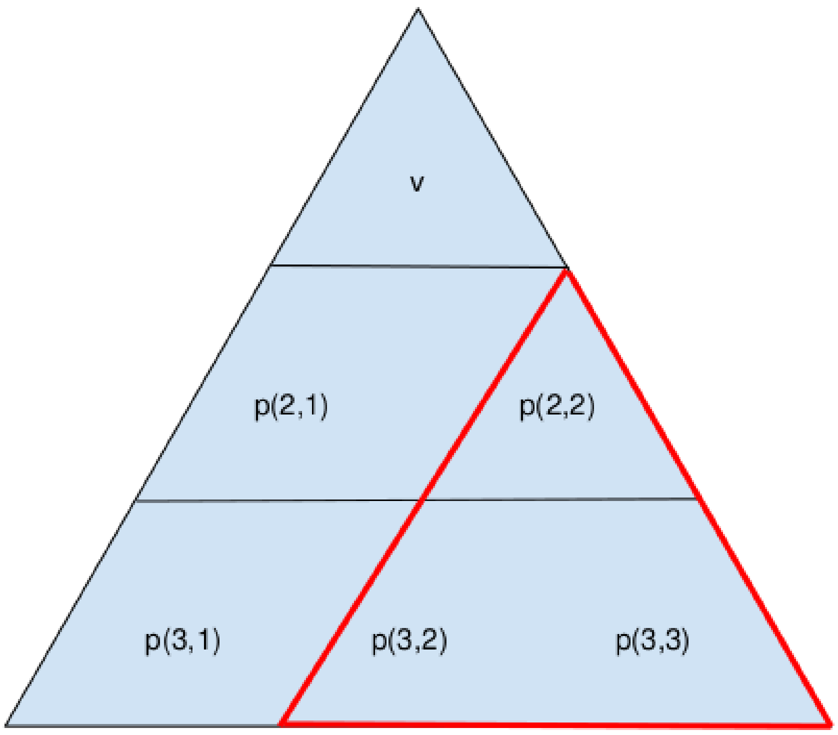
# Problem A. A Pyramidal Task

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 5 seconds |
| Memory limit: | 512 mebibytes |

An *pyramid* is a table of numbers $p(i, j)$ defined for integers $1 \le j \le i \le n$. Values of $p(i, j)$ are defined as follows:

- $p(1, 1) = v$;
- $p(i, 1) = (c \cdot p(i-1, 1)) \bmod m$, for $2 \le i \le n$;
- $p(i, j) = (a \cdot p(i, j-1) + b \cdot p(i-1, j-1)) \bmod m$, for $2 \le i \le n$ and $2 \le j \le i$.

Given an pyramid, you must answer queries about maximum value in requested sub-pyramids. A *sub-pyramid* is defined by coordinates of its top element $(r, s)$ and side length $x$. For example, a sub-pyramid with $r = 2$, $s = 2$ and $x = 2$ on the picture below is highlighted with red.



## Input

The first line of the input contains six integers $n$, $v$, $a$, $b$, $c$ and $m$ ($1 \le n \le 4000$, $1 \le v, a, b, c \le 10^9$, $2 \le m \le 10^9$) — parameters of the pyramid.

The next line contains one integer $q$ ($1 \le q \le 5 \cdot 10^5$) — the number of queries.

Then $q$ lines follow, each containing three integers $r_i$, $s_i$ and $x_i$ ($1 \le r_i \le n$, $1 \le s_i \le r_i$ and $1 \le x_i \le n - r + 1$) — parameters of the sub-pyramid in $i$-th query.

## Output

For each query, print the maximum value in the requested sub-pyramid.

## Examples

| standard input | standard output |
|---|---|
| 3 1 1 1 1 3<br>2<br>1 1 1<br>1 1 2 | 1<br>2 |
| 5 3 1 1 1 5<br>4<br>1 1 1<br>3 3 2<br>1 1 4<br>2 1 1 | 3<br>4<br>4<br>3 |
| 34 72 111 13 17 912131<br>5<br>17 17 9<br>17 9 10<br>11 10 8<br>13 9 1<br>3 3 30 | 877499<br>896373<br>838846<br>429006<br>911914 |

# Problem B. Big Storm

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 512 mebibytes |

A big storm tossed a swarm of bees on the coordinate plane, and each bee is currently at a point with integer coordinates. All the bees want to meet at the same point (also with integer coordinates). In one second, **at most one** bee can move one cell in one of the allowed directions, which are among eight possible options: north, northeast, east, southeast, south, southwest, west, and northwest. North direction corresponds to $0y$ axis direction, and east corresponds to $0x$ axis direction.

Find the minimum number of seconds the bees need to gather at any point with integer coordinates.

## Input

The first line of the input contains one integer $d$ ($1 \le d \le 8$) — the number of allowed directions. The next line contains $d$ space-separated strings that describe these directions. Strings "`N`", "`NE`", "`E`", "`SE`", "`S`", "`SW`", "`W`", and "`NW`" are used for north, northeast, east, southeast, south, southwest, west, and northwest directions, respectively. The third line contains one integer $n$ ($1 \le n \le 50$) — the number of bees in the swarm.

The $i$-th of the next $n$ lines contains two integers $x_i$ and $y_i$ ($-10^6 \le x_i, y_i \le 10^6$) — coordinates of the initial position of the $i$-th bee. You may assume that no two bees are initially located at the same point.

## Output

Print one integer — the minimum number of seconds bees need to meet all together at some point with integer coordinates. You may assume that solution exists for all input data.

## Example

| standard input | standard output |
|---|---|
| 3<br>N NE NW<br>5<br>1 1<br>5 1<br>1 4<br>3 3<br>2 3 | 13 |
| 3<br>N SE SW<br>3<br>-4 1<br>-4 -2<br>4 -2 | 16 |
| 6<br>N E W SW NE NW<br>3<br>249 218<br>443 -290<br>231 -444 | 1013 |
| 3<br>S SE NE<br>4<br>437 -368<br>387 -47<br>-422 136<br>-407 -300 | 2025 |

# Problem C. Cheese!

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

John has a flat piece of cheese in the form of an $h \times w$ rectangle. He calls a straight cut along some segment with endpoints on opposite vertical sides (sides of length $h$) an *A-type cut*, and a cut along some segment with endpoints on opposite horizontal sides (sides of length $w$) a *B-type cut*.

There are $n$ small holes strictly inside the piece, each of them can be treated as a point. While dividing the cheese John makes $a$ cuts of type A and $b$ cuts of type B. John makes sure that no two segments corresponding to cuts of the same type share a common point, and that no cut goes through any hole.

Given information about all the holes and cuts, print the maximum number of holes among resulting cheese slices (that is, connected regions after all the cuts are made). You may treat the holes as points, and assume that holes are strictly inside the resulting slices.

## Input

The first line of the input contains two integers $w$ and $h$ ($1 \le w \le 10^7$, $1 \le h \le 10^7$) — the width and the height of the cheese piece respectively. The next line contains one integer $n$ ($1 \le n \le 10^5$) — the number of holes in the piece. The $i$-th of the following $n$ lines contains two integers $x_i$, $y_i$ ($0 < x_i < w$, $0 < y_i < h$) — coordinates of the $i$-th hole. You may assume that no two holes coincide.

The next line contains one integer $a$ ($1 \le a \le 10^5$) — the number of A-type cuts. The $i$-th of the next $a$ lines contains two integers $y_l$ and $y_r$ — $y$-coordinates of left and right endpoints of the $i$-th A-cut ($0 \le y_l, y_r \le h$).

The next line contains one integer $b$ ($1 \le b \le 10^5$) — the number of B-type cuts. The $i$-th of next $b$ lines contains two integers $x_u$ and $x_d$ — $x$-coordinates of upper and lower endpoints of the $i$-th B-cut ($0 \le x_u, x_d \le w$).

You may assume that no cut goes through any hole, and that no two cuts of the same type have a common point (neither intersect nor touch).

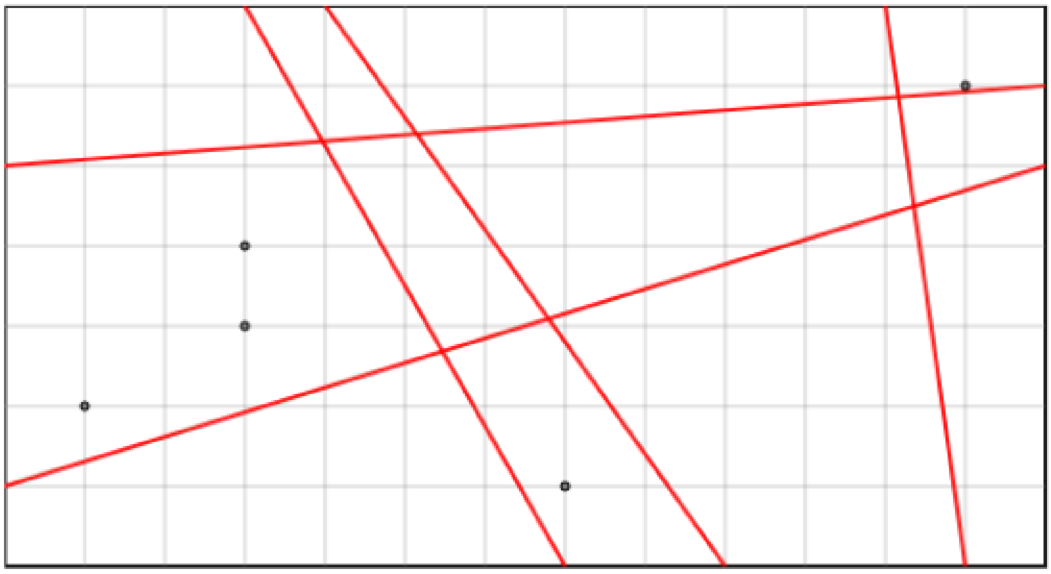
## Output

Print one integer — maximum number of holes in one connected slice.

## Example

| standard input | standard output |
|---|---|
| 13 7<br>5<br>1 2<br>3 3<br>7 1<br>3 4<br>12 6<br>2<br>1 5<br>5 6<br>3<br>3 7<br>4 9<br>11 12 | 3 |
| 15 10<br>8<br>6 2<br>2 4<br>5 3<br>8 2<br>9 3<br>6 3<br>5 6<br>5 4<br>3<br>2 6<br>1 2<br>9 8<br>3<br>3 2<br>8 11<br>14 13 | 5 |

## Note



The picture above illustrates the first sample.

# Problem D. Direct The Streets

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Councillors from the Jumbo-City are overwhelmed with complaints about traffic-jams in the city centre. Most people refuse to walk and prefer local means of transportation — the elephants. The problem is that downtown lanes are quite narrow and local elephants are rather large. When two elephants come across in the opposite directions they usually stuck in the lane and jam the traffic for a very long time.

To prevent these collisions, the councillors have decided to change all lanes into one-way streets. However, the department of traffic and transportation couldn't comply, because there are not enough traffic signs for such an operation. Situation is getting critical so the councillors have asked for your help. You'll get a map of the town described as an undirected planar graph and you're supposed to direct all edges. The department of traffic and transportation warned you that from each vertex could lead at most 3 outgoing edges (number of incoming edges is not limited). You are not supposed to take care about any other details (such as vertex reachability or strong connectivity). If there are more correct solutions for a given graph you may present any one of them.

## Input

The first line of the input contains two integers $N$ and $M$ ($1 \le N \le 2 \cdot 10^5$, $1 \le M \le 10^6$) where $N$ represents number of vertices and $M$ — number of edges. The following $M$ lines describe edges. Each edge is defined by numbers of two incident vertices $i$ and $j$ ($1 \le i, j \le N$). The graph is guaranteed to be planar, i.e., it is possible to draw it in the plane (vertices are points, edges are line segments) in such a way that no two edges have a point in common except possibly for their end vertices. There are no loops nor multiple edges in the graph.

## Output

Results are written in the very same format as the input is. Just do not forget that this time the edges are oriented (edge that is defined by vertices $i$ and $j$ is oriented from $i$ to $j$). If the given graph has no solution at all you are supposed to write "`no`" string into the output file.

## Example

| standard input | standard output |
|---|---|
| 7 11<br>1 4<br>1 6<br>2 3<br>3 4<br>5 1<br>5 2<br>5 3<br>5 4<br>5 6<br>5 7<br>6 7 | 7 11<br>1 6<br>1 4<br>2 3<br>3 4<br>4 5<br>5 3<br>5 2<br>5 1<br>6 7<br>6 5<br>7 5 |

# Problem E. Enormous Numbers

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 4 seconds |
| Memory limit: | 512 mebibytes |

The plane is divided into $1 \times 1$ axes-aligned squares with integer coordinates of all vertices. A square with bottom left corner at the point $(p, q)$ contains the number $p! \cdot q!$ (in this problem we will only consider non-negative coordinates).

We have a closed polyline such that:

- all segments of the polyline are parallel to horizontal or vertical axis;
- the polyline passes through each point at most once;
- all vertices of the polyline are points with integer non-negative coordinates;
- no three consecutive vertices of the polyline lie on the same straight line.

You have to calculate the sum of numbers inside this polyline modulo $10^9 + 7$.

## Input

The first line of the input contains one integer $n$ ($4 \le n \le 100$) — the number of vertices of the polyline.

Each of the next $n$ lines contains two integers between 0 and $10^9$, inclusively — coordinates of vertices of the polyline in order of traversing the polyline.

## Output

Print one integer — the answer to the problem.

## Example

| standard input | standard output |
|---|---|
| 4 | 1 |
| 1 1 | |
| 1 2 | |
| 2 2 | |
| 2 1 | |
| 4 | 9 |
| 3 3 | |
| 1 3 | |
| 1 1 | |
| 3 1 | |
| 18 | 119 |
| 0 0 | |
| 2 0 | |
| 2 1 | |
| 3 1 | |
| 3 2 | |
| 5 2 | |
| 5 3 | |
| 4 3 | |
| 4 4 | |
| 3 4 | |
| 3 3 | |
| 2 3 | |
| 2 4 | |
| 0 4 | |
| 0 3 | |
| 1 3 | |
| 1 1 | |
| 0 1 | |

# Problem F. Fill The Boxes

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

You want to play a computer game "Fill The Boxes". There are $n$ initially empty boxes at the bottom of the screen. At the top of the screen letters appear one at a time. Each letter is chosen from the set $\{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$ randomly, using the given probability distribution. Each letter is chosen independently from the previous letters.

As soon as the next letter appears, the player should place it in one of the empty boxes. The game ends when all boxes are filled. The player wins if the letters are arranged lexicographically in non-descending order.

Given the probability for each letter to appear on each step, calculate the probability for the player to win when playing optimally.

## Input

The first line contains an integer $n$ ($1 \le n \le 250$) — the number of empty boxes at the bottom and the number of letters appearing at the top of the screen one by one. The second line contains four integers $p_a$, $p_b$, $p_c$ and $p_d$ — appearance probabilities of letters 'A', 'B', 'C', and 'D' respectively, in percents ($0 \le p_a, p_b, p_c, p_d \le 100$, $p_a + p_b + p_c + p_d = 100$).

## Output

Print the probability to win when playing optimally with absolute or relative error $10^{-6}$ or better.

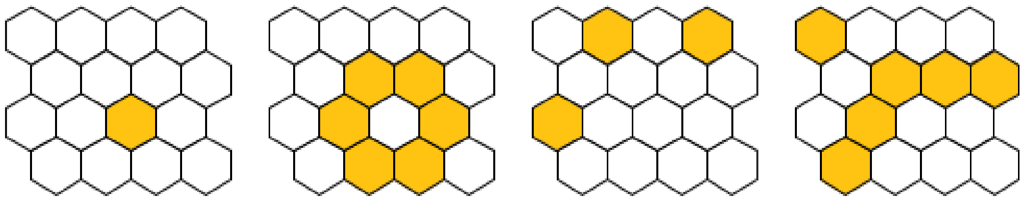
## Example

| standard input | standard output |
|---|---|
| 3<br>25 25 25 25 | 0.75 |
| 10<br>0 50 0 50 | 1 |
| 5<br>10 20 30 40 | 0.62116 |

## Problem G. Grid With Honey

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 1.5 seconds |
| Memory limit: | 512 mebibytes |

Every day bees store honey in a honeycomb, which can be represented as a regular hexagonal grid, arranged in rows and columns. During the night, larvae eat all the honey from the comb so that at the beginning of each day all cells are empty.



It's a little known fact that bees select the cells to store honey depending on which cells it was stored in at the end of the previous day. Namely, a cell will contain honey if and only if the number of neighboring cells (not including the current cell itself) containing honey on the previous day was odd. Otherwise, the cell will be empty.

Given the initial distribution of honey, calculate the placement of honey in honeycomb after $k$ days.

### Input

The first line of the input contains three integers $n$, $m$ and $k$ ($2 \le n, m \le 10$, $1 \le k \le 2^{63} - 1$) — the number of rows and columns in the honeycomb, and the number of days, respectively.

Each of the next $n$ lines contains $m$ characters. Corresponding character is equal to '`.`' if the cell is empty on the first day, or '`*`' otherwise.

The honeycomb is given in such a way that first hexagon of the first line is placed above and to the left from the first hexagon of the second line (see the picture for clarity).

### Output

Print $n$ lines containing $m$ characters each — the honeycomb state after $k$ days in the same format as in the input.

### Example

| standard input | standard output |
|---|---|
| `4 4 3`<br>`....`<br>`....`<br>`..*.`<br>`....` | `*...`<br>`.***`<br>`.*..`<br>`*...` |
| `3 10 8`<br>`*.**..*...`<br>`*.**.*....`<br>`...*..*...` | `*..*...**.`<br>`.*.*...***`<br>`*..**.**..` |

# Problem H. How To Cheat In Lottery

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1.5 seconds |
| Memory limit: | 512 mebibytes |

Petya is addicted to the lottery called The Jackpot Sequence.

The lottery consists of two phases:

1. At the beginning of the day Petya buys a ticket on which a sequence of numbers is printed.
2. At the end of the day the winning sequence is drawn from the lottery machine.

If the drawn sequence coincides with the one printed at the ticket, Petya wins the jackpot.

The lottery machine can only contain *valid* sequences. As mentioned in the rules of the lottery, a sequence is *valid* if and only if:

- it consists of integers between 1 and $n$, inclusive;
- all integers in the sequence are pairwise distinct;
- each pair of neighboring numbers in sequence belongs to the whitelist.

The whitelist can be represented as a list of unordered pairs of integers from the range $[1, n]$.

Petya's friend Vasya managed to get the whitelist, and after some studies the friends noticed that it is impossible to choose three distinct valid sequences so that they are *triple linked.*

Three sequences are *triple linked* if:

- The first number of all three sequences is the same (denote it as $a$).
- The last number of all three sequences is the same (denote it as $b$).
- There is no other number (except for $a$ and $b$) that is common to more than one of these sequences.

Petya is wondering about the distribution of valid sequences, so he asks you to calculate the number of valid sequences of length $l$ for each $l$ between 1 and $n$, inclusive. Since the answers may be too big, print them modulo $10^9 + 7$.

## Input

The first line contains two integers $n$ and $p$ ($1 \le n \le 4000$, $0 \le p \le 10^5$) — the maximum possible integer in a valid sequence, and the length of the whitelist. Each of next $p$ lines contains a pair from the whitelist — two distinct integers between 1 and $n$, inclusive. You may assume that each unordered pair is listed in the whitelist at most once.

## Output

Print $n$ integers. The $i$-th of these integers should be equal to the number of valid sequences of length $i$ modulo $10^9 + 7$.

## Example

| standard input | standard output |
|---|---|
| 3 2<br>1 2<br>2 3 | 3 4 2 |
| 3 3<br>1 3<br>2 3<br>1 2 | 3 6 6 |

# Problem I. Inside The Matrix

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 512 mebibytes |

Consider a square matrix of size $n \times n$. Integers from 1 to $n^2$ are placed in this matrix in a spiral order. The process starts at the left up corner, with a spiral going down, then turn right, then up, left and the process repeats until there is at least one empty cell left.

Here is the example of already filled $4 \times 4$ matrix:

| | | | |
|---|---|---|---|
| 1 | 12 | 11 | 10 |
| 2 | 13 | 16 | 9 |
| 3 | 14 | 15 | 8 |
| 4 | 5 | 6 | 7 |

Find the sum of integers in a rectangle submatrix formed by cells in rows from $r_1$ to $r_2$ and columns from $c_1$ to $c_2$.

## Input

The first line of the input contains one integer $n$ ($1 \le n \le 10^9$). The second line contains four integers $r_1$, $r_2$, $c_1$, $c_2$ ($1 \le r_1 \le r_2 \le n$, $1 \le c_1 \le c_2 \le n$).

## Output

Print one integer — the sum of all elements in the given submatrix modulo $10^9 + 2012$.

## Examples

| standard input | standard output |
|---|---|
| 4<br>2 3 1 3 | 63 |

# Problem J. Join Them With OR

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 512 mebibytes |

You are given a sequence $a_1, \ldots, a_n$ of $n$ non-negative integers.

For every integer $k$ between 1 and $n$, find the greatest integer $x_k$ that can be obtained as a result of bitwise OR of $k$ consecutive elements of the sequence $a_1, \ldots, a_n$.

Note, that in the bitwise OR of a set of integers the $i$-th bit of the result is equal to 0 if and only if the $i$-th bit in all integers from the set is equal to 0. Otherwise, the $i$-th bit is equal to 1.

## Input

The first line of the input contains integer $n$ ($1 \le n \le 5 \cdot 10^5$) — the number of elements in the sequence. The $i$-th of the next $n$ lines contains one integer $a_i$ ($0 \le a_i \le 2^{30}$).

## Output

Print $n$ lines. The $k$-th of those lines should contain the largest integer that can be represented as a result of bitwise OR of $k$ consecutive elements of sequence $a_1, \ldots, a_n$.

## Example

| standard input | standard output |
|---|---|
| 8<br>10<br>7<br>11<br>4<br>7<br>6<br>2<br>16 | 16<br>18<br>22<br>23<br>23<br>31<br>31<br>31 |

# Problem K. Key Validation

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 2 seconds |
| Memory limit: | 512 mebibytes |

In this problem a *key number* is a sequence consisting of exactly sixteen decimal digits. Such sequences are often used for bank cards, personal ID's, some discount cards etc. However, to avoid accidental typos not every sixteen-digit sequence is considered to be a valid key number. To validate the key number, the *Luhn formula* is used.

The formula verifies a number against its included check digit. In this particular problem the last digit is always the check digit. To be considered valid, the key number must pass the following test:

- from the rightmost digit, which is the check digit, and moving left, double the value of every second digit. That is, if the check digit has index 16, you should double digits at positions 15, 13, etc. If the result of this doubling operation is greater than 9 (e.g., $8 \cdot 2 = 16$), then sum the digits of the product (e.g., 16: $1 + 6 = 7$, 18: $1 + 8 = 9$) or alternatively (the result is always the same, just another way to treat the process) subtract 9 from the product (e.g., 16: $16 - 9 = 7$, 18: $18 - 9 = 9$);
- take the sum of all the digits;
- if the total modulo 10 is equal to 0 (if the total ends in zero) then the number is valid, otherwise it is invalid.

Given a key number, you need to check if this number is valid.

## Input

Inpit contains a key number — one string consisting of exactly 16 decimal digits.

## Output

If the given key number is valid, print "`YES`". Otherwise print "`NO`".

## Example

| standard input | standard output |
|---|---|
| 1111222233334444 | YES |
| 7827889687678773 | YES |
| 0987654321098765 | NO |

# Problem L. Letter Manipulations

| | |
|---|---|
| Input file: | *standard input* |
| Output file: | *standard output* |
| Time limit: | 5 seconds |
| Memory limit: | 512 mebibytes |

You are given a string $s$ of lowercase English letters, and an integer $k$. You can choose at most $k$ positions and replace them with some other characters. The goal is to maximize the length of the longest substring that appears in the new string at least twice.

## Input

The first line of the input consists of one integer $k$ ($0 \le k \le 5000$). The second line contains non-empty string $s$ consisting of at most 5000 lowercase English letters.

## Output

Print one integer — maximum possible length of the substring, appearing in the modified string at least twice. If there is no solution, print 0 instead.

## Example

| standard input | standard output |
|---|---|
| 1<br>abcba | 3 |
| 7<br>aaaaaaaaaa | 9 |