2018 ICPC Asia Singapore Regional Contest

Problem A. Largest Triangle

Time limit	2000 ms
Mem limit	1048576 kB
OS	Linux

Given N points on a 2-dimensional space, determine the area of the largest triangle that can be formed using 3 of those N points. If there is no triangle that can be formed, the answer is 0.

Input

The first line contains an integer N ($3 \leq N \leq 5\,000$) denoting the number of points. Each of the next N lines contains two integers x and y ($0 \leq x, y \leq 4 \cdot 10^7$). There are **no** specific constraints on these N points, i.e. the points are not necessarily distinct, the points are not given in specific order, there may be 3 or more collinear points, etc.

Output

Print the answer in one line. Your answer should have an absolute error of at most 10^{-5}.

Sample 1

Input	Output
7 0 0 0 5 7 7 0 10 0 0 20 0 10 10	100.00000

Problem B. Hoppers

Time limit	2000 ms
Mem limit	1048576 kB
OS	Linux

You are part of an elite hacking group who has just invented a new type of malware, called the hoppers. Hoppers can operate in any high-security computer networks; however, they can only propagate themselves in a very limited way.

A computer network is a set of hosts V ($|V| = N$) and direct links E ($|E| = M$). Each direct link connects two different hosts, allowing them to communicate in both directions. If a host u is infected with a hopper, then it will propagate itself to all hosts W where W is the set of hosts which are connected to any neighbor of u. Formally, $W = \{w \mid \{(u, v), (v, w)\} \subseteq E\}$. The newly-infected host w can now proceed to infect other hosts in the same manner.

You are trying to infect all hosts in a particular high-security network by compromising a single host. Of course, a compromised host will be infected with a hopper. Though it may be an impossible mission, you realize that you can trick the administrator into installing new direct links between hosts by submitting IT requests.

Too many requests may arouse suspicion. Find the minimum number of IT requests you have to submit to ensure that there exists a single host such that if it is infected with a hopper, then it will propagate to the entire network.

Input

The first line contains two integers N and M, where $3 \leq N \leq 5 \cdot 10^5$ and $2 \leq M \leq 5 \cdot 10^5$. The next M lines contain two integers u and v ($1 \leq u, v \leq N$), representing a direct link between hosts u and v. There is at most one direct link between any pair of hosts, and no host is directly linked to itself.

Output

The output contains an integer in a line representing the minimum number of IT requests you must send.

Explanation

In the first example, there are $N = 3$ hosts and $M = 2$ direct links $\{(1, 2), (2, 3)\}$. In this network, there is no way to spread the hopper to all hosts by compromising only a single

host, e.g., if we compromise host 1, then only hosts 1 and 3 are infected, etc. We need to submit one IT request to connect $(1, 3)$ to achieve our goal. By doing so, if we compromise host 1, all hosts 1, 2, and 3 are infected. $1 \rightarrow 3 \rightarrow 2$ (thus, host 2 is infected). $1 \rightarrow 2 \rightarrow 3$ (thus, host 3 is infected).

Sample 1

Input	Output
3 2 1 2 2 3	1

Sample 2

Input	Output
5 10 1 2 2 3 3 4 4 5 5 1 1 3 2 4 3 5 4 1 5 2	0

Sample 3

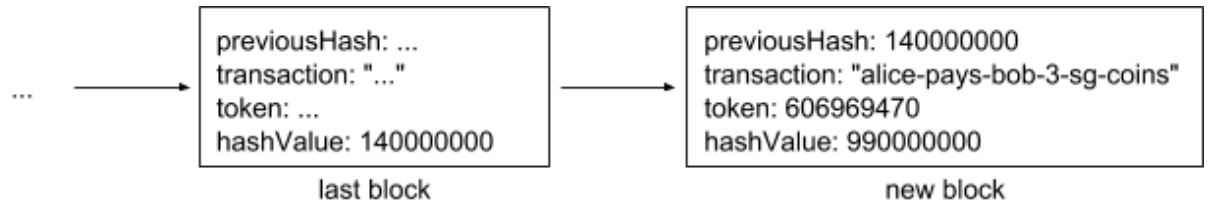
Input	Output
12 13 2 3 3 4 4 5 5 2 6 7 7 8 8 9 9 10 10 7 9 12 12 11 11 6 12 7	3

Problem C. SG Coin

Time limit	1000 ms
Mem limit	1048576 kB
OS	Linux

There is a new cryptocurrency called SG coin. Alice purchases an item from Bob and pays in SG coin. The transaction is recorded in a blockchain. A blockchain is a chain of blocks where each block contains the hash value of the previous block, a transaction string, a token, and the hash value of the block (a positive integer with 7 trailing zeros).

Example: Suppose the hash value of the last block in the blockchain is 140 000 000 and the next transaction string to be recorded is "`alice-pays-bob-3-sg-coins`". The new block containing the transaction is appended (chained) to the last block in the blockchain like this:



The token in the new block must be set to a number between 0 to $10^9 - 1$ such that the hashValue of the new block is positive and has 7 trailing zeros. If the token is 606 969 470 then the hashValue of the new block is computed as follow:

```
H(140000000,"alice-pays-bob-3-sg-coins",606969470)=990000000
```

The function `H` is given below (in C++ and Java format only). There can be many other valid values for the token that produces a valid hashValue (with 7 trailing zeros), e.g.,

```
H(140000000,"alice-pays-bob-3-sg-coins",306969470)=690000000
```

In a distributed system, a new block can be added to the blockchain by anyone and then broadcast to everyone. Everyone will receive the new block and verify that it is valid (i.e., the last block's hashValue, transaction string, and token of the new block produces a hashValue with 7 trailing zeros). Everyone keeps track of the longest chain and only appends a new block to the longest chain that they have.

Charlie has a lot of computing power and he sees an opportunity to attack the distributed blockchain system by generating new blocks rapidly, thus producing the longest chain quickly. This allows him to monopolize new transactions to be appended to the blockchain.

Let's see an attack scenario. Alice pays Bob 3 SG coins for purchasing an item, she puts the transaction in the blockchain and broadcasts. Bob sees the transaction in the blockchain and then gives the purchased item to Alice. Charlie then comes in to disrupt this transaction by creating two consecutive new blocks and broadcasting. Everyone sees the two new blocks from Charlie and keeps them, discarding the new block from Alice (which is shorter).

You are Charlie and you want to make life harder for Alice and Bob.

The hash function in C++:

```
long long H(long long previousHash, string &transaction,
            long long token) {
  long long v = previousHash;
  for (int i = 0; i < transaction.length(); i++) {
    v = (v * 31 + transaction[i]) % 1000000007;
  }
  return (v * 7 + token) % 1000000007;
}
```

The hash function in Java:

```
static long H(long previousHash, String transaction,
              long token) {
  long v = previousHash;
  for (int i = 0; i < transaction.length(); i++) {
    v = (v * 31 + transaction.charAt(i)) % 1000000007;
  }
  return (v * 7 + token) % 1000000007;
}
```

Input

The first line is the hashValue of the last block in the blockchain. The hashValue is between 0 to 1 000 000 006 inclusive and guaranteed to have 7 trailing zeros.

Output

Produce two new blocks A and B as fast as possible. A should be chained to the last block (from the given input) and B should be chained to A. Print the transaction strings and the tokens of the two new blocks.

The first line contains the transaction string and the token for A separated by a space and the second line is for B. The transaction string is non-empty and can be anything you want containing only lowercase English characters, digits (0–9), or hyphen with at most 100 characters. The token must be an integer between 0 to $10^9 - 1$.

Explanation

The last three blocks have hashValues of: $140\,000\,000 \rightarrow 930\,000\,000 \rightarrow 730\,000\,000$.

Sample 1

Input	Output
140000000	charlie-pays-to-eve-9-sg-coins 218216710 icpc-sg-2018-at-nus 620658977

Problem D. Bitwise

Time limit	3000 ms
Mem limit	1048576 kB
OS	Linux

Barr the Bear is playing the game *Bits* with Swanky Shen!

Bits is a very simple game. At the start, a circle of N non-negative integers A_1, A_2, A_3, ..., A_N is shown to both players. That is, to the left of the integer A_i is the integer A_{i-1} if $i > 1$; and the integer A_N otherwise. To the right of the integer A_i is the integer A_{i+1} if $i < N$; and the integer A_1 otherwise. Also an integer K is given to both players.

To win this game, one must divide the circle of integers into exactly K contiguous non-empty sections, such that the bitwise `AND` of the *powers* of all sections is maximized. The power of a contiguous section of integers is the bitwise `OR` of all integers in that section.

Barr the Bear is lazy and knows that you are wise with bits. Hence, he has hired you to help him to win the game!

Note: The binary bitwise operators `OR` and `AND` operate on the base-2 representation of the integers and correspond to the operators `|` and `&` respectively in C++ or Java.

Input

The first line contains integers N and K ($1 \leq K \leq N \leq 5 \cdot 10^5$), namely the number of integers and the number of contiguous non-empty sections required.

The next line contains N integers, the i^{th} of which is the integer A_i ($0 \leq A_i \leq 10^9$).

Output

Output a single integer in one line: The maximum bitwise `AND` of the powers of the sections in an optimal division of the circle of integers.

Explanation

In the first sample, the circle is $(2, 3, 4, 1)$. A possible division is $(3, 4)$ and $(1, 2)$. $(3, 4)$ has power 7 and $(1, 2)$ has power 3. The bitwise `AND` of 7 and 3 is 3. Note that a section can possibly wrap around the circle.

In the second sample, a possible division is $(2, 2, 4)$, (4), $(4, 2)$. The sections' powers are 6, 4 and 6 respectively, which have bitwise `AND` of 4. Note that we require the sections to be

contiguous integers, so the division $(2, 4), (2, 4), (2, 4)$ is not permissible.

In the third sample, we can only have one section. This section will have all the integers, and thus have power 3.

Sample 1

Input	Output
4 2 2 3 4 1	3

Sample 2

Input	Output
6 3 2 2 2 4 4 4	4

Sample 3

Input	Output
4 1 0 1 2 3	3

Problem E. Magical String

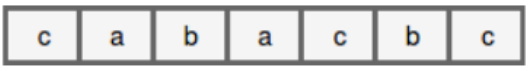
Time limit	1000 ms
Mem limit	1048576 kB
OS	Linux

Lajuk loves to play with magic, especially with a magical string.

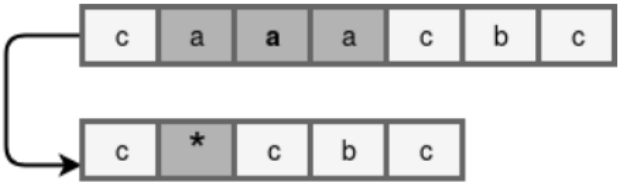
A magical string consists of N lowercase English characters which has no substring containing 3 or more identical characters, e.g., "`aabbaacbb`", "`icpc`", "`asia`", "`singapore`" are magical strings. On the other hand, "`waaaah`" and "`singapooore`" are not magical strings as they contain "`aaaa`" (in the first example) and "`ooo`" (in the second example).

We can convert any character in the magical string into any other character as long as the direct consequence of the conversion results in some substring containing 3 or more identical characters. When this happens, that particular substring which contains identical characters of the maximum length will magically transform into an asterisk ('`*`') character. This is the reason why it is called a magical string.

For example, consider a magical string $S[1, 7] =$ "`cabacbc`".



If we convert the third character ('`b`') into '`a`', then the substring $S[2, 4]$ becomes "`aaa`" which contains 3 identical characters. This substring is then transformed into an asterisk '`*`' and the string becomes "`c*cbc`".



Note that the asterisk character does not have any magical power, i.e. we cannot convert any asterisk character, and no asterisk character will ever be transformed (e.g., when there are 3 consecutive asterisk characters, nothing happens). Basically, asterisk characters are dead.

Now, let's move on to the fun part. Lajuk gives you a magical string S and an integer K. Your task is to convert at most K characters (one at a time) in the string such that the total number of characters which are transformed into asterisks is as large as possible.

In the example above ("cabacbc"), if $K = 2$, then the output is 6, i.e. convert $S[3]$ from 'b' into 'a' (3 characters turn into an asterisk), and convert $S[6]$ from 'b' into 'c' (3 characters turn into an asterisk).

Input

Input begins with an integer $T (1 \le T \le 50)$ representing the number of cases. Each case contains a magical string $S (1 \le |S| \le 1\,000)$ followed by an integer $K (1 \le K \le 1\,000)$ (separated by a single space) in a line. The string S contains only lowercase English characters ['a'..'z'].

Output

For each case, output in a line containing one integer representing the maximum number of characters in the given magical string which can be turned into asterisks by converting at most K characters.

Explanation

For the second case in the sample input, convert $S[3]$ from 'b' into 'a'. "aabaacad" $\rightarrow$ "aaaaacad" $\rightarrow$ "*cad" (5 characters turn into asterisks). For the third case in the sample input, convert $S[6]$ from 'c' into 'a'. "aabaacad" $\rightarrow$ "aabaaaad" $\rightarrow$ "aab*d" (4 characters turn into asterisks). Then, convert $S[3]$ from 'b' into 'a'. "aab*d" $\rightarrow$ "aaa*d" $\rightarrow$ "**d" (3 more characters turn into asterisks).

Sample 1

Input	Output
3 cabacbc 2 aabaacad 1 aabaacad 2	6 5 7

Problem F. Wi Know

Time limit	2000 ms
Mem limit	1048576 kB
OS	Linux

You are the boss of Wi Know, an upstanding company in information theory, especially in message encryption.

The counter-counter-intelligence branch of your upstanding company managed to intercept a message sent by the counter-intelligence agency of the local Helsinkian government. This message is, of course, of utmost importance, and its content can probably be used for the "greater good" later. The message is a sequence S of N positive integers not greater than N, indexed from 1 to N. Let S_i be the i^{th} integer of S.

As the first step to mine useful information from this message, you first have to find patterns in it. At the moment, the pattern we're interested in is whether there exists two different integers A and B such that the pattern A, B, A, B appears as a (not necessarily contiguous) subsequence of the original message. That is, whether there exists four indices $1 \leq c < d < e < f \leq N$ such that $S_c = S_e$, $S_d = S_f$, and $S_c \neq S_d$.

Your task is to find such a pattern, if any, and print both A and B. If there are multiple such pairs (A, B), find the lexicographically smallest one. That is, if there are multiple such pairs (A, B), print the one whose A is minimized. If there are still multiple such patterns, print the one whose B is minimized.

Input

The first line contains a non-negative integer $4 \leq N \leq 400\,000$, giving the number of integers in S. Thereafter follow N lines, the i^{th} line contains a single integer $1 \leq S_i \leq N$.

Output

If $A \neq B$ exists and the pattern A, B, A, B appears as a subsequence of S, you should print two integers A and B on a single line separated by a single space, denoting the lexicographically smallest pair of (A, B) as described in the problem statement. Otherwise, if there is no such pair, you should print a single integer -1.

Sample 1

Input	Output
8 1 3 2 4 1 5 2 4	1 2

Sample 2

Input	Output
8 1 2 3 4 5 6 7 1	-1

Sample 3

Input	Output
4 2 1 2 1	2 1

Problem G. Rectangular City

Time limit	1000 ms
Mem limit	1048576 kB
OS	Linux

The festive season is coming and Rectangular City is going to organize N different events for N consecutive days. Rectangular City, as its name suggests, consists of $R \times C$ blocks. Each event will be hosted on a rectangular area within the city. In order to lighten-up the events, the mayor decides to spend some of the budgets to decorate K blocks on the condition that every decorated block must be used in all N events.

The city council is interested in the number of different ways to organize all N events such that it is possible to choose K different blocks to decorate. Two ways are considered different if there exists a day in which the corresponding event is hosted on a different area. Since the answer can be very large, modulo the output by $1\,000\,000\,007$.

Input

The input contains a line with four integers N, R, C, and K ($1 \leq N \leq 10^6$; $1 \leq R, C \leq 5\,000$; $1 \leq K \leq R \cdot C$).

Output

The output contains an integer representing the total number of different ways to organize all events, modulo $1\,000\,000\,007$.

Explanation

In the Sample Input, the city has 2×3 blocks, and the mayor decides to decorate 4 blocks for 2 different events (days). There are a total of 7 ways to organize the events.

```
#1   Day 1: oo.     Day 2: oo.    Decorated: **.
            oo.            oo.               **.

#2   Day 1: oo.     Day 2: ooo    Decorated: **.
            oo.            ooo               **.

#3   Day 1: .oo     Day 2: .oo    Decorated: .**
            .oo            .oo               .**

#4   Day 1: .oo     Day 2: ooo    Decorated: .**
            .oo            ooo               .**

#5   Day 1: ooo     Day 2: oo.    Decorated: **.
```

```
            ooo           oo.               **.

  #6   Day 1: ooo     Day 2: .oo    Decorated: .**
            ooo           .oo               .**

  #7   Day 1: ooo     Day 2: ooo    Decorated: **.
            ooo           ooo               **.
```

Note that, in the example above, configuration #7 uses the whole city for both events, thus, the decorated blocks can be any K blocks in the city. The shown decorated blocks are only given as examples.

Sample 1

Input	Output
2 2 3 4	7

Problem H. Sliding Blocks

Time limit	4000 ms
Mem limit	1048576 kB
OS	Linux

Sliding Blocks is an interesting puzzle game which can be played on most mobile devices. In this game, you are given an $N \times M$ board with exactly one cell containing a block; all the other cells are empty. You can add a block to the board by sliding the new block from the edge of the board. The new block slides until it bumps into an existing block and it stops.

For example, let the board be 5×6 and there is a block at $(2, 3)$–the 2^{nd} row and 3^{rd} column.

```
......
..X...
......
......
......
```

If we slide a new block at the third column from the bottom edge, then the new block stops at $(3, 3)$ as it bumps into the block at $(2, 3)$.

```
......     ......
..X...     ..X...
......     ..X...
......     ......
......     ......
  ^
```

If we slide a new block at the second row from the right edge, at the fourth column from the top edge, and at the third row from the left edge, respectively, then the new blocks stop at $(2, 4)$, $(1, 4)$, and $(3, 2)$.

```
              v
......     ......     ...X..     ...X..
..X...<    ..XX..     ..XX..     ..XX..
..X...     ..X...    >..X...     .XX...
......     ......     ......     ......
......     ......     ......     ......
```

Note that a move which does not add a new block into the board is considered illegal, e.g. sliding a new block at the fourth column from the top edge or at the fifth row from the right edge in the last board of the previous example.

```
   v  [illegal]
...X..                ...X..
..XX..                ..XX..
```

```
.XX...              .XX...
......              ......
......              ......<  [illegal]
```

In each level of the game, you are given a target board which is your goal; in other words, you have to add zero or more new blocks such that your board becomes exactly the same as the target board. It is guaranteed that the target board contains blocks which form a tree (as in graph theory), i.e. from any block, there exists a unique simple path to any other block which only passes through other blocks. A block is connected to another block if they are next to each other.

The game seems good and you (the developer) are ready to enjoy the success. Until one day, an incoming user report states (and unfortunately, also proves) that a certain puzzle level is impossible to be solved. For example, consider the following initial board and target board.

```
....    XXXX
....    X..X
X...    X.XX
```

The right target board cannot be achieved from the initial board on the left as there is no way to add a new block at $(3, 3)$.

Now, you have to re-evaluate every puzzle in the game properly. Given the initial and target board configurations, determine whether it is possible to make your initial board become exactly the same as the target board.

Input

The first line contains three integers N, M, and B ($1 \leq N, M, B \leq 4 \cdot 10^5$) representing the board size (number of rows and columns) and the number of blocks in the target board, respectively. The next B lines each contains two integers r and c ($1 \leq r \leq N; 1 \leq c \leq M$) representing a block at the r^{th} row and c^{th} column in the target board. The first block which is given in the input (second row) already exists in your initial board. The target board configuration is guaranteed to contains only blocks which form a tree.

Output

The output contains a line denoting whether it is possible to make your board becomes exactly the same as the target board. If it is not possible, output "`impossible`" (without quotes) in one line. If it is possible, output "`possible`" (without quotes) in one line followed by $B - 1$ lines representing the moves which can achieve the goal. Each move is represented by a character c (c \in \{ \texttt{<},\texttt{>},\texttt{^},\texttt{v}\}) and an integer k, separated by a single space.

- < denotes sliding the new block from the right edge of the board towards the left.
- > denotes sliding the new block from the left edge of the board towards the right.
- \texttt{^} denotes sliding the new block from the bottom edge of the board upward.
- v denotes sliding the new block from the top edge of the board downward.

The integer k denotes the row or column in which the move is performed on. If $c \in \{<, >\}$, then the move is performed on the k^{th} row and $1 \leq k \leq N$. If c \in \{ \texttt{^},\texttt{v}\} , then the move is performed on the k^{th} column $1 \leq k \leq M$. Note that the moves are performed in the output order. If there is more than one move configuration which can achieve the goal, you may output any one of them.

Sample 1

Input	Output
3 4 6 1 1 1 2 2 2 2 3 3 3 3 4	possible < 1 ^ 2 < 2 ^ 3 < 3

Sample 2

Input	Output
3 4 9 3 1 2 1 1 1 1 2 1 3 1 4 2 4 3 4 3 3	impossible

Problem I. Prolonged Password

Time limit	2000 ms
Mem limit	1048576 kB
OS	Linux

Kang the Penguin has forgotten some letters of his password, help him figure them out!

Of course, Kang knows that something as important as a password should be easy to remember, but it also cannot be too short. Thus, he knows that he originally decided to generate his password in the following manner. First, he starts with some non-empty string S, consisting of lowercase letters only. Then, he chooses 26 non-empty strings $T_a, T_b, \ldots, T_z$, each consisting of at least two lowercase English letters. He defines a function f, which converts each character i to its corresponding string T_i and concatenates the results. For example, if T_a was "`abc`" and T_b was "`cba`", applying f to "`aba`" would result in the string "`abccbaabc`".

Now, he applies f repeatedly to S, applying it K times in total. The final result is his password $P = f^K(S)$.

While he remembers most of his password, he has forgotten M of the letters. The i^{th} letter that he has forgotten is in position m_i in the string P. It is guaranteed that each m_i will be less than or equal to the final length of the password $|P|$. Help Kang to figure out what the forgotten letters are!

Input

The 1^{st} line of the input contains a single lowercase string S, where $1 \leq |S| \leq 1\,000\,000$.

The 2^{nd} line of the input contains 13 strings $T_a, T_b, \ldots, T_m$, separated by spaces, where $2 \leq |T_a|, |T_b|, \ldots, |T_m| \leq 50$.

The 3^{rd} line of the input contains 13 strings $T_n, T_o, \ldots, T_z$, separated by spaces, where $2 \leq |T_n|, |T_o|, \ldots, |T_z| \leq 50$.

The strings $T_a, T_b, \ldots, T_z$ each contains only lowercase English characters (`a`–`z`).

The 4^{th} line of the input contains a single integer K, where $1 \leq K \leq 10^{15}$.

The 5^{th} line of the input contains a single integer M, where $1 \leq M \leq 1\,000$.

The 6^{th} line of the input contains M integers, the i^{th} of which is the integer m_i, where $1 \leq m_i \leq \min(|f^K(S)|, 10^{15})$.

Output

Output M lines, each containing a single lowercase character. The i^{th} line of the output should contain the letter in the m_i^{th} position of the password P.

Sample 1

Input	Output
abca bc cd da dd ee ff gg hh ii jj kk ll mm nn oo pp qq rr ss tt uu vv ww xx yy zz 1 2 1 8	b c

Sample 2

Input	Output
ab ba ab cc dd ee ff gg hh ii jj kk ll mm nn oo pp qq rr ss tt uu vv ww xx yy zz 2 2 1 8	a b

Problem J. Free Food

Time limit	1000 ms
Mem limit	1048576 kB
OS	Linux

Do you know what attracts almost any college student to participate in an event? Yes, free food. It doesn't matter whether the event involves a long (sometimes boring) seminar. As long as free food is served for the event, then students will surely come.

Suppose there are N events to be held this year. The i^{th} event is scheduled from day s_i to day t_i, and free food is served for that event every day from day s_i to day t_i (inclusive). Your task in this problem is to find out how many days there are in which free food is served by at least one event.

For example, let there be $N = 3$ events. The first event is held from day 10 to 14, the second event is held from day 13 to 17, and the third event is held from day 25 to 26. The days in which free food is served by at least one event are $10, 11, 12, 13, 14, 15, 16, 17, 25, 26$, for a total of 10 days. Note that both events serve free food on days 13 and 14.

Input

The first line contains an integer N ($1 \leq N \leq 100$) denoting the number of events. Each of the next N lines contains two integers s_i and t_i ($1 \leq s_i \leq t_i \leq 365$) denoting that the i^{th} event will be held from s_i to t_i (inclusive), and free food is served for all of those days.

Output

The output contains an integer denoting the number of days in which free food is served by at least one event.

Sample 1

Input	Output
3 10 14 13 17 25 26	10

Sample 2

Input	Output
2 1 365 20 28	365

Sample 3

Input	Output
4 29 29 48 48 102 102 94 94	4

Problem K. Conveyor Belts

Time limit	2000 ms
Mem limit	1048576 kB
OS	Linux

Your factory has N junctions (numbered from 1 to N) connected by M conveyor belts. Each conveyor belt transports any product automatically from one junction to another junction in exactly one minute. Note that each conveyor belt only works in one direction. There can be more than one conveyor belt connecting two junctions, and there can be a conveyor belt connecting a junction to itself.

There are K producers (machines which produce the products) located at the first K junctions, i.e. junctions $1, 2, \ldots, K$. The producer at junction j produces an product each minute $(x \cdot K + j)$ for all integers $x \geq 0$ and $j = 1, 2, \ldots, K$. All products are transported immediately via the conveyor belts to the warehouse at junction N, except for those produced at junction N (if any). Items produced at junction N are directly delivered to the warehouse (there is no need to use the conveyor belts).

At each junction, there is a robot deciding which conveyor belts the incoming product should go to in a negligible time (instantly). The robots can be programmed such that all products produced by a producer are always delivered to the warehouse via the same route. Once the robots are programmed, the routing can no longer be changed. Items from different producers may have the same or different routes.

A prudent potential investor comes and wants to inspect the factory before making any decision. You want to show to the potential investor that your factory employs a good risk management policy. Thus, you want to make sure that each conveyor belt only transports at most one product at any time; i.e. two products cannot be on the same conveyor belt at the same time. On the other hand, there is no limit on the number of products at the junctions (the robots have a lot of arms!). To achieve this, you may turn off zero or more producers, but you still want to maximize the production, hence, this problem.

Find the maximum number of producers that can be left running such that all the produced products can be delivered to the warehouse and each conveyor belt transports at most 1 product at any time.

Input

The first line contains three integers N, K, and M ($1 \leq K \leq N \leq 300$; $0 \leq M \leq 1\,000$) representing the number of junctions, the number of producers, and the number of

conveyor belts, respectively.

The next M lines, each contains two integers a and b ($1 \leq a, b \leq N$) representing a conveyor belt connecting junction a and junction b with the direction from a to b.

Output

The output contains an integer denoting the maximum number of producers which can be left running such that all the produced products can be delivered to the warehouse and each conveyor belt transports at most one product at any time.

Explanation

In Sample Input 1, $N = 4$, $K = 2$, $M = 3$, and the directed edges are $\{(1, 3), (2, 3), (3, 4)\}$. There is only one possible delivery route for each producer, i.e. $1 \rightarrow 3 \rightarrow 4$ for producer 1, and $2 \rightarrow 3 \rightarrow 4$ for producer 2. Both producers are using conveyor belt $(3, 4)$, however, the products from producer 1 are on the conveyor belt $(3, 4)$ on minutes $2, 4, 6, \ldots$ (even minutes), while the products from producer 2 are on the conveyor belt $(3, 4)$ on minutes $3, 5, 7, \ldots$ (odd minutes). Therefore, both producers can be left running.

In Sample Input 2, $N = 5$, $K = 2$, $M = 4$, and the directed edges are $\{(1, 3), (3, 4), (2, 4), (4, 5)\}$. Similar to the previous example, there is only one possible delivery route for each product produced by each producer. In this example, only one producer can be left running as products from both producers (1 and 2) are on the conveyor belt $(4, 5)$ at the same time if both are running.

Sample 1

Input	Output
4 2 3 1 3 2 3 3 4	2

Sample 2

Input	Output
5 2 4 1 3 3 4 2 4 4 5	1

Sample 3

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

Problem L. Non-Prime Factors

Time limit	1000 ms
Mem limit	1048576 kB
OS	Linux

In many programming competitions, we are asked to find (or count the number of) Prime Factors of an integer i. This is boring. This time, let's count the number of Non-Prime Factors of an integer i, denoted as `NPF(i)`.

For example, integer 100 has the following nine factors: $\{1, \underline{2}, 4, \underline{5}, 10, 20, 25, 50, 100\}$. The two which are underlined are prime factors of 100 and the rest are non-prime factors. Therefore, `NPF(100)` = 7.

Input

The first line contains an integer Q ($1 \leq Q \leq 3 \cdot 10^6$) denoting the number of queries. Each of the next Q lines contains one integer i ($2 \leq i \leq 2 \cdot 10^6$).

Output

For each query i, print the value of `NPF(i)`.

Warning

The I/O files are large. Please use fast I/O methods.

Sample 1

Input	Output
4 100 13 12 2018	7 1 4 2