# A. Cookie Tower

time limit per test: 1 second
memory limit per test: 256 megabytes

Cookie Monster is playing with his three cookies, which have distinct sizes $a$, $b$, and $c$, respectively. He wants to create a cookie tower by stacking them up. Cookie Monster wants the largest cookie to be on the bottom (for a strong foundation). However, he wants the top of the tower to be big as well, so he wants to place the second largest cookie on top and the smallest cookie in the middle.

Given the sizes of Cookie Monster's cookies $a$, $b$, and $c$, help him order the cookies in the order biggest, smallest, middle.

### Input

The first line will contain the three integers $a$, $b$, and $c$ ($1 \leq a, b, c \leq 100$). It is guaranteed that $a \neq b$, $b \neq c$, and $a \neq c$.

### Output

Print three integers corresponding to the cookies in the order Cookie Monster wants them in.

### Examples

| input |
| --- |
| 1 2 3 |
| **output** |
| 3 1 2 |

| input |
| --- |
| 63 33 15 |
| **output** |
| 63 15 33 |

# B. Placing Dominoes

time limit per test: 1 second
memory limit per test: 256 megabytes

Cookie Monster has an empty board with $m$ rows and $n$ columns. He wants to place as many $2 \times 1$ dominoes as possible on the board without overlapping. Dominoes can be placed either vertically or horizontally, and each domino must cover exactly two adjacent cells.

Determine whether it is possible for Cookie Monster to tile the entire board with such dominoes so that no cell remains uncovered.

### Input

A single line containing two integers $m$ and $n$ ($1 \leq m, n \leq 100$).

### Output

Print "YES" if the board can be completely tiled with $2 \times 1$ dominoes. Otherwise, print "NO".

### Examples

| input |
| --- |
| 3 4 |
| **output** |
| YES |

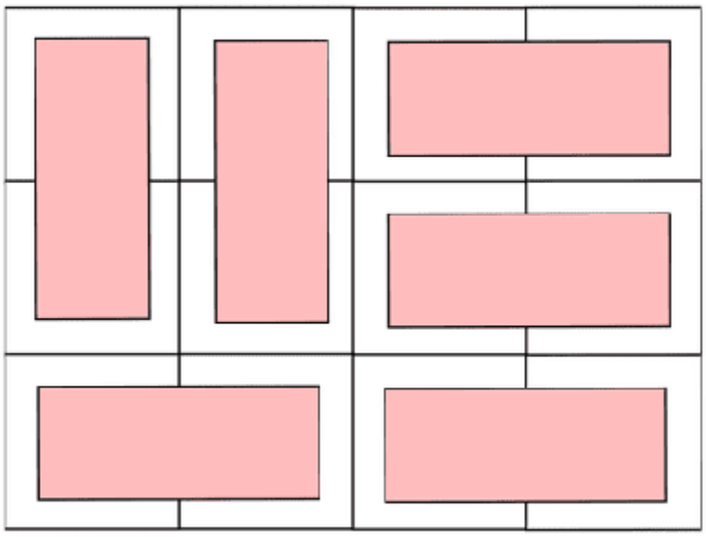
| input |
| --- |

| 5 5 |
|---|

| output |
|---|
| NO |

### Note

Images representing the first sample test cases are given below.



## C. Card Game

time limit per test: 1 second
memory limit per test: 256 megabytes

Cookie Monster is bored on a long car ride and has nothing to do except play with a deck of specially numbered cards. He has already played every card game he knows, so he is trying to come up with a new one. After a while, he comes across an interesting idea.

Cookie Monster has $n$ cards, each containing one of the digits in the set $\{0, 1, 2, 5, 6, 8, 9\}$. The cards are fixed together in a specific order, but Cookie Monster can rotate the entire set of cards $180$ degrees. Now he asks you a question: given the set of $n$ cards he currently has and the order they are in, what will he get when he rotates it by $180$ degrees?

See the notes section for more specifics on how the rotation operation affects the card numbers.

### Input

The first line contains the integer $n$ ($1 \leq n \leq 100$).

The second line contains $n$ space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 9$).

### Output

Output $n$ integers representing the resulting set of cards.

### Examples

| input |
|---|
| 3<br>2 5 8 |

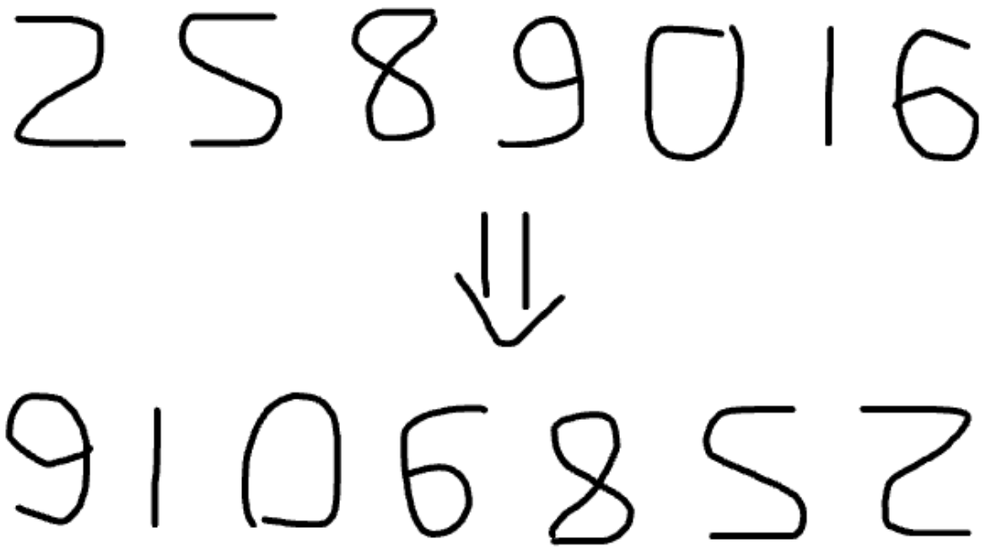
| output |
|---|
| 8 5 2 |

| input |
|---|
| 7<br>2 5 8 9 0 1 6 |

| output |
| --- |
| 9 1 0 6 8 5 2 |

### Note

For the second sample test case, an image is provided below.



## D. Cookie Arrangement

time limit per test: 1 second
memory limit per test: 256 megabytes

It's the last day of school, and Cookie Monster wants to give each of his $n \cdot m$ students a gift — their own personalized cookie! Cookie Monster has affectionately assigned each of his students a distinct integer ID from $1$ to $n \cdot m$, and he labeled each cookie with the ID of the intended recipient.

Cookie Monster chose a very interesting classroom arrangement. Students sit in a grid of desks with $n$ rows and $m$ columns. Cookie Monster has labeled each desk with a number from $1$ to $n \cdot m$. The topmost row is numbered from $1$ to $m$ from left to right, the next row is numbered $m + 1$ to $2m$ from left to right, and so on. Each student sits in the desk corresponding to their ID.

Unfortunately, Cookie Monster fell sick on the last day of school, and Elmo is now the substitute teacher. Even worse, Elmo thinks that the desks are numbered differently. He thought the leftmost column was numbered from $1$ to $n$ from top to bottom, the next column was numbered from $n + 1$ to $2n$ from top to bottom, and so on.

Elmo placed each gift at the desk he thought corresponded to the ID labeled on the cookie. Given $n$ and $m$, determine the number of gifts Elmo placed on the correct desk, as Cookie Monster intended.

### Input

The first line of the input will contain two integers $n$ and $m$ ($1 \leq n, m \leq 10^9$).

### Output

Print a single integer — the number of gifts Elmo placed on the correct desk.

### Examples

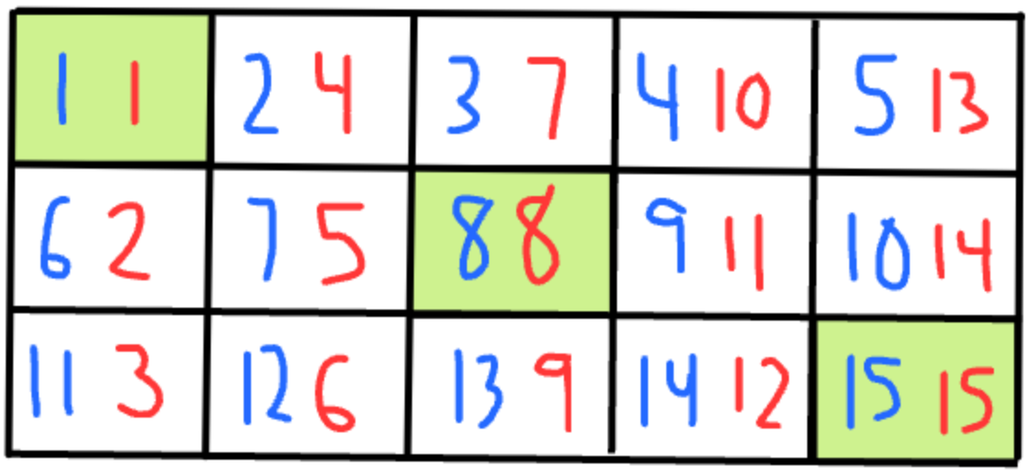
| input |
| --- |
| 3 5 |

| output |
| --- |
| 3 |

| input |
| --- |
| 1 10 |
| **output** |
| 10 |

### Note

For the first sample test case, the image below shows the classroom arrangement. Cookie Monster's ordering is shown in blue, Elmo's ordering is shown in red, and the green color corresponds to the desks where the gifts were placed correctly.




## E. Cookie Monster Sorts

time limit per test: 1 second
memory limit per test: 256 megabytes

Cookie Monster has discovered an incredibly fast algorithm to sort a permutation of $n$ integers quicker than the quickest quick sort. A permutation of length $n$ is defined as an array of $n$ distinct integers in the range $[1, n]$.

One iteration of Cookie Monster's algorithm does the following:

- For each $i$ from $1$ to $n$ in order, perform $\text{swap}(i, a_i)$, where $\text{swap}(i, j)$ is a function that swaps the values $a_i$ and $a_j$ in place.

After each iteration, if the permutation is sorted, the algorithm terminates. If the permutation is initially sorted, then $0$ iterations are done.

Cookie Monster has proven that this algorithm will always terminate with the correct answer. However, he wants to determine the runtime of his sorting algorithm over many cases. So, given a permutation, you need to help him determine the number of iterations his algorithm will take to sort the permutation.

### Input

The first line of the input will contain $n$ ($1 \leq n \leq 2 \cdot 10^5$).

The second line of the input will contain the permutation $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$). It is guaranteed that all $a_i$ are distinct.

### Output

Print a single integer, the minimum number of iterations that Cookie Monster's sort needs to sort the permutation.

### Example

| input |
| --- |
| 4<br>2 3 4 1 |
| **output** |
| 2 |

### Note

In the sample test, Cookie Monster's algorithm does the following to the permutation:

- After one operation, the permutation is $[3, 2, 1, 4]$.
- After two operations, the permutation is $[1, 2, 3, 4]$.

Thus, Cookie Monster needs $2$ operations to sort the permutation in this case.

## F. Forest

time limit per test: 1 second
memory limit per test: 256 megabytes

Cookie Monster has a forest of $n$ nodes and $m$ edges. He wants to add $n - m - 1$ edges to the forest to create a tree rooted at node $1$. What is the maximum possible height (the maximum number of edges on a path from the root, node $1$, to any other node) of this tree?

### Input

The first line of input contains two integers $n$ ($1 \leq n \leq 2 \cdot 10^5$) and $m$ ($0 \leq m \leq n - 1$).

The next $m$ lines each contain two integers $u$ and $v$ ($1 \leq u, v \leq n$), indicating there is an edge connecting nodes $u$ and $v$. It is guaranteed that the graph contains no self-edges or cycles, and no edge appears multiple times.

### Output

Output a single integer, the maximum height that Cookie Monster can achieve over all possible resulting trees.

### Example

| input |
| --- |
| 10 7<br>5 9<br>4 6<br>4 1<br>4 10<br>5 7<br>5 3<br>8 2 |
| **output** |
| 7 |

### Note

Cookie Monster can add an edge between nodes $7$ and $10$ and an edge between nodes $2$ and $3$ to form a tree. The height of this tree is $7$, the distance from node $1$ to node $8$.

## G. The Ross Program

time limit per test: 1 second

memory limit per test: 256 megabytes

Cookie Monster and Elmo are playing a game on an $8 \times 8$ chessboard, but Cookie Monster is a prolific fraudster with a fake CS degree from MIT. The game is too hard for him, so Cookie Monster probably plans to cheat using a hidden vibrating device, but before this, he needs your help.

On each square of the board is a number $a_{ij}$. With one move, either player can increase the value of a single square $a_{ij}$ by $1$. First, Cookie Monster performs no more than $x$ moves on the board. Second, Elmo performs no more than $y$ moves on the board.

At the end of each of their turns, Cookie Monster's score is the number of $a_{ij}$ that are prime. Cookie Monster wants to maximize his score, while Elmo wants to minimize it. Determine the maximum score Cookie Monster can attain.

### Input

The first line will contain two integers $x$ and $y$ ($0 \leq x, y \leq 64$).

The following $8$ lines will each contain $8$ integers $a_{ij}$ ($3 < a_{ij} \leq 10^6$).

### Output

Output a single integer, the maximum possible attainable score for Cookie Monster if both players act optimally.

### Example

| input |
| --- |
| 18 20<br>3 4 5 6 7 8 9 10<br>11 12 13 14 15 16 17 18<br>19 20 21 22 23 24 25 26<br>27 28 29 30 31 32 33 34<br>35 36 37 38 39 40 41 42<br>43 44 45 46 47 48 49 50<br>51 52 53 54 55 56 57 58<br>59 60 61 62 63 64 65 66 |
| **output** |
| 14 |

### Note

There are $17$ prime numbers between $3$ and $66$ and exactly $17$ other numbers that are one less than a prime. Cookie Monster can use $17$ of his $18$ available moves to make these numbers prime, and Elmo will use his $20$ moves to remove $20$ of them, leaving $14$ remaining.

## H. Bay

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Cookie Monster runs $n$ cookie shops, numbered from $1$ to $n$. Since his cookies are very popular, he needs to keep track of the amount he has in stock at any moment.

At the beginning of the day, the $i$-th cookie shop has $a_i$ cookies. For each of the next $t$ minutes, a customer will buy cookies from one of Cookie Monster's shops. Specifically, on the $j$-th minute, a customer will buy $c_j$ cookies from shop $s_j$. It is guaranteed that a customer never buys more cookies than there are available at the shop.

Cookie Monster wants to determine if he will ever be in danger of running out of cookies. He asks you $q$ questions of the form: "What is the first time in minutes that the total number of cookies in shops $i, \ldots, j$ drops below $x$ cookies?" For each question, help Cookie Monster find the first time his condition is satisfied, or print $-1$ if it is never true.

### Input

The first line will contain integers $n$, $t$, and $q$ ($2 \leq n \leq 10^5$, $1 \leq t, q \leq 5 \cdot 10^4$).

The second line will contain integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$) - the number of cookies in each shop at the start of the day.

The $i$-th of each of the next $t$ lines will contain integers $s_i$ and $c_i$ ($1 \leq s_i \leq n$, $0 \leq c_i \leq a_{s_i}$).

Each of the next $q$ lines will contain the integers $i$, $j$, and $x$ ($1 \leq i \leq j \leq n$, $1 \leq x \leq 10^{18}$) for each query.

### Output

Print $q$ integers - the answer for each query Cookie Monster asks.

### Example

| input |
| --- |
| 4 5 5<br>8 5 3 5<br>1 4<br>3 2<br>2 2<br>4 1<br>2 3<br>1 3 5<br>3 4 4<br>2 3 4<br>1 4 16<br>2 2 1 |
| **output** |
| 5<br>-1<br>3<br>2<br>5 |

### Note

In the sample test, the number of cookies in each shop over time is as follows:

- $t = 0$: $[8, 5, 3, 5]$ - $t = 1$: $[4, 5, 3, 5]$ - $t = 2$: $[4, 5, 1, 5]$ - $t = 3$: $[4, 3, 1, 5]$ - $t = 4$: $[4, 3, 1, 4]$ - $t = 5$: $[4, 0, 1, 4]$

## I. Lock Combinations

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Cookie Monster has an interesting $n$ digit combination lock, with each digit being an integer ranging from $[0, 998\,244\,353)$. The lock starts out with all zeroes, and it is interesting because the digits are arranged in a circular fashion. When Cookie Monster goes to rotate one digit, the next $m$ digits in the clockwise direction also change with it. Help Cookie Monster determine if a specific combination is reachable from all zeroes and give him a set of steps he can take to achieve it.

### Input

The first line contains two space-separated integers $n$ and $m$ ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m < n$).

The second line consists of $n$ space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < 998\,244\,353$) which represent a specific combination.

### Output

Output "YES" if Cookie Monster can create the combination or "NO" if he can't. Then, if you said

"YES", on the next line print $n$ integers $c_1, c_2, \ldots, c_n$ each in the range $[0, 998\,244\,353)$, where $c_i$ corresponds to the number of times Cookie Monster should increase the digit on the $i$-th lock.

### Examples

| input |
| --- |
| 6 3<br>1 1 1 2 2 2 |
| **output** |
| NO |

| input |
| --- |
| 6 3<br>1 1 1 1 2 2 |
| **output** |
| YES<br>998244352 1 0 1 0 1 |

## J1. Cookie Monster is Sigma (Easy Version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes

**This is the easy version of the problem. The only differences between the two versions are the constraints on $n$ and $k$ as well as the time limit.**

At the annual Monster Convention, there are $n$ monsters present numbered from $1$ to $n$, with Cookie Monster assigned number $n$. These monsters convene at a circular table with $n$ seats. Given a value of $k$, a monster is considered *mogged* if another monster assigned a higher number is sitting up to $k$ seats away.

If the $n$ monsters randomly sit in one of the $n$ seats such that no two monsters share a seat, find the probability, modulo $998\,244\,353$, that every monster except Cookie Monster is *mogged*, making Cookie Monster the Sigma.

### Input

The first line of the input contains integers $n$ and $k$ ($1 \leq k < n \leq 1000$).

### Output

Output the probability of every monster except Cookie Monster being *mogged*, modulo $998\,244\,353$.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where $p$ and $q$ are integers and $q \neq 0 \mod M$. Output the integer equal to $p \times q^{-1} \mod M$.

### Examples

| input |
| --- |
| 4 1 |
| **output** |
| 665496236 |

| input |
| --- |
| 9 2 |
| **output** |
| 748683265 |

### Note

In the first sample test, the answer is $\frac{2}{3}$ as the only arrangement where a monster avoids being *mogged* is when Cookie Monster and monster number $3$ are in opposite seats.

In the second sample test, the answer is $\frac{1}{4}$.

## J2. Cookie Monster is Sigma (Hard Version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes

**This is the hard version of the problem. The only differences between the two versions are the constraints on $n$ and $k$ as well as the time limit.**

At the annual Monster Convention, there are $n$ monsters present numbered from $1$ to $n$, with Cookie Monster assigned number $n$. These monsters convene at a circular table with $n$ seats. Given a value of $k$, a monster is considered *mogged* if another monster assigned a higher number is sitting up to $k$ seats away.

If the $n$ monsters randomly sit in one of the $n$ seats such that no two monsters share a seat, find the probability, modulo $998\,244\,353$, that every monster except Cookie Monster is *mogged*, making Cookie Monster the Sigma.

### Input

The first line of the input contains integers $n$ and $k$ ($1 \leq k < n \leq 10^5$).

### Output

Output the probability of every monster except Cookie Monster being *mogged*, modulo $998\,244\,353$.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where $p$ and $q$ are integers and $q \neq 0 \mod M$. Output the integer equal to $p \times q^{-1} \mod M$.

### Examples

| input |
| --- |
| 4 1 |
| **output** |
| 665496236 |

| input |
| --- |
| 9 2 |
| **output** |
| 748683265 |

### Note

In the first test case, the answer is $\frac{2}{3}$ as the only arrangement where a monster avoids being *mogged* is when Cookie Monster and monster number $3$ are in opposite seats.

In the second test case, the answer is $\frac{1}{4}$.

## K. Cookie Scoring

time limit per test: 4 seconds
memory limit per test: 256 megabytes

Cookie Monster has a peculiar way of determining the deliciousness of cookies. Suppose he had an

array $a$ of $n$ cookies, where the $i$-th cookie has $a_i$ chocolate chips. He would say that the deliciousness of the array is the maximum bitwise AND ($\&$) of a subsequence of $a$ with length $k$.

Unfortunately, Cookie Monster doesn't have $n$ cookies right now. In fact, due to production issues, he currently has $0$ cookies. Since you feel sad for him, you want to bake him an array of $n$ cookies. As you consider how to do this, you wonder: in how many ways can I create an array of $n$ cookies where each cookie has strictly less than $2^m$ chocolate chips so that the deliciousness of the array is exactly $x$? Of course, you are only interested in the answer modulo $998\,244\,353$.

### Input

The first line contains $4$ integers $n$, $m$, $k$, and $x$ ($1 \leq k \leq n \leq 2 \cdot 10^5$, $1 \leq m \leq 30$, $0 \leq x < 2^m$)

### Output

Output the answer modulo $998\,244\,353$. Two arrays are considered different if there is an index $i$ where the corresponding cookies at index $i$ have a different number of chocolate chips. Cookies can have $0$ chocolate chips.

### Examples

| input |
| --- |
| 3 2 1 1 |
| **output** |
| 7 |

| input |
| --- |
| 3 2 2 2 |
| **output** |
| 22 |

| input |
| --- |
| 200000 30 34867 1073741823 |
| **output** |
| 361899829 |

### Note

In the first sample test, there are $7$ ways that you can bake $3$ cookies with deliciousness $1$:

- $[1, 1, 1]$
- $[1, 1, 0]$
- $[1, 0, 1]$
- $[1, 0, 0]$
- $[0, 1, 1]$
- $[0, 1, 0]$
- $[0, 0, 1]$

## L. Pathogen Focus

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Cookie Monster is playing with an infinite square grid, $G$. Each square of $G$ has an integer in it, which is initially zero.

Cookie Monster also has an $n \times m$ array with him filled with integers $a_{i,j}$ ($-10^9 \leq a_{i,j} \leq 10^9$). In one operation, Cookie Monster can slide his rectangle anywhere on the infinite grid and add a fixed integer multiple of each value in the rectangle to the corresponding cell it lies on.

More formally, Cookie Monster can pick any integers $x$, $y$, and $c$ and perform the following operation:

```
for i <- 1 to n do
    for j <- 1 to m do
        G[i + x, j + y] <- G[i + x, j + y] + c * a[i, j]
```

After performing some finite sequence of operations, determine if it is possible for the infinite grid to have one $1$, one $-1$, and the rest of the entries $0$. If it is possible, minimize the $\text{Manhattan Distance}^{\dagger}$ between the square with the $1$ and the square with the $-1$. Since the Manhattan Distance could be very large, please output the desired quantity modulo $998244353$.

$^{\dagger}$The Manhattan distance of two squares with coordinates $(x_1, y_1)$ and $(x_2, y_2)$ is $|x_1 - x_2| + |y_1 - y_2|$.

### Input

The first line of the input contains two integers, $n$ and $m$ $(1 \leq n, m \leq 500)$.

The $i$-th of the next $n$ lines contains $m$ integers: $a_{i1}, a_{i2}, \ldots, a_{im}$.

### Output

If it's impossible to obtain an infinite square grid with one $1$, one $-1$, and all other entries $0$, output "$-1$" without the quotes.

Otherwise, output one integer: the minimum Manhattan distance, modulo $998\,244\,353$, between the square with the $1$ and the square with the $-1$ after the operations.

### Example

| input |
| --- |
| 2 3<br>1 0 0<br>0 0 1 |
| **output** |
| 6 |

### Note

Cookie Monster can perform two operations as follows:

- $x = 0$, $y = 0$, $c = 1$
- $x = 1$, $y = 2$, $c = -1$