Historical revisionism (`bitcoin2`)

Since last time, William became very skeptical about Bitcoin. Its value is just *too changeable*! People will never use it if they can't be sure of its value.

For this reason, William intends to fake some historical data in order to convince people that Bitcoin is slowly *converging to some final value*. He can only delete the historical entry of some days: he cannot add or change the data, someone will notice it!

Figure 1: William changing the data.

We say that a sequence of prices is *converging* when:

- It's alternating (switching between "highs" and "lows").
- Its highs are decreasing (if the price hits a high of 1000€, the next high will be at most 999€).
- Its lows are increasing (if the price hits a low of 100€, the next low will be at least 101€).

For example, let's assume that William has the history of 7 price changes:

$$600, \quad 400, \quad 800, \quad 1500, \quad 700, \quad 900, \quad 800$$

In this case, William could fake the data like this:

- Delete the historical entry with price 600.
- Delete the historical entry with price 800.

In this way, by deleting *just two* entries, the prices will look like they are converging to some value around 800 $\sim$ 900. Help William by writing a program that finds the least number of entries he has to fake for a given sequence!

☞ Among the attachments of this task you may find a template file `bitcoin2.*` with a sample incomplete implementation.

Input

The first line contains a single integer N, the number of Bitcoin price entries. The second line contains N positive integers V_i, the price of Bitcoin for each day.

Output

You need to write a single integer: the minimum possible number of historical entries to delete in order to obtain a converging sequence.

Constraints

- $1 \le N \le 300$.
- $1 \le V_i \le 1\,000\,000$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**20 points**]: $N \le 15$.
- **Subtask 3** [**15 points**]: Removing at most one price solves the problem.
- **Subtask 4** [**45 points**]: $N \le 100$.
- **Subtask 5** [**20 points**]: No additional limitations.

Examples

input.txt	output.txt
7 600 400 800 1500 700 900 800	2
9 1000 3000 2000 1000 1000 200 2500 1000 2000	4

Explanation

The **first sample case** is the example discussed before.

In the **second sample case** the solution is to keep prices `3000 200 2500 1000 2000`.

Irreversible fight (`breakup`)

After years of idyllic cooperation, Giorgio and William had a big fight on how to organize the final round of the OIS and eventually broke up, leaving the whole Italian Informatics Olympiads (OII) team shaken to the core.

The OII team consists of N people, where 0 is William and $N-1$ is Giorgio. Among them, there are M directed friendship bonds, so that person A_i values friendship with B_i as V_i worth. As usual with breakups, everyone now needs to choose on which side they want to be: should they blame William or Giorgio?

Of course, this choice is never done based on the actual facts happened. Instead, everyone chooses the side to which he is *more bonded*. More precisely, if S is a set of people and x is a person, the *total bond* $\text{bond}_S(x)$ of x to S is equal to the sum of values V_i of bonds between x and his friends in S. Every person chooses to support Giorgio or William trying to maximise the total bond with other people supporting the same person. After some transitory initial period, this will inevitably lead to a *stable splitting* $\langle G, W \rangle$. In such splitting, we will have that $0 \in W$, $N-1 \in G$ and:

- For each $x \in W$ apart from 0, $\text{bond}_W(x) \geq \text{bond}_G(x)$ (people in W do not want so switch sides);
- For each $x \in G$ apart from $N-1$, $\text{bond}_G(x) \geq \text{bond}_W(x)$ (people in G do not want so switch sides).

William knows these mechanics very well, and wants to exploit them to keep as many friends as possible, while leaving Giorgio to a grim isolation. Help William find the stable splitting $\langle G, W \rangle$ with the largest value for $\text{bond}_W(0) - \text{bond}_G(N-1)$!

☞ Among the attachments of this task you may find a template file `breakup.*` with a sample incomplete implementation.

Input

The first line contains integers N, M. The following M lines contain integers A_i, B_i, V_i.

Output

You need to write a string of N characters `‘W’` or `‘G’` depending on who the i-th person supports.

Constraints

- $2 \leq N \leq 100\,000$.
- $2N - 4 \leq M \leq 200\,000$.
- $0 \leq A_i, B_i \leq N-1$ and $A_i \neq B_i$ for each $i = 0 \ldots M-1$.
- $1 \leq V_i \leq 1000$ for each $i = 0 \ldots M-1$.

- There are no repeated friendship bonds, but the bond of x to y is possibly different from the bond of y to x. (love is not always paid!)
- Both Giorgio and William are bonded to all other people $1 \ldots N-2$.
- If there are multiple solutions, you can output any of them.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**30 points**]: Each person $1 \ldots N-2$ is bonded to a single other one.
- **Subtask 3** [**35 points**]: $N \leq 10$.
- **Subtask 4** [**25 points**]: $N \leq 1000$.
- **Subtask 5** [**10 points**]: No additional limitations.

Examples

input.txt	**output.txt**

4 8
0 1 13
0 2 42
3 2 25
3 1 30
1 0 10
2 0 14
1 2 20
2 3 15
	WGGG

4 8
0 1 13
0 2 42
3 2 25
3 1 30
1 0 10
2 0 14
1 2 20
2 3 14
	WWWG

Explanation

In the **first sample case**, the splitting is stable: $\text{bond}_G(2) = 15 \geq 14 = \text{bond}_W(2)$, $\text{bond}_G(1) = 20 \geq 10 = \text{bond}_W(1)$. Even though $\text{bond}_W(0) - \text{bond}_G(N-1) = -55$, there is no better splitting for William.

In the **second sample case**, the splitting is stable since $\text{bond}_W(2) = 14 \geq 14 = \text{bond}_G(2)$ and $\text{bond}_W(1) = 20 \geq 0 = \text{bond}_G(1)$, scoring an optimal $\text{bond}_W(0) - \text{bond}_G(N-1) = 55$.

Copyright notice (copyright)

As every respectable web developer, Luca could not properly celebrate last New Year's Eve. The reason is pretty obvious: he had to cope, like every year, with the annoying duty of updating the copyright notice in his website at the stroke of midnight.

Figure 1: Everybody in the world has the same problem: `http://updateyourfooter.com`.

☞ The copyright string has the following format (spaces are highlighted):

```
Copyright␣(C)␣YEARS.␣All␣rights␣reserved.
```

where `YEARS` is a comma-and-space separated list respecting these rules:

- Each element is either a single four-digit year (e.g., `2016`) or an interval of years (e.g., `2012-2016`).
- Years are listed in chronological order.
- A new year can be added to an already existing interval only when it is in temporal continuity. This means that `2012-2017` can be extended to `2012-2018` and, on the contrary, `2012-2016` must be updated as `2012-2016, 2018`.

In order to make sure that this task will not bother him in the future, Luca has decided to write a small program that does the job for him. Help him by writing a program which given the HTML source of a website updates the copyright for the current (2018) year!

☞ Among the attachments of this task you may find a template file `copyright.*` with a sample incomplete implementation.

Input

The first line contains the only integer N, the number of characters of the HTML source. The rest of the input consists in exactly N characters which represent the HTML source of the web page.

Output

The first line contains the number of characters of the HTML file *after* updating the copyright. In the following lines, you need to write *exactly* the same HTML source received in input with the updated copyright.

Constraints

- $41 \le N \le 1\,000\,000$.
- Years in the current copyright notice range from 1900 to 2017.
- There is exactly one copyright notice in the current HTML file and it respects all the aforementioned rules.
- The string S which represents the content of the HTML file is composed only by printable ASCII characters. S may contain “new-line” (`\n`) characters.
- The string S may not be a valid HTML source.
- The input file always ends with a new-line character, which is *included* in the total of N characters.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**30 points**]: S contains only the copyright notice.
- **Subtask 3** [**30 points**]: The copyright in S was updated in 2017.
- **Subtask 4** [**30 points**]: The copyright in S was not updated in 2017.
- **Subtask 5** [**10 points**]: No additional limitations.

Examples

```
input.txt                                    output.txt
-------------------------------------------- --------------------------------------------
180                                          180
<html>                                       <html>
  <head>                                       <head>
    <title> My website! </title>                 <title> My website! </title>
  </head>                                      </head>
  <body>                                       <body>
    I <3 HTML                                    I <3 HTML
  </body>                                      </body>
  <footer>                                     <footer>
    Copyright (C) 2010, 2012-2017.               Copyright (C) 2010, 2012-2018.
        All rights reserved.                         All rights reserved.
  </footer>                                    </footer>
</html>                                      </html>
-------------------------------------------- --------------------------------------------
164                                          170
<html>                                       <html>
  <head>                                       <head>
    <title> My website! </title>                 <title> My website! </title>
  </head>                                      </head>
  <body>                                       <body>
    Internet seems cool...                       Internet seems cool...
    Copyright (C) 1994-1996. All                 Copyright (C) 1994-1996, 2018.
       rights reserved.                              All rights reserved.
  </body>                                      </body>
</html>                                      </html>
```

Explanation

In the **first sample case** the most recent copyright was specified for the interval of years from 2012 to 2017. Thus, we extend the interval to 2018 and it becomes 2012-2018.

In the **second sample case** the copyright was specified for the interval of years from 1994 to 1996. As 2018 is not consecutive to 1996, it must be added as a separate year.

Desk ordering (desk)

William is notoriously messy, and his desk is a wild arrangement of paper piles. As usual, he is starting to write the OIS problems at the very last minute, but there is a major problem: his laptop does not fit on the desk because of all the piles!

Figure 1: William hard working at his desk.

William's desk is N *piles of paper* wide (each of them consisting in H_i sheets) and his laptop is L piles wide. In order to fit, the laptop needs to be put *horizontally* on a contiguous sequence of L piles, which is possible only if the two highest piles among them differ by at most one (the laptop need two supports to stay still). Obviously, the piles contain very important documents he cannot throw away, or even shuffle too much. Thus, the only operation he can do is moving **once** the top K sheets of a pile to another. What is the minimum number K of sheets he need to move?

☞ Among the attachments of this task you may find a template file `desk.*` with a sample incomplete implementation.

Input

The first line contains the integers N and L. The second line contains N integers H_i.

Output

You need to write a single line with an integer: the minimum number of sheets to move.

Constraints

- $2 \leq L \leq N \leq 1\,000\,000$.
- $0 \leq H_i \leq 10^9$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**30 points**]: $L = N$.
- **Subtask 3** [**20 points**]: $N \leq 1\,000$.
- **Subtask 4** [**15 points**]: $N \leq 100\,000$ and all the H_i are distinct.
- **Subtask 5** [**15 points**]: $N \leq 100\,000$.
- **Subtask 6** [**20 points**]: No additional limitations.

Examples

input.txt	output.txt
5 3 5 15 22 10 8	3
4 4 9 13 10 15	1

Explanation

In the **first sample case**, William can move 3 sheets from the third pile to the second, and then place the laptop on `18 19 10`.

In the **second sample case**, William can move 1 sheet from the last pile to the second, and then place the laptop on `9 14 10 14`.

Game simulator (`dominion`)

Giorgio is notoriously fond of board games, and has a particular devotion for the card game *Dominion*. However, he is recently getting skeptical about the mechanics: too often, the best strategy is just to greedily buy treasure and victory cards. There is only one way to prove his conjecture: writing a game simulator!

Figure 1: Some cards used by the game.

You are given the description of N different cards, numbered from 0 to $N-1$. Each card is available in A_i copies, has a price P_i in coins, a revenue R_i in coins, and a value V_i in points.

You are also given the description of a starting deck, consisting of M cards C_i among the ones described, indexed from 0 (top deck) to $M-1$ (bottom deck). The simulated game should consist in T turns as follows:

- draw the top 5 cards of the deck (maintaining their relative order);
- sum the revenues of the cards drawn obtaining a number R;
- buy a copy of the most expensive available card you can afford with R coins, putting it at the bottom of the deck;
- discard the 5 cards drawn at the beginning, putting them at the bottom of the deck while maintaining their relative order.

At the end of T turns, the resulting deck is scored by summing the values V_i in points of each card in it. Help Giorgio computing the total value after T turns!

☞ Among the attachments of this task you may find a template file `dominion.*` with a sample incomplete implementation.

Input

The first line contains the integers N, M, T. The second line contains M integers C_i. The following N lines contain integers A_i, P_i, R_i, V_i.

Output

You need to write a single line with an integer: the total value in the deck after T turns.

Constraints

- $1 \leq N \leq 100\,000$.
- $5 \leq M \leq 100\,000$.
- $1 \leq T \leq 1\,000\,000$.
- $0 \leq C_i \leq N - 1$ for each $i = 0 \ldots M - 1$.
- $0 \leq A_i, P_i, R_i \leq 100\,000\,000$ for each $i = 0 \ldots N - 1$.
- $-10 \leq V_i \leq 10$ for each $i = 0 \ldots N - 1$.
- Prices P_i are all distinct, and there is always a card with $P_i = 0$, $A_i \geq T$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**40 points**]: $N, M, T \leq 1000$.
- **Subtask 3** [**30 points**]: $N \leq 100$.
- **Subtask 4** [**20 points**]: $A_i = T$ for each i.
- **Subtask 5** [**10 points**]: No additional limitations.

Examples

input.txt	output.txt

`3 5 2`
`1 2 0 2 1`
`1 20 10 -1`
`9 0  2   0`
`5 10 5   1`
	`1`

`8 10 3`
`0 4 0 0 4 4 0 0 0 0`
`10 0 1 0`
`10 3 2 0`
`10 6 3 0`
`10 9 5 0`
`10 2 0 1`
`10 5 0 3`
`10 8 0 6`
`10 11 0 10`
	`6`

Explanation

In the **first sample case**, we first draw cards `1 2 0 2 1` which provide total revenue $R = 24$. The most expensive available card we can afford is number 0, so after the first turn the deck is `0 1 2 0 2 1`. In the second and last turn, we draw cards `0 1 2 0 2` which provide revenue $R = 32$. Since card 0 is not available any more, the most expensive card we can afford is number 2. The final deck is thus `1 2 0 1 2 0 2` which scores $1 - 1 + 1 - 1 + 1 = 1$ points.

In the **second sample case**, the deck evolves as follows (cards drawn are underlined, cards bought are circled):

$$\begin{array}{c}
\underline{0\ 4\ 0\ 0\ 4}\ 4\ 0\ 0\ 0\ 0 \\
\underline{4\ 0\ 0\ 0\ 0}\ \textcircled{1}\ 0\ 4\ 0\ 0\ 4 \\
\underline{1\ 0\ 4\ 0\ 0}\ 4\ \textcircled{1}\ 4\ 0\ 0\ 0\ 0 \\
\underline{4\ 1\ 4\ 0\ 0}\ 0\ 0\ \textcircled{5}\ 1\ 0\ 4\ 0\ 0
\end{array}$$

Project groups (`projectgroup`)

Group projects are perhaps one of the most hated things in the academic world. Luca carefully studied the psychology behind this issue, and he believes he found the root cause: students hate group projects because they are forced to work with someone they don't really get along with.

EVERY GROUP PROJECT
DOES 99% OF THE WORK
HAS NO IDEA WHAT'S GOING ON THE WHOLE TIME
SAYS HE'S GOING TO HELP BUT HE'S NOT
DISAPPEAR AT THE VERY BEGINNING AND DOESN'T SHOW UP AGAIN TIL THE VERY END
IN SCHOOL YOU HAVE EVER DONE

Figure 1: Common issues in a project group.

Wouldn't it be great if everyone could be in a group only made by friends? Creating such groups manually is quite a challenging task: help Luca by writing a program that will serve this purpose for the next academic projects!

There are N students and M friendship relations between two students (friendship is symmetric). Try to form as many groups of *three* students as you can, respecting their friendship relations.

☞ Among the attachments of this task you may find a template file `projectgroup.*` with a sample incomplete implementation.

Input

The first line contains two integers, N and M. The following M lines contain each a pair of integers A_i and B_i, meaning that students A_i and B_i are friends.

Output

You need to write $G+1$ lines. The first line must contain the only integer G, the number of possible groups you have identified. The next G lines must contain three space-separated integers each, the students in a group you have found.

Constraints

- $3 \leq N \leq 1000$.
- $3 \leq M \leq 2000$.
- $0 \leq A_i, B_i < N$ for each $i = 0 \ldots N-1$.
- A student can be part of only **one** group.
- Groups of students can be printed in any order, as students in a group can be printed in any order (i.e., you can print either `0 1 2` or `2 0 1`).

Scoring

Your program will be tested against several test cases grouped in subtasks. The score in each subtask will be calculated as the **minimum** score obtained in any of its test cases, multiplied by the value of the subtask. The score in a test case will be equal to:

$$\max(2\frac{G_{\text{found}}}{G_{\max}} - 1, 0)$$

where G_{found} is the number of groups you correctly identify and $G_{\max}$ is the maximum number of groups that can be identified.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**15 points**]: $N \leq 5$.
- **Subtask 3** [**15 points**]: $N = M$.
- **Subtask 4** [**40 points**]: $N \leq 50$, $M \leq 100$.
- **Subtask 5** [**30 points**]: No additional limitations.

Examples

input.txt	**output.txt**

`5 6`
`0 1`
`0 2`
`2 1`
`2 3`
`4 2`
`4 3`
	`1`
	`0 1 2`

`6 8`
`0 1`
`2 0`
`3 1`
`4 3`
`3 5`
`4 5`
`3 2`
`1 2`
	`2`
	`4 5 3`
	`0 1 2`

Explanation

In the **first sample case** the maximum number of groups that can be created is just one: either formed by students 0, 1 and 2 or by students 2, 3 and 4.

In the **second sample case** two groups can be formed, putting students 0, 1 and 2 in a group and students 3, 4 and 5 in another.

Water park (`waterslide`)

There's a new water park in the city and Edoardo, the owner, cannot wait to open! However, he is in doubt on where to put the life guards.

To access a pool you have to take a ride (that is, a path from the launch pad) through some slides into a pool. While going down, you may encounter a junction. Note that there could be junctions where many slides enter and many exit: in this case, you will choose one of the many exits *uniformly at random*. Obviously, there cannot be any cyclic paths since water always flows down.

Figure 1: A nice waterslide.

Edoardo knows very well that most of the accidents happen where the most people are. You are given a description of the junctions and slides connecting them. Help Edoardo find which is the ending pool where most of the people will arrive!

☞ Among the attachments of this task you may find a template file `waterslide.*` with a sample incomplete implementation.

Input

The first line contains the number of junctions N, the number of slides M and the number of pools P. The launch pad has index 0, the ending pools have a number between $N - P$ and $N - 1$. The next M lines contain two integers A_i, B_i each: the starting and ending junction of each slide (respectively).

Output

You need to write a single line with an integer: the index of the pool where most of the people will arrive.

Constraints

- $1 \leq P < N \leq 100\,000$.
- $1 \leq M \leq 200\,000$.
- $0 \leq A_i, B_i < N$ for all $i = 0 \ldots M - 1$.

- The network of slides is always connected.
- There are no slides starting from pools.
- Every path has to eventually reach a pool.
- If multiple equivalent solutions are possible, you can print any one of them.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**15 points**]: Exactly one slide enter and one exits each junction.
- **Subtask 3** [**30 points**]: $N \leq 1000$.
- **Subtask 4** [**30 points**]: Exactly one slide enter each junction.[1]
- **Subtask 5** [**25 points**]: No additional limitations.

Examples

`input.txt`	`output.txt`

`5 6 2`
`0 1`
`0 2`
`1 2`
`2 3`
`2 4`
`0 4`
	`4`

`7 7 3`
`0 6`
`0 1`
`1 5`
`1 2`
`2 4`
`2 3`
`3 4`
	`6`

Explanation

In the **first sample case**, one third of the people will arrive at pool 3 and two thirds at pool 4.

In the **second sample case**, $\frac{1}{4}$ will arrive at pool 4, $\frac{1}{4}$ at pool 5 and $\frac{1}{2}$ at pool 6.

[1]For this purpose, pools do not count as junctions, so many slides might enter in them.