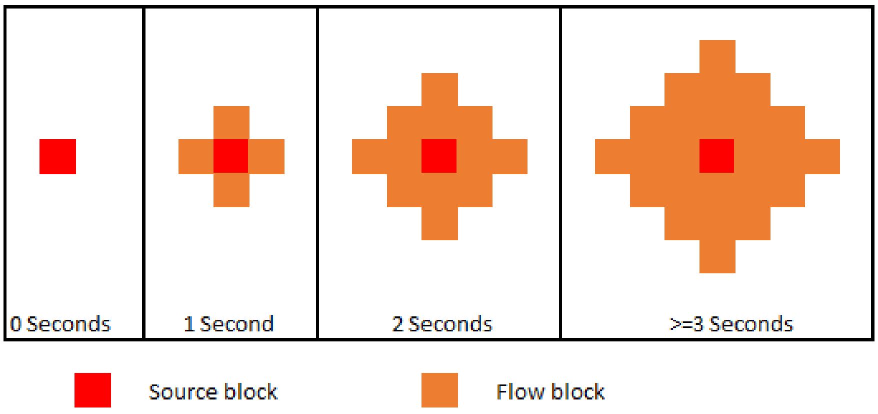
Problem A. An overly elaborate trap

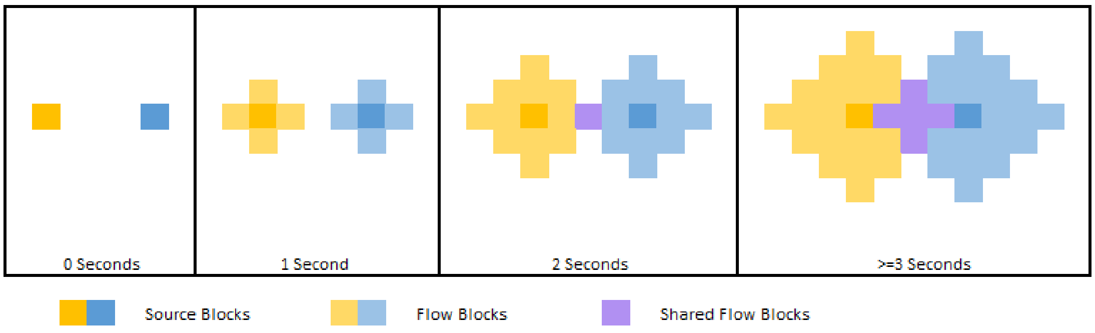
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

We all have played Minecraft at least once, it's such an iconic from our childhood! as everyone knows when on a flat surface Minecraft's lava flows like in the following image:



Lava flow, note that it stops flowing when the timer reaches 3 seconds.

If two or more lava source blocks happen to be close enough for their flow/source blocks to touch they will act independently of one another and their shared lava blocks will just overlap as shown in the image.



Simultaneous lava flow.

xX_ABSALON_Xx got a server running with his friend ISMA777_YT (who happens to like playing pranks on him). ISMA777_YT planned an elaborate prank, he dug up a hole starting on the (0,0) coordinate and expanded it r blocks north and c blocks east, ensuring that the bottom was completely flat, and placed M markers each signaling a place where a lava block will be placed, once xX_ABSALON_Xx returns to the server he plans to teleport him to a block inside the hole, lock him in place and put at the same time M lava blocks in all the markers. Now ISMA777_YT doesn't want xX_ABSALON_Xx to die to the lava and lose all his stuff so he wants you to determine which block will be the last reached by the lava (if a block is never touched by the lava, then it is considered to be touched last), in case of a tie you should pick the block with the highest row value if there is still a tie you should pick the one with the lowest column value.

Input

The first line of the input file contains two integers r and c ($1 \leq r, c \leq 10^3$) indicating the number of blocks in which the hole expands north and east respectively.

The second line contains an integer M $(1 \leq M \leq r \times c)$ indicating the number of markers to be placed in the hole.

The next M lines will contain two integers r_i $(0 \leq r_i < r)$ and c_i $(0 \leq c_i < c)$ indicating the $i - th$ marker's position for all $1 \leq i \leq m$. No two markers will be placed in the same spot.

Output

Print two integers r_{ans} and c_{ans} separated by a space, representing the coordinate where the lava will reach last, if there are multiple answers follow the tiebreaker rules listed on the problem statement. If the whole hole is filled with lava from moment 0, output -1.

Examples

standard input	standard output

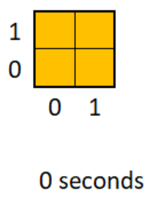
`2 2`
`4`
`0 0`
`0 1`
`1 0`
`1 1`
	`-1`

`3 4`
`2`
`2 0`
`0 3`
	`2 2`

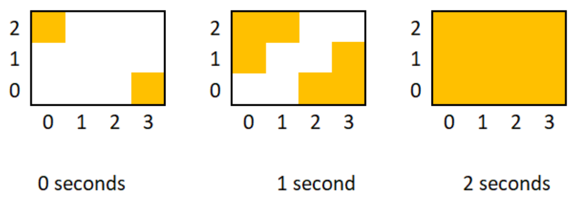
`2 5`
`1`
`0 0`
	`1 3`

Note

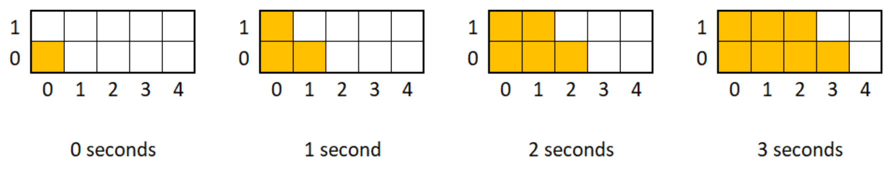
On each test case lava flows as illustrated:



test 1



test 2



test 3

Problem B. Brainfuck

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

An esoteric programming language (ess-oh-terr-ick), or esolang, is a computer programming language designed to experiment with weird ideas, to be hard to program in, or as a joke, rather than for practical use.

There is a small but active Internet community of people creating esoteric programming languages and writing programs in them, as well as debating their computational properties (e.g. if said languages are Turing-complete).

Brainfuck is one of the most famous esoteric programming languages, and has inspired the creation of a host of other languages. Due to the fact that the last half of its name is often considered one of the most offensive words in the English language, it is sometimes referred to as "brainf***", "brainf*ck", "brainfsck", "b****fuck" (as a joke), "brainf**k", "branflakes", "brainoof", "brainfrick", "bf", etc. This can make it a bit difficult to search for information regarding brainfuck on the web, as the proper name might not be used at all in some articles.

Esolang Wiki

For the sake of this problem you may assume the following:

Brainfuck operates on a global array of memory cells, each initially set to 32. There is a pointer, initially pointing to the first memory cell. The commands are:

Command	Description
$>$	Move the pointer to the right, if the pointer is already at the right-most cell move it to the first cell.
$<$	Move the pointer to the left, if the pointer is already at the first cell move it to the rightmost one.
$+$	Increment the memory cell at the pointer(**Note:** If the value of the memory cell is 126 at the moment the operation is called its new value will be 32)
$-$	Decrement the memory cell at the pointer (**Note:** If the value of the memory cell is 32 at the moment the operation is called its new value will be 126)
.	Output the character signified by the cell at the pointer

In this particular problem, you will make a simplified brainfuck parser that, given a string P of at most 10^5 characters consisting only of $<> + -$. and an integer S denoting the size of the global array, will evaluate the program specified in P.

Input

The first line of the input file contains a string P of at most 10^5 characters representing the program you must evaluate. It is guaranteed that the program denoted by P has at least one print operation.

The second line will contain an integer S ($1 \leq S \leq 10^9$) denoting the size of the global array.

Output

Print the resulting output after evaluating the program contained on P.

Examples

standard input	standard output

`+.`
`2`
	`!`

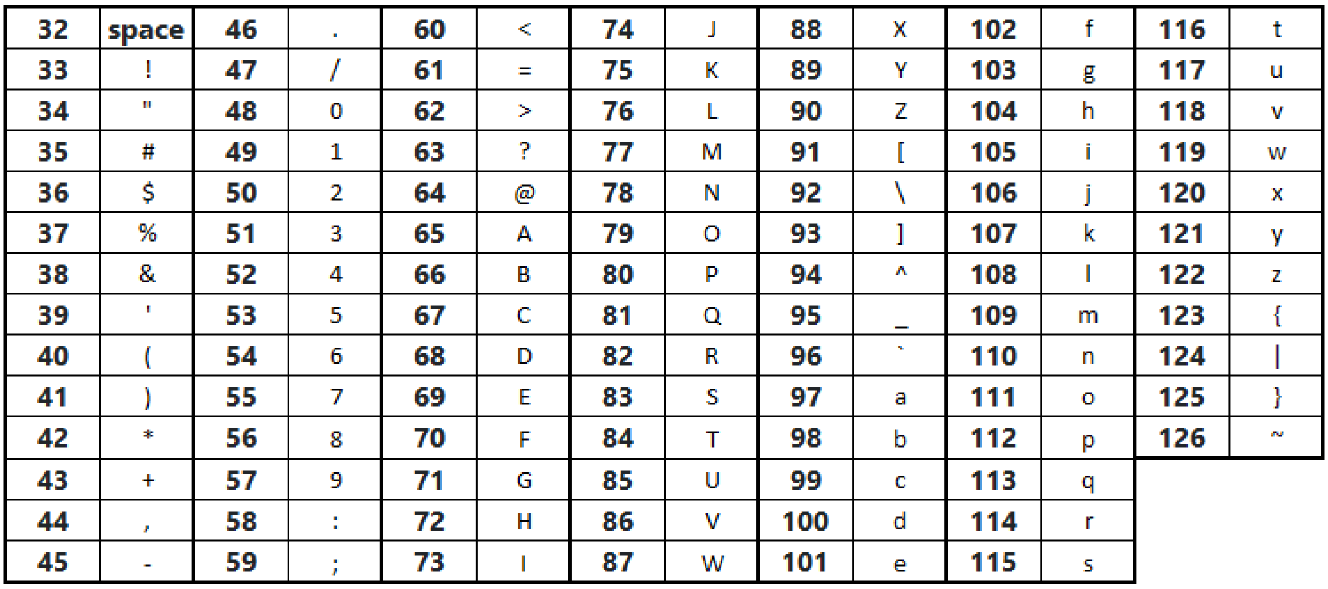
`+.>+.<+.`
`1`
	`!"#`

`-.<+.>++.`
`2`
	`~!!`

`<++>>++.`
`2`
	`$`

Note

In most programming languages there is a relation integer character, in this particular case is expressed in the following table.



32	space	**46**	.	**60**	<	**74**	J	**88**	X	**102**	f	**116**	t
33	!	**47**	/	**61**	=	**75**	K	**89**	Y	**103**	g	**117**	u
34	"	**48**	0	**62**	>	**76**	L	**90**	Z	**104**	h	**118**	v
35	#	**49**	1	**63**	?	**77**	M	**91**	[	**105**	i	**119**	w
36	$	**50**	2	**64**	@	**78**	N	**92**	\	**106**	j	**120**	x
37	%	**51**	3	**65**	A	**79**	O	**93**	]	**107**	k	**121**	y
38	&	**52**	4	**66**	B	**80**	P	**94**	^	**108**	l	**122**	z
39	'	**53**	5	**67**	C	**81**	Q	**95**	_	**109**	m	**123**	{
40	(	**54**	6	**68**	D	**82**	R	**96**	`	**110**	n	**124**	|
41	)	**55**	7	**69**	E	**83**	S	**97**	a	**111**	o	**125**	}
42	*	**56**	8	**70**	F	**84**	T	**98**	b	**112**	p	**126**	~
43	+	**57**	9	**71**	G	**85**	U	**99**	c	**113**	q		
44	,	**58**	:	**72**	H	**86**	V	**100**	d	**114**	r		
45	-	**59**	;	**73**	I	**87**	W	**101**	e	**115**	s		

Problem C. Cost of doing business

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Planet FGH is located in the JKL solar system, this planet is inhabited by human-like creatures called BNM, which divided a long time ago the planet into countries, each named after a natural number, BNMs like small positive numbers, so naturally, when a country emerged they named it after the smallest unclaimed number.

International shipments are tough, some countries let in shipments coming from friends and allies after collecting a tariff, while not allowing those coming from everywhere else. These kinds of measures are a great way to either protect domestic industries, collect revenue, or as political leverage.

In this particular planet, trade blocks are formed between countries on a closed interval. A country allows incoming shipments if the company or individual that is handling them, bought a permit from at least one of the trading blocs to which the country belongs and is coming from one of its member states.

While mostly beneficial in the broader scope, tariffs and bans ultimately affect businesses, which are profit-driven entities, thus they naturally want the cheapest way to circumvent them if any, in most cases this need translates into the hiring of another company to handle the problem for them.

Jaime, having a good eye for business decided that it would be a good idea to create his own international transport company, Jaime is from country 0 and wants to know what is the minimum amount of money he needs to spend on permits such that he can send stuff to any country on the closed interval $[0 - M]$, so he asked you to write a program for him.

Input

In the first line of input there are two integers M $(0 \leq M \leq 10^5)$ and T $(1 \leq T \leq 10^5)$, representing the furthest country Jaime wants to operate in and the number of trading treaties.

The rest of the input is T lines, each representing a standing trade treaty.

For each line, three integers separated by whitespaces will be given: L_i R_i C_i representing a treaty involving countries from L_i to R_i, whose permit costs C_i dollars. $0 \leq L_i \leq R_i \leq 10^{10}$, $0 \leq C_i \leq 10^9$

Output

Print a single integer representing the minimum amount of money Jaime needs to spend on permits or -1 if there is no way to reach a country in the interval $[0 - M]$.

Examples

standard input	standard output

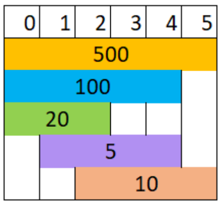
5 5
0 5 500
0 4 100
0 2 20
1 4 5
2 5 10
	30

6 3
0 1 1
2 5 1
6 6 1
	-1

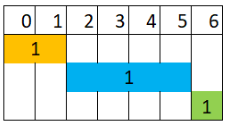
0 3
0 5 10
0 2 5
0 0 10
	0

Note

In the first test case, it is optimal to choose treaties 3,4 and 5 with a total cost of 30.



In the second test case, it is impossible to reach country 6 from country 0, as all trading blocs are disjoint.



In the third test case, as Jaime lives in country 0 and wants to reach as far as country 0, he doesn't need to buy any permits.

Problem D. Damage Control

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are currently working for a gift factory, this factory makes two kinds of products the first kind is transported on belt A, while the second is transported using belt B. At some point, both belts converge and each product in belt A must be packed with the product in belt B with the same index.

Your company's marketing department has discovered that if a person buys a gift package, and considers one of the two products in it of inferior quality their disgruntlement towards the purchase will be proportional to the difference in quality between the products.

Taking this into consideration, quality engineers have decided it would be best that given the two belts with output N, only K products should be kept in each belt and the rest will be discarded, the rest of the packaging process is left as is for efficiency purposes.

Now before approving this method, your company's analysts need to test it, and they decided that using a computer simulation would be the most cost-efficient way to do so. So now they want you to make a program that, given N, K, and the quality of the products in both belts outputs the least sum of the quality difference between products on the same package on all the K resulting packages.

Input

In the first line of input two integers will be given N ($1 \le N \le 13$) and K ($1 \le K \le N$), representing both belt's capacity and the number of packages that should be made.

The second line will contain N integers A_i such that $0 \le A_i \le 10^9$ holds, representing the items on belt A.

The third line will contain N integers B_i ($0 \le B_i \le 10^9$), representing the items on belt B.

Output

Print a single integer representing the minimum value of the sum of the quality difference between products on the same package on all the K resulting packages.

Examples

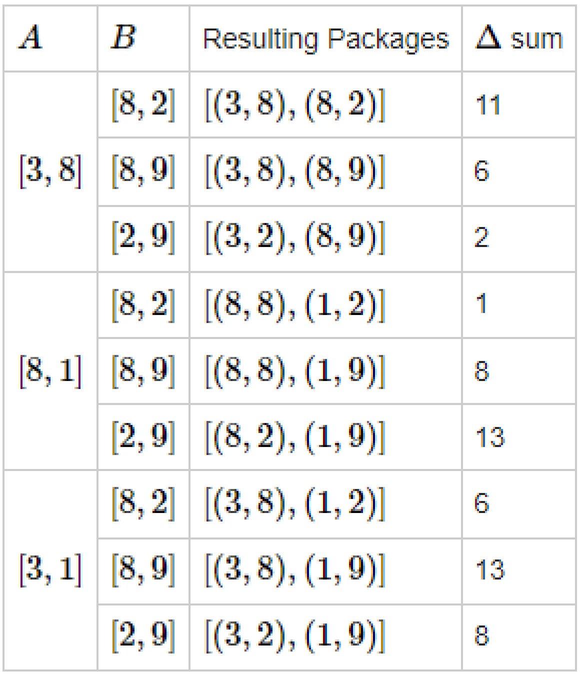
standard input	standard output

`2 2`
`5 2`
`3 8`
	`8`

`3 2`
`3 8 1`
`8 2 9`
	`1`

Note

In the first test case, products on belt A are $[3, 8, 1]$ and those corresponding to belt B are $[8, 2, 9]$, from those you must keep 2 on each belt, there are 9 ways to do so:



A	B	Resulting Packages	Δ sum
$[3, 8]$	$[8, 2]$	$[(3, 8), (8, 2)]$	11
	$[8, 9]$	$[(3, 8), (8, 9)]$	6
	$[2, 9]$	$[(3, 2), (8, 9)]$	2
$[8, 1]$	$[8, 2]$	$[(8, 8), (1, 2)]$	1
	$[8, 9]$	$[(8, 8), (1, 9)]$	8
	$[2, 9]$	$[(8, 2), (1, 9)]$	13
$[3, 1]$	$[8, 2]$	$[(3, 8), (1, 2)]$	6
	$[8, 9]$	$[(3, 8), (1, 9)]$	13
	$[2, 9]$	$[(3, 2), (1, 9)]$	8

From those 9 possible settings, the most desirable is the one with the least sum of the quality difference between products on the same package on all the K resulting packages (Δ sum) thus the 4-th row is selected.

Problem E. Elimination Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Sorting methods are fundamental while constructing software, usually, there are built-in sorting functions in every programming language, but sometimes programmers implement their own, either out of an absolute and specific necessity or to sharpen their programming skills.

Today you will implement an elimination sort named after a specific authoritarian leader. You will be given a list of integers, then you have to iterate through the list eliminating all the elements that are smaller than their predecessor, this process will be repeated until the resulting array is sorted.

Input

The first line of the input file contains an integer N ($1 \leq N \leq 1000$) indicating the size of the list you will be given.

The second and last line will contain N integers L_i ($0 \leq L_i \leq 1000$) separated by a space.

Output

Print the resulting list after sorting using elimination sort. Whitespaces must separate each element of said list.

Examples

`standard input`	`standard output`

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

`6`
`8 4 4 10 5 11`
	`8 10 11`

Note

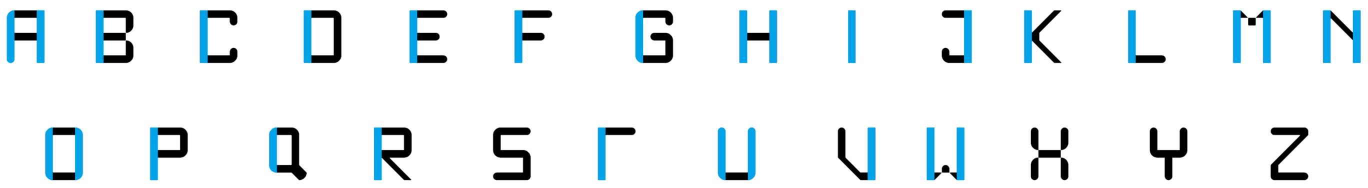
Consider the second testcase: With the original array $[8, 4, 4, 10, 5, 11]$, the algorithm will behave as follows

- During the first pass the first 4 and 5 are eliminated from the array, resulting in $[8, 4, 10, 11]$. The array is still unsorted, as 8 is greater than 4.
- On the second pass the remaining 4 is eliminated from the array, resulting in $[8, 10, 11]$. The array is now sorted, which means no further passes are required.

Problem F. F(r)ontier

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Consider the following font:



The uppercase English alphabet written using Vertical Seriff.

Let's call this particular font Vertical Seriff, as you may have noticed what makes Vertical Seriff so odd looking is the fact that almost every letter has a vertical line in it. To highlight these lines, we have colored them blue.

For any text written in Vertical Serif, we define Total Vertical Lines (TVL) as the sum of the vertical lines in all the letters. If multiple vertical lines are adjacent, they merge into a single visible line, and we define the Visible Vertical Lines (VVL) as the count of these visible lines.

The *slickness* of a text in Vertical Serif is defined as TVL minus VVL.

Given N lines of text in uppercase English letters and spaces, your task is to calculate the *slickness*of each line.

Input

The first line of the input file contains an integer N ($1 \le N \le 10^5$) indicating the number of lines of text to be written.

Each of the next N lines will contain a string of upper case English letters and spaces t_i, representing the text to be written using Vertical Seriff. It is guaranteed that the sum of the lengths of t_i for all i is less than 10^5 and that no line has leading/trailing whitespaces or spaces next to each other.

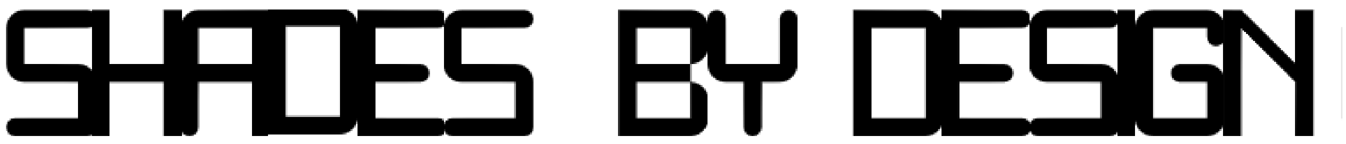
Output

For each line of text, print its slickness followed by a linebreak.

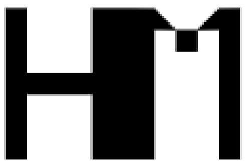
Example

standard input	standard output
2 SHADES BY DESIGN HIM	3 2

Note



Consider the text **SHADES BY DESIGN** the sum of vertical lines of all its letters is 11, but when written using the font the vertical lines of A and D merge into one making the visible vertical lines amount to 10, thus the slickness of this line in particular is 1.



While the sum of vertical lines of the text **HIM** is 5, when written with Vertical Seriff only 3 lines are visible, thus its slickness is 2.

Problem G. Growing bacteria

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	100 megabytes

Scientists from the renowned UACJ led by professor Saul are studying a new kind of slow-reproducing bacterium that can have broad applications in the medical world. This particular bacterium reproduces by a variant of binary fission following a reproductive cycle.

In simpler terms, there are cells, each with its own clock starting at a certain value t_i. After a certain amount of time (x milliseconds), any cell with a clock value of x will create a new cell and reset its own clock to a new value D while the new cell's clock will start at $2 \times D$.

For his next experiment, Saul needs at least R cells of the slow-reproducing bacteria, as R can be a huge number, and the bacteria is known for its slow reproduction cycle Saul asked you to write a program to tell him how many milliseconds he must wait for the number of bacteria to be greater than or equal to R.

Input

The first line of input has three integers N,R and D ($1 \le N \le 1000$, $0 \le R \le 10^{13}$, $1 \le D \le 10^7$), denoting the initial amount of bacterium Saul has, the amount he needs and the value a cell must restart its internal clock back to respectively.

The Second line contains N integers t_1 t_2 ... t_n separated by whitespaces indicating the initial internal clock value of each of the N bacterium Saul has. For each integer t_i, $1 \le t_i \le D$ holds.

Output

Print a single integer denoting the number of milliseconds Saul must wait in order to have at least R cells of his slow-reproducing bacteria.

Examples

standard input	standard output

`4 4 100`
`2 5 8 8`
	`0`

`4 10 10000000`
`2 5 8 8`
	`10000005`

Problem H. Harolds CUBG Conundrum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The year is 2018, the world is a simpler place, there is no virus going around and the online gaming community is at the height of the battle royale frenzy, during this magical year a lot of games of this genre were released, among them Collatzes unknown battlegrounds (also called CUBG).

CUBG is just like any other battle royale game, with one exception, its map is modeled as a set of "nodes" identifiable by an ID value being the map center's ID 1, players spawn in a random node, and they can choose to stay in their nodes and let the barrier catch up to them or they can go through the node's portal and move to the next node, there is no portal on the map center. The node a portal may put a player on may seem random to the inexperienced observer but veteran players figured out the system ages ago, if you do a quick google search on this mechanic you can find on the game's forums the following statement: "Given a node V, the portal will put you on node V/2 if V is even, or in 3V + 1 otherwise", another interesting piece of trivia is that no matter the initial nodes the game may put the players on they will always have the possibility to fight on the map center.

Harold and his N friends are playing a single-life deathmatch CUBG mod that for better or worse has all the base game features, the main problem with this mod is that (as the original game) it sometimes spawns the players in different nodes making the rounds last way longer than they should. Harold is well versed in the game and its inner workings but not so in programming so he asked for your help, he wants you to make a program for him that given the spawning nodes of X and his N mates, determines a common node X such as the max distance from any of the spawning nodes to X is minimized and how long should Harold wait on that node before all his friends arrive considering it takes a second to move from one node to another.

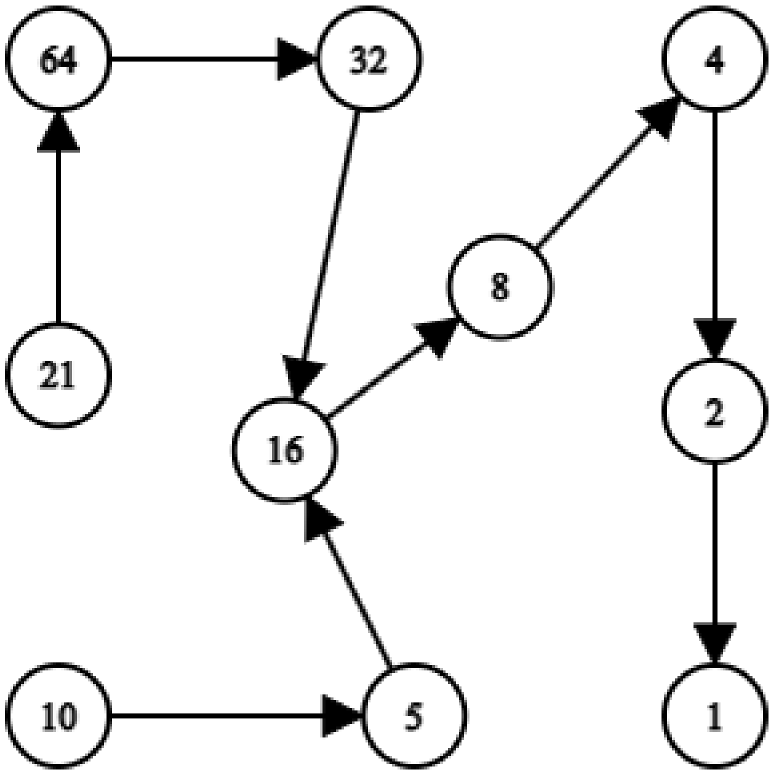


Figure 1: Two players path assuming they spawn in nodes 10 and 21, and travel to the map center.

Input

The first line of the input file contains two integers N ($1 \leq N \leq 10^5$) and S_{Ant} indicating the number of friends that are playing with Harold and Harold's spawning point respectively.

The next line will contain N integers separated by spaces S_i representing the spawning point of each player. (It is guaranteed that the sum of the distance from a_i to the center of the map over all i plus the distance from S_{Ant} to the center is less than 10^5).

Note: No matter which nodes the players spawn at, it will never take them more than 10^5 portal leaps to reach the map center, and they will never go through a node with an ID value exceeding 10^{18}

Output

On a line ending with a linebreak print the following message without the double quotes: "*Harold must reach node X and wait T second(s) for his friend(s) to arrive*" where X and T are both integers denoting the common node where all players must meet and the number of seconds Harold must wait on that node before all his friends arrive.

Note: T and N being 1 or 0 doesn't influence the output formatting, their units should remain as "*second(s)*" and "*friend(s)*" respectively.

Examples

standard input	standard output

`1 21`
`10`
	`Harold must reach node 16`
	`and wait 0 second(s) for his friend(s) to arrive`

`4 13`
`10 26 3 6`
	`Harold must reach node 10`
	`and wait 1 second(s) for his friend(s) to arrive`

Problem I. Integer Inversion

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A trivial problem statement for a trivial problem, how fitting!, you are given an integer, your only task is to print it in reverse order without any leading zeroes, meaning if you are given 2980 you have to print 892.

Input

A single integer N such that $1 \leq N \leq 10^8$.

Output

An integer ans such that it's the same as N in reverse order without any leading zeroes.

Examples

`standard input`	`standard output`
5000	5
1234	4321
852020	20258

Problem J. Permutation Checker

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Samantha is a young programmer who loved solving challenging problems. One day, while browsing through omegaUp, she stumbled upon an interesting problem.

The problem was to write a program that would determine whether a given list of numbers was a permutation or not. Samantha knew that a permutation was a sequence of distinct integers from 1 to N, where N is the length of the sequence. She solved the problem in a matter of minutes, and now she wants you to give it a try.

Input

The first line of input has an integer N ($1 \leq N \leq 100$), denoting the length of the list you will be given. The Second line contains N sorted integers $a_1 a_2 ... a_n$ separated by whitespaces indicating the elements of the list, such that each element is an integer between 1 and 1000 inclusive.

Output

Output on a single line:

- *YES* if the given list is a permutation.
- *NO* otherwise.

Examples

standard input	standard output
`1` `1`	`YES`

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

`5`
`1 3 4 5 5`
	`NO`

`5`
`1 2 4 5 6`
	`NO`

Problem K. Kodemania?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

I know that the correct way to spell 'Kodemania' is, in fact, with a 'C.' However, I prefer it when problem names match their corresponding letter in the problem set, so here we are. Your task is simple: you will be given a single character of the English alphabet (either in lowercase or uppercase), and you need to print whether that character is in 'Kodemania2023'

Input

You will be given a single English character on a line.

Output

Output on a single line:

- *YES* if the given character is in 'Kodemania2023'.
- *NO* otherwise.

Examples

standard input	standard output
`m`	`YES`
`2`	`YES`
`B`	`NO`
`E`	`YES`
`M`	`YES`