words with compatibility M.

Input example 1	**Output example 1**

`5 4`
`rat*`
`ru*d`
`rot*`
`r*ta`
`r*ta`
	`rata 3`

Input example 2	**Output example 2**

`5 4`
`bon*`
`fon*`
`n*no`
`*eto`
`*ano`
	`nano 2`

Problem G
Geometry of Triangles

Every polygon can be constructed by joining triangles. In particular, we can do this iteratively: we start with a triangle, we add a second triangle identifying one of its sides to one of the sides of the initial triangle, we add a third triangle identifying one of its sides to one of the free sides of one of the original triangles, and so on. We will only consider polygons that can be constructed in this way, where each added triangle touches (and is identified with) exactly one side of a previously positioned triangle.

Given a polygon P, let T be the set of triangles used to form it. The sides of each triangle are line segments. Let L be the set of segments that are sides of some triangle in T. Note that each element of L is one side of one or two elements of T.

Once we have a polygon positioned in the plane, in some cases we can remove some of the triangles that compose it, without changing the set L. We want to remove triangles so that the set L is maintained and the total area of the remaining triangles is minimal. Equivalently, we want to select a subset S of triangles from T such that:

1. Every element of L is the side of at least one triangle in S; and
2. The sum of the areas of the elements of S is as small as possible.

Input

The first line of the input contains an integer N, $1 \leq N \leq 10^5$ corresponding to the number of triangles in the triangulation of P. Each of the following N lines contains 6 numbers, x_1, y_1, x_2, y_2, x_3 and y_3, indicating the existence of a triangle with coordinates $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3). The triangles are given in arbitrary order. All coordinates will be integers with absolute value at most 10^6.

Output

Print the minimum area possible, respecting the conditions of the problem, with exactly one decimal place.

Input example 1	Output example 1

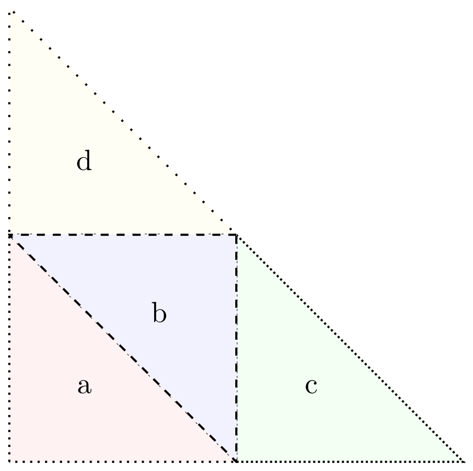
`4`
`0 0 0 10 10 0`
`10 10 0 10 10 0`
`10 10 0 10 0 20`
`10 10 20 0 10 0`
	`150.0`

Input example 2	Output example 2

`3`
`0 0 0 10 10 0`
`10 10 0 10 10 0`
`10 10 20 0 10 0`
	`150.0`

Input example 3	Output example 3

`1`
`0 0 1 0 0 1`
	`0.5`



In the figure above , the triangulations $T_1 = \{a, b, c, d\}$ and $T_2 = \{a, b, c\}$ represent, respectively, the first and second examples. Note how $S_1 = \{a, c, d\}$ is a valid subset for the first case. Triangle b is left out, but all of its sides are present in the selected triangles.

Problem H
Helping the Transit

The president of Nlogonia decided, by decree, that all the streets of Nlogonia should be one-way. Due to the lack of knowledge of elementary science, there was no proper planning for the changes. After the new system came in place, people would not be able to go to work, or would not be able to return home from work, for example. As a result, there was chaos and riots in lots of cities.

The president was impeached and the new administration of the country hired a team of scientists to solve the problem. In turn, the committee hired you, an expert in complexity of algorithms, to help them with the efficient computation of solutions.

So, for each city, you are given the *reference points* of the city, and the one-way streets, each of which connects two reference points. Your task is to determine the minimum number of one-way bridges that must be built in order to have full connectivity in the city. Each bridge should also connect two reference points.

Input

The first line of the input contains two integers, N and M ($1 \leq N \leq 10^4, 1 \leq M \leq 10^6$), where N is the number of reference points and M is the number of streets. Each one of the next M lines contains two integers, R and S, $1 \leq R, S \leq N$, $R \neq S$, that corresponds to a street connecting R to S, so that every vehicle in that street must move away from R, towards S.

Output

Your program must print a single line containing the minimum number of bridges that are necessary to make the inhabitants happy.

Input example 1	**Output example 1**

7 7
1 2
2 3
3 1
6 1
6 4
4 5
7 6
	2

Input example 2	**Output example 2**

7 7
2 1
3 2
1 3
1 6
4 6
5 4
6 7
	2

Input example 3	**Output example 3**
2 1 1 2	1

Input example 4	**Output example 4**

`3 3`
`1 2`
`2 3`
`3 1`
	`0`

Input example 5	**Output example 5**
`2 0`	`2`

Input example 6	**Output example 6**

`6 4`
`1 2`
`1 3`
`4 6`
`5 6`
	`3`

Problem I
Intercepting Information

Spies Breaching Computers (SBC) is a private digital spy agency that is developing a new device for intercepting information using electromagnetic waves, which allows spying even without physical contact with the target.

The device tries to collect information one byte at a time, this is, a sequence of 8 bits where each of them, naturally, can have a value of 0 or 1. In certain situations, due to interference from other devices, the reading cannot be done successfully. In this case, the device returns the value 9 for the corresponding bit, informing that the reading could not be performed.

In order to automate the recognition of the information the device reads, a request was made for a program that, based on the information read by the device, informs whether all bits were read successfully or not. Your task is to write this program.

Input

The input consists of a single line, containing 8 integers $N_1, N_2, N_3, N_4, N_5, N_6, N_7$ and N_8, indicating the values read by the device (N_i is `0`, `1` or `9` for $1 \leq i \leq 8$).

Output

Print a single line containing the capital letter "`S`" if all bits are read successfully; otherwise print a single line containing the capital letter "`F`", corresponding to a failure.

Input example 1	**Output example 1**
`0 0 1 1 0 1 0 1`	`S`

Input example 2	**Output example 2**
`0 0 1 9 0 1 0 1`	`F`

Problem J
Playing 23

Twenty-three is a simple card game played by kids. Like its name suggests, it is a variation of Blackjack, which is the most widely played game in casinos and gaming sites.

The game uses a deck of 52 cards, with four suits, each suit with 13 cards (ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, jack, queen and king). Card suits are not relevant. The picture cards (jack, queen and king) are worth ten points, cards with numbers are worth their number in points (for example, the 4 card is worth four points) and the ace is worth one point.

The player who has the most points wins, provided it does not exceed 23. If a player has a number of points greater than 23 we say the player *busts*.

The game rules are simple: at each game, initially the deck is shuffled, the cards are placed in a stack and each player receives two cards from the stack. All cards are dealt face up (all players see all players' cards). The next step, called *round*, is repeated as long as there are active players: a card is taken from the stack and set on the table face up. This card, called *common card*, counts to all players. If a player busts, he leaves the game. The winner is the player that in a given round has a total of 23 points (considering their two starting cards plus the common cards), or if the player is the only active player at the end of the round. Note that there can be more than one winner (whose cards add up to 23) and that there may be no winner in a match.

John and Mary are playing twenty-three. The two are the only players in the game, neither of them busted and neither of them has 23 points. Furthermore, the players score is such that the next common card may cause the game to end.

Given John and Mary initial cards and the common cards, determine the lowest possible value of a card that should be taken from the stack in the next round for Mary to win the game.

Input

The first line of input contains a single integer N ($1 \leq N \leq 8$), the number of rounds already played. Each card is described by an integer I ($1 \leq I \leq 13$). Note that the picture cards (jack, queen and king) are represented in the input by the values 11, 12 and 13 which is not how many points they are worth. The second line contains two integers, describing the two initial cards for John. The third line contains two integers, describing the two initial cards for Mary. The fourth and last line contains N integers, describing the common cards, in the order they were taken from the stack.

Output

Your program should output a single line, containing a single integer, the value of the lowest card that must be taken from the stack in the next round for Mary to win the game, or -1 if Mary cannot win the match in that next round.

Input example 1	**Output example 1**

```
1
10 5
9 10
1
```
	`3`

Input example 2	**Output example 2**

```
1
10 5
8 7
2
```
	`6`

Input example 3

```
1
9 10
10 5
1
```

Output example 3

```
4
```

Input example 4

```
2
8 4
4 1
4 4
```

Output example 4

```
5
```

Input example 5

```
8
2 1
1 1
1 2 2 2 3 3 3 3
```

Output example 5

```
-1
```

Problem K
Kalel, the Jumping Frog

Kalel is a frog that likes jumping over stones.

There are N stones in a row, numbered from 1 to N from left to right. Kalel begins at stone 1 and he wants to reach stone N.

At each move, Kalel can choose among M types of jump. The j-th jump allows him to jump from stone x to stone $x + d_j$ and costs p_j energy points. It may happen that p_j equals 0 for some j. You can assume Kalel never runs out of energy.

Given N and K, calculate in how many ways Kalel can reach stone N spending at most K energy points in total. Two ways are considered different if the sequence of jump choices is different. As this number can become very large, we are only interested in its remainder modulo 10^9 (one billion).

Input

The first line contains three integers, N, M and K ($1 \leq N \leq 10^9$, $1 \leq M \leq 10^5$, $0 \leq K \leq 400$). The next M lines contain two integers each, the numbers d_j and p_j ($1 \leq d_j \leq 10$, $0 \leq p_j \leq K$).

Output

Print a single line, containing in how many different ways Kalel can get to the rock N spending a maximum of K energy points, modulus 10^9 (one billion).

Input example 1	Output example 1

`5 3 10`
`1 3`
`2 0`
`3 1`
	`6`

Input example 2	Output example 2

`100000 3 10`
`1 9`
`2 0`
`7 3`
	`85449877`

Problem L
Listing Tedious Paths

A tree is a graph that is connected (there exists a path between any two of its vertices), undirected (the edges of the graph have no direction), and acyclic (there are no cycles).

A colorful tree is a tree in which each of its vertices has a specific color.

A tedious path is a path in the tree such that both the initial and final vertices have the same color, and there is no vertex or edge that appear more than once in the path. Note that the color of the intermediate vertices, if there are any, are irrelevant.

Given a colorful tree, with N vertices, your task is calculate, for each of the edges, the number of tedious paths that go through that edge.

Input

The first line contains the number of vertices N ($1 \leq N \leq 10^5$). The second line contains N integers $C_1, \ldots, C_N$, where C_i ($1 \leq C_i \leq N$) represents the color of the vertex i. The next $N-1$ lines contains two integers each, u and v, representing an edge ($1 \leq u, v \leq N$ and $u \neq v$). It's guaranteed that the given graph is a tree.

Output

Print $N-1$ integers, representing the number of tedious paths that go through each edge, following the same order of the edges as they are given in the input.

Input example 1	**Output example 1**

6
1 1 1 2 2 1
1 2
2 3
4 6
2 4
1 5
	4 3 3 4 1

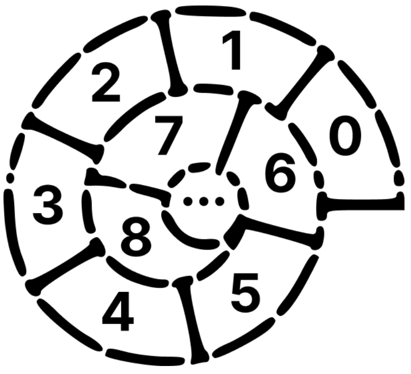
Input example 2	**Output example 2**

12
1 2 3 1 2 2 1 3 2 3 1 2
1 2
2 3
2 4
4 5
4 6
1 7
7 8
7 9
9 10
6 11
6 12
	10 2 10 4 9 9 2 6 2 3 4

Problem M
Hopscotch Marathon

October 8th, 2022. This is the date of the most awaited event of the year by computer science students across the country. No, we are not talking about ICPC.

We are talking, of course, about Hopscotch! For those unfamiliar, Hopscotch is an annual competition traditionally held as an ICPC side event. Live streamed to spectators from all continents, and to practitioneers of the most esoteric programming languages, this exotic variant of the popular children's game takes place in an infinite, spiral-shaped court, subdivided into sequentially numbered areas starting at zero, as depicted below.



This year, Hopscotch has attracted a record number of N participants, numbered sequentially from 1 to N. It is known that the i-th participant starts in the area numbered A_i.

Hopscotch consists of Q rounds. During the q-th round, Carlão, beloved Hopscotch organizer for longer than anyone can remember, will communicate two integers to participants: c_q and d_q. This is an order for all participants with identifying number i such that i and c_q share a common integer factor larger than 1 to retrogress d_q positions in the Hopscotch court, one by one, never going back further than position 0. (Any participant who eventually returns to position 0 should remain there indefinitely, ignoring any further retrogress commands, so as not to leave the court.)

Under the assumption that the participants have executed the instructions perfectly (they would never want to disappoint Carlão), your task is to determine, for each participant, the number of the round in which he or she returns to position 0 (or otherwise indicate that this never happens).

Input

The first line contains the integers N and Q ($1 \leq N, Q \leq 10^5$). The second line contains N integers, namely, A_1, A_2, $\cdots$, A_N ($1 \leq A_i \leq 10^9$). Each of the next Q lines contains two integers, c_q and d_q ($1 \leq c_q \leq 10^5$, $1 \leq d_q \leq 10^9$).

Output

You must output N lines. The i-th line should contain a single integer, indicating the number of the round when the i-th participant returns to position 0 (or the value -1, if that never happens).

Input example 1	**Output example 1**

```
7 6
10 20 30 40 50 60 70
2 25
3 36
100 42
5 10
7 70
1 1000
```
	```
	-1
	1
	2
	3
	4
	2
	5
	```

Input example 2	**Output example 2**

```
6 4
100 100 100 100 100 100
2 50
3 50
5 99
5 1
```
	```
	-1
	-1
	-1
	-1
	4
	2
	```

Problem N
Numbers on both Sides

You have just won a deck of cards with N cards. Each of these cards has two numbers written on it: one on the front side, and another on the back side.

Your friend has challenged you to a game. He shuffled the cards and put them on a table. The cards are laid out on a line, side by side, with the front side facing up.

From left to right, you know that the number written on the front side of the i-th card is A_i, and that the number written on the back side of the i-th card is B_i.

The game is divided into two parts.

In the first part you should pick K cards of the deck. To pick a card, you have to choose either the first card on the left, or the first card on the right of the table, and take it for you.

After that, you have to choose L of the cards you picked up and flip them.

Your score will be equal to the sum of the numbers written on the front side of all of the K cards you picked, plus the sum of the numbers written on the back side of the L cards you flipped.

The goal? To achieve the highest possible score, of course!

Input

The first line contains an integer $N (1 \leq N \leq 10^5)$. The second line contains N integers A_1, A_2, $\ldots$, A_N, $(1 \leq A_i \leq 10^9)$. The third line contains N integers B_1, B_2, $\ldots$, B_N, $(1 \leq B_i \leq 10^9)$. The fourth line contains two integers K and L $(1 \leq L \leq K \leq N)$.

Output

Print one line containing an integer, representing the highest score possible.

Input example 1	Output example 1

5
9 7 2 2 9
5 2 2 3 1
2 1
	23

Input example 2	Output example 2

5
9 7 2 2 9
5 9 2 3 1
2 1
	25