Chat System (chat)

In his never-ending quest for improvements of the Italian training website, William decided to rewrite the chat system from scratch. This system allows students to communicate with one another in real time.

A key component in any chat system is the "recent chat list". This list shows all the names of people who have been contacted at least once, sorted by last contact time, that is: in the top there will be the user that has been most recently contacted (no other user has been contacted yet), in the bottom there is the user that has been least recently contacted (no other user was last contacted before). Of course, this final list cannot have duplicates by construction.

Help William write the "recent chat list" component. You will be given the list of people contacted, from the first person contacted to the last. There can be multiple entries in this list (when a person is contacted multiple times). Print the order in which those people will appear in the final list.

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

Input

The first line contains the only integer N, the number of people that have been contacted. The following N lines contain the usernames of the people contacted (from the first person contacted to the last).

Output

You need to print the "recent chat list" as described before.

Constraints

- $1 \leq N \leq 100\,000$.
- Usernames consist of between 1 and 10 lowercase alphabetical 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** [**5 points**]: Examples.
- **Subtask 2** [**25 points**]: $N \leq 10$.
- **Subtask 3** [**40 points**]: $N \leq 1000$.
- **Subtask 4** [**30 points**]: No additional limitations.

Examples

input.txt	output.txt
6 william luca goofy luca william luca	luca william goofy
4 luca luca william william	william luca

Explanation

In the **first sample case** the recent chat list changes among these states:

- william
- luca, william
- goofy, luca, william
- luca, goofy, william
- william, luca, goofy
- luca, william, goofy

In the **second sample case** the recent chat list changes among these states:

- luca
- luca
- william, luca
- william, luca

Connect X (connectx)

Giorgio and William are spending some time together brainstorming to come up with impossible tasks for the forthcoming round of the *IOI-team*. While pondering about the subtleties of their algorithmic puzzles, they enjoy releasing tension by playing *Connect X*, a generalization of the well-known *Connect Four*™ game (where 4 is replaced by a possibly different number X). More precisely:

- Giorgio and William play in an $H \times W$ grid, held vertically on a table;
- The game proceeds in alternating turns by inserting tokens into non-full columns;
- When a token is inserted into a column it falls through empty cells, eventually stopping on the lowest empty cell in that column;
- The game ends as soon as a straight (vertical, horizontal or diagonal) line of X tokens belonging to the same player is completed, awarding victory to that player;
- If no line of X tokens of a same player is completed before completely filling the grid, the game is declared a draw.

Figure 1: A common Connect Four board.

Since Giorgio and William are not very enthusiastic about this game (and would like to avoid petty quarrels which inevitably arise from victories and losses), they will both play to draw (i.e. neither of them wants to win).

Help them concentrate on preparing better tasks for the next round by computing a **drawing strategy** for *Connect X*!

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

Input

The first and only line contains the three integers X, H, W.

Output

You need to write a single line with $H \times W$ integers: a sequence of moves (column indexes from 0 to $W - 1$) which fills the grid without creating lines of length X.

Constraints

- $3 \leq X \leq 100$.
- $1 \leq H, W \leq 1000$.

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.

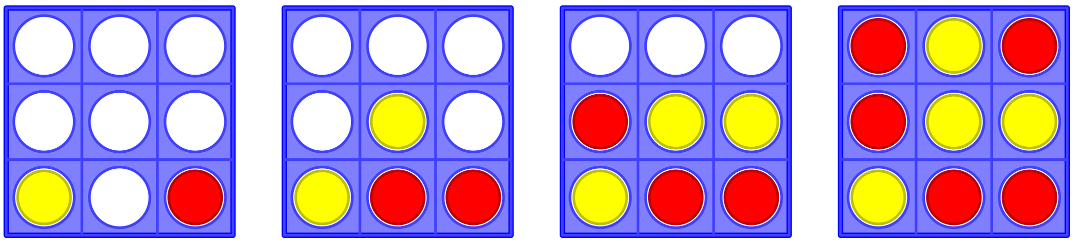
- **Subtask 1** [**5 points**]: Examples.
- **Subtask 2** [**10 points**]: $H, W \leq 4$.
- **Subtask 3** [**15 points**]: $H < X$.
- **Subtask 4** [**20 points**]: $X \geq 30$.
- **Subtask 5** [**30 points**]: W is even.
- **Subtask 6** [**20 points**]: No additional limitations.

Examples

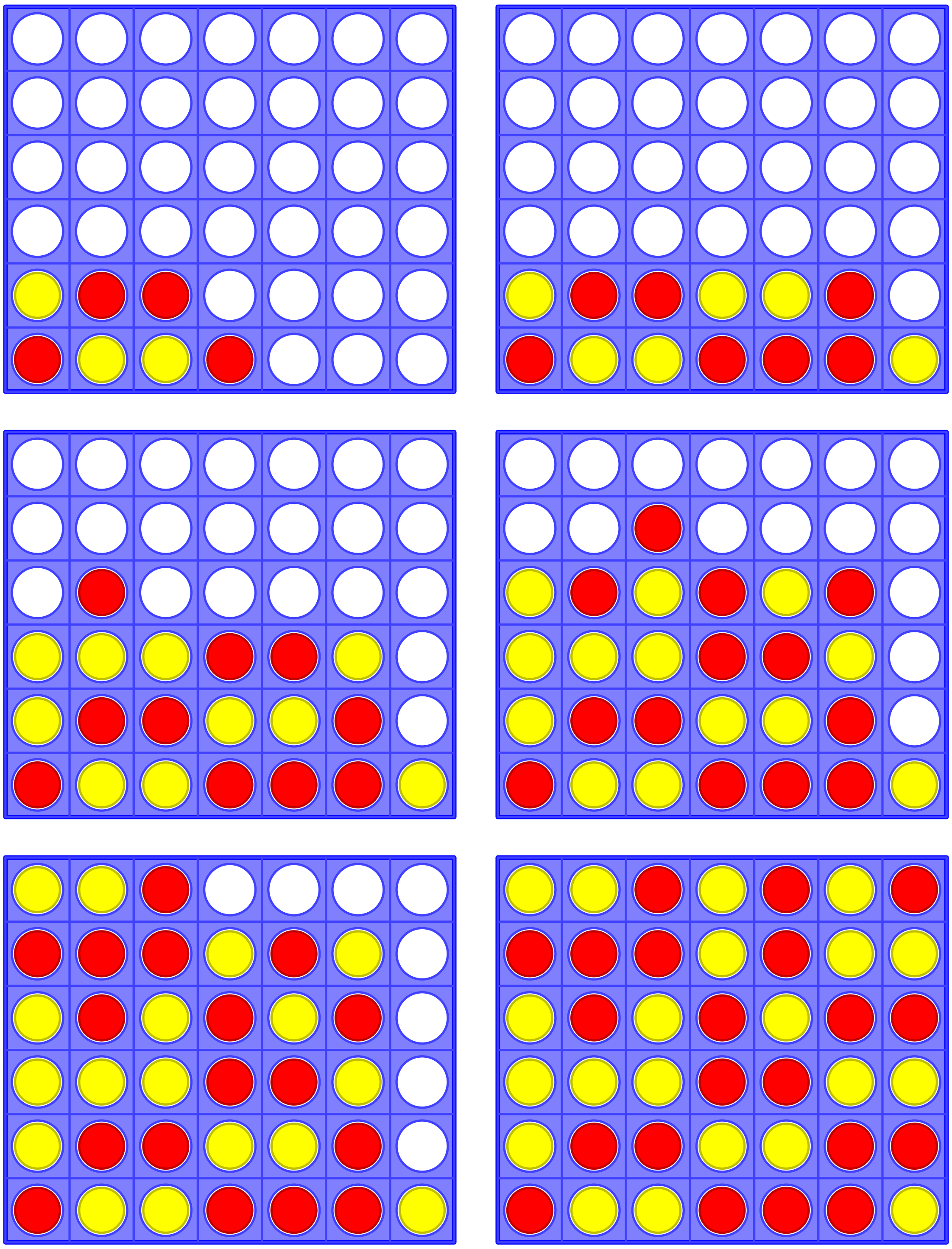
input.txt	output.txt
`3 3 3`	`2 0 1 1 0 2 0 1 2`
`5 6 7`	`0 1 3 2 1 0 2 3 4 6 5 4 5 1 1 2 3 0 4 5 5 4 3 2 2 0 1 1 0 0 2 3 4 5 6 6 6 6 6 5 4 3`

Explanation

In the **first sample case**, the grid is filled in the following way:



In the **second sample case**, the grid is first filled as in Figure 1 and then completed. Altogether, it is filled in the following way:



Perfect Crossover (`crossover`)

This month William got interested in genetic algorithms, that he is now using for solving even the simplest everyday tasks. However, he is unsatisfied with the performance of the *crossover* phase. In this phase, two strings `s` and `t` formed by features of selected individuals are combined together into a new string `s&t`, in order to mix them up exactly a bit.

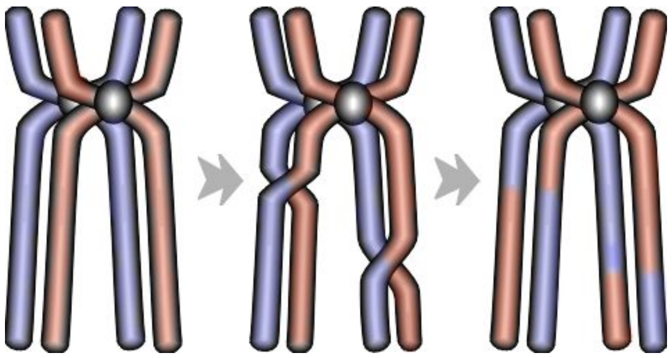


Figure 1: A representation of crossover on animal chromosomes.

After days of excruciating research, William finally came up with few requirements that (in his opinion) would characterize a perfect crossover (where `s`, `t` stand for strings and `a`, `b`, `c` stand for single characters):

1. `s & "" = "" & s = s`, i.e., a crossover with an empty string does not change the other string;
2. `(cs) & (tc) = t&s`, i.e., a crossover among strings such that the first character of the first string equals the last character of the last string results in that equal characters being removed and the remaining strings swapped;
3. `(as) & (tb) = b(s&t)a`, i.e., when the first and last characters are not equal the crossover results in swapping those characters, interleaved by the crossover of the remaining strings (unaltered).

Notice that this operation *is not commutative*. For example, if we crossover `adamo` and `emma` in the two possible ways we obtain:

```
  adamo & emma          emma & adamo
       ↓                      ↓
   emm & damo           o(mma & adam)e
       ↓                      ↓
  o(mm & dam)e          o(ada & ma)e
       ↓                      ↓
   o(da & m)e            o(m & da)e
       ↓                      ↓
 o(m(a & "")d)e        o(a("" & d)m)e
       ↓                      ↓
     omade                  oadme
```

Help William's research by implementing the crossover operation!

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

Input

The first line contains the first string `s`. The second line contains the second string `t`.

Output

You need to write a single line with a string: the crossover `s&t`.

Constraints

- `s`, `t` are between 1 and 100 000 characters long.
- `s`, `t` consist entirely of lowercase letters `'a'`–`'z'`.

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** [**5 points**]: Examples.
- **Subtask 2** [**10 points**]: Rule (3) never applies[1].
- **Subtask 3** [**10 points**]: Rule (2) never applies[2].
- **Subtask 4** [**15 points**]: `s`, `t` are at most 10 characters long.
- **Subtask 5** [**30 points**]: `s` is at most 10 characters long.
- **Subtask 6** [**20 points**]: `s`, `t` share the same length.
- **Subtask 7** [**10 points**]: No additional limitations.

Examples

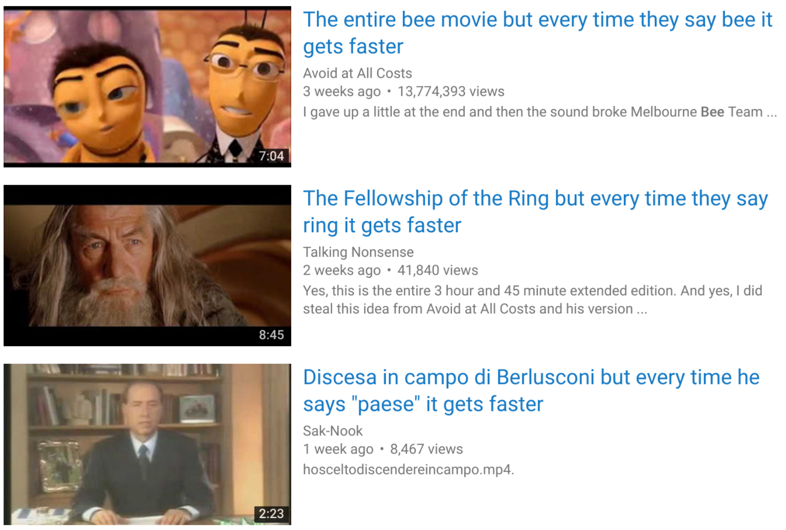
`input.txt`	`output.txt`
`adamo` `emma`	`omade`
`emma` `adamo`	`oadme`

[1]That is, the relevant characters are always compared equal.
[2]That is, the relevant characters are always compared different.

But It Keeps Getting Faster (`faster`)

Everybody knows that William is constantly looking for the rarest and dankest memes on the magical place that is the Internet. Today he found a new one: the "but it keeps getting faster" meme. This meme consists of a video which is initially played at normal speed, but then just keeps getting faster.

In most cases, the speedup is triggered by a specific event. For example, everytime someone says a particular word (typically a word that is used *a lot* throughout the video) like in the following cases:


The entire bee movie but every time they say bee it gets faster
Avoid at All Costs
3 weeks ago • 13,774,393 views
I gave up a little at the end and then the sound broke Melbourne **Bee** Team ...
7:04

The Fellowship of the Ring but every time they say ring it gets faster
Talking Nonsense
2 weeks ago • 41,840 views
Yes, this is the entire 3 hour and 45 minute extended edition. And yes, I did steal this idea from Avoid at All Costs and his version ...
8:45

Discesa in campo di Berlusconi but every time he says "paese" it gets faster
Sak-Nook
1 week ago • 8,467 views
hosceltodiscendereincampo.mp4.
2:23

Figure 1: Some "but it keeps getting faster" videos.

After watching all these memes, William decided to create one himself. However, to make a meme as dank as possible, he needs to know in advance how long it will be. He will start with an existing video which is N seconds long. We can assume that all the words in the video are spoken at a constant speed of 1 word per second. William has the entire transcript of the video, composed of N words.

To edit the video, William will increase the speed by 1 word per second everytime a special word W is spoken. However, William's editor has some limitations.

- *A new speed can be set only at the precise start of a new second*: for this reason, every time a word is spoken the speed-up is postponed to the next integral second, where we will get an increase of k words per second if in the last second the special word has been spoken k times.
- *The length of every video need to be an integer amount of seconds*: for this reason, William will add some blank frames at the end of the video rounding up the length.

Help him calculate how long (in seconds) his video will be!

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

Input

The first line contains the special "trigger word" W. The second line contains the number N of words in the video. The following N lines contain the transcript, that is, all the words spoken in the video.

Output

You need to write a single line with an integer: the duration (in seconds) of the edited video.

Constraints

- $1 \leq N \leq 100\,000$.
- Every word consists of between 1 and 10 lower case alphabetical letters.

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** [**5 points**]: Examples.
- **Subtask 2** [**10 points**]: The trigger word never appears.
- **Subtask 3** [**30 points**]: It's all the same word: the trigger word.
- **Subtask 4** [**55 points**]: No additional limitations.

Examples

`input.txt`	`output.txt`

`hello`
`8`
`hello`
`its`
`me`
`hello`
`can`
`you`
`hear`
`me`
	`4`

`hello`
`7`
`hello`
`hello`
`hello`
`hello`
`hello`
`hello`
`hello`
	`3`

input.txt	output.txt

```
bee
48
according
to
all
known
laws
of
aviation
there
is
no
way
a
bee
should
be
able
to
fly
its
wings
are
too
small
to
get
its
fat
little
body
off
the
ground
the
bee
of
course
flies
anyway
because
a
bee
doesnt
care
what
humans
think
is
impossible
```
	`28`

Explanation

In the **first sample case** the words spoken in each second are:

(hello), (its me), (hello can), (you hear me)

In the **second sample case** the words spoken in each second are:

(hello), (hello hello), (hello hello hello hello)

Enrichment Center (glados)

After playing *Portal*, Giorgio decided to open his own *Enrichment Center* and be like GLaDOS in real life. In this kind of research facility, test subjects are locked into chambers and faced with several dangerous puzzles, which need to be solved to gain access to the exit (and to another even riskier chamber). Obviously, there are annoying laws about human rights that would interfere with a faithful implementation of this concept; but Giorgio already planned to get around that by building a very small (and cheap!) Enrichment Center designed for lab rats.

Of course, lab rats are not exceptionally smart so the puzzles need to be fairly straightforward. In the first test chamber, Giorgio is planning to arrange into a grid $H \times W$ some blocks of the following types:

- plain floor, represented by ‘`.`’;
- walls, represented by ‘`#`’;
- exits, represented by ‘`O`’;
- transparent glue, represented by ‘`@`’.

Figure 1: A prototypical chamber in the *Aperture Science*$^{\text{TM}}$ Enrichment Center.

Giorgio put the first lab rat in row-column coordinates $(R; C)$ into the grid, facing north. Quite surprisingly, he noticed that the rat is using one of the most ancient techniques for getting out of a maze: *stick to your right hand.*

In other words, the rat is following the wall that was on his right at the beginning, as if always sticking one hand on it.

Giorgio can't wait to know whether the rat will end up stuck in the glue, or if it will keep cycling forever or eventually reach an exit. Satisfy Giorgio's curiosity by calculating the rat's fate!

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

Input

The first line contains the four integers H, W, R, C. Other H lines follow, each containing a string consisting of W characters among ‘`.`’, ‘`#`’, ‘`O`’, ‘`@`’.

Output

You need to write a single line with a single word:

- `stuck` if the rat will first reach a block with the glue,
- `free` if the rat will first reach a block with an exit,
- `cycling` if the rat will keep cycling forever.

Constraints

- $3 \leq H, W \leq 100$.
- $(R; C)$ vary from $(0; 0)$ — top left to $(H-1, W-1)$ — bottom right.
- All blocks on the perimeter are walls (so the rat cannot start on a border block).
- There is a wall on the right of the rat at the beginning (i.e., in position $(R, C+1)$).

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** [**5 points**]: Examples.
- **Subtask 2** [**45 points**]: The answer is not `cycling`.
- **Subtask 3** [**50 points**]: No additional limitations.

Examples

input.txt	**output.txt**

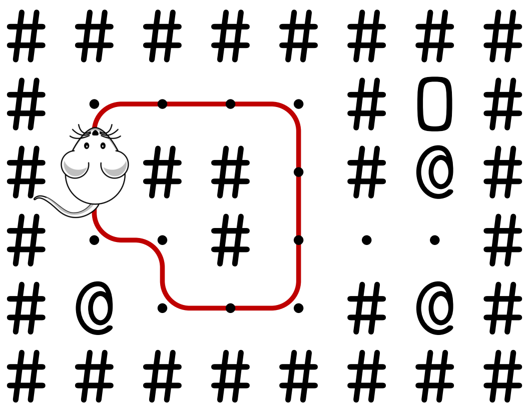
`6 8 2 1`
`########`
`#....#O#`
`#.##.#@#`
`#..#...#`
`#@...#@#`
`########`
	`cycling`

`7 5 5 3`
`#####`
`#O#.#`
`#.#.#`
`#...#`
`#.#.#`
`#.@.#`
`#####`
	`free`

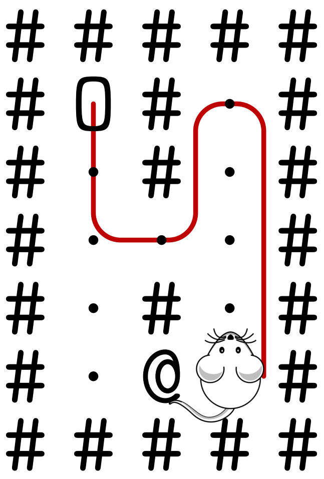
`6 8 2 1`
`########`
`#..@.#O#`
`#.##.#@#`
`#..#...#`
`#@...#@#`
`########`
	`stuck`

Explanation

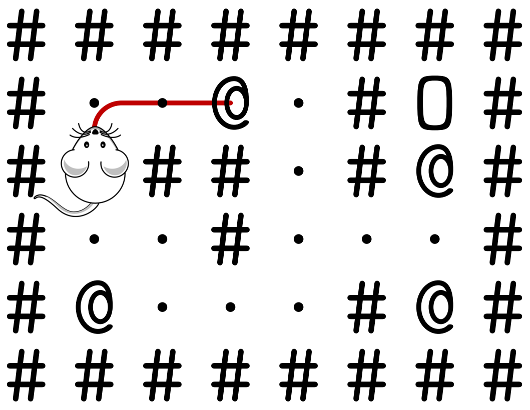
In the **first sample case**, the path followed by the rat is the following:



In the **second sample case**, the path followed by the rat is the following:



In the **third sample case**, the path followed by the rat is the following:



Mosaic Composition (`mosaic`)

Giorgio has recently started to build mosaics out of mosaics. A “mosaic of mosaics” is a mosaic that, instead of using small pieces of stone or glass, uses *actual mosaics* as “tiles”. For simplicity, Giorgio will only use black-and-white square mosaics as tiles for his mosaic. In other words, each tile is a $A \times A$ binary matrix.

Giorgio has N of such tiles available, and he doesn’t need to use all of them. He just needs to arrange some of those pieces in some way as to form a $B \times B$ matrix of tiles (that is: a matrix of matrices).

Two tiles can be adjacent in the final mosaic only if the colors of the edges match (pixel by pixel). **Remember:** tiles cannot be rotated nor reflected!

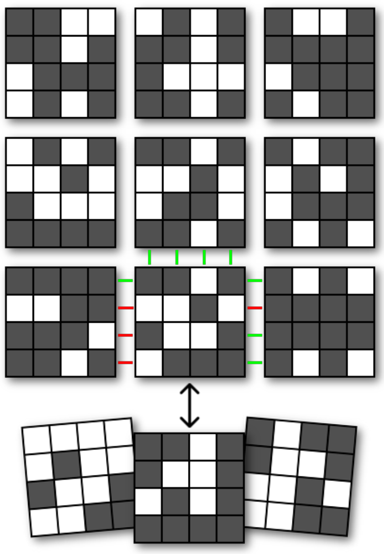


Figure 1: An almost completed mosaic with 3 unused tiles. One of the tiles can be replaced with one from the unused ones in order to complete the mosaic, as shown in the picture.

Help Giorgio find an arrangement of mosaics that correctly forms a mosaic of mosaics.

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

Input

The first line contains two space-separated integers A and B, respectively: the size of the mosaic tiles and the size of the “mosaic of mosaics”. The second line contains a single integer N, the number of tiles available. N matrices follow, each one represented by A lines (each line A characters long), representing the N available tiles.

Output

You need to print a $B \times B$ matrix of integers: the indices (starting from 0) of the tiles used for each position in the "mosaic of mosaics". If there are multiple valid answers, print any one of them.

Constraints

- $2 \leq A \leq 10$.
- $2 \leq B \leq 5$.
- $4 \leq N \leq 50$.
- Each row of a tile is written as a sequence of characters '0' and '1'.
- It's guaranteed that at least 1 answer exists.

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.

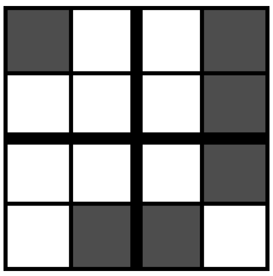
- **Subtask 1** [**5 points**]: Examples.
- **Subtask 2** [**25 points**]: $B = 2$.
- **Subtask 3** [**40 points**]: $N \leq 10$.
- **Subtask 4** [**30 points**]: No additional limitations.

Examples

input.txt	output.txt

2 2
4
01
10
00
01
10
00
01
01
	2 3
	1 0

For this **first sample case** this is the final "mosaic of mosaics":



input.txt	output.txt

input.txt

```
4 3
12
1001
1111
0111
1011
1101
1001
0101
1111
0101
0010
1000
1111
1101
0010
1001
0111
1011
1001
0010
0011
1011
0101
0111
1010
1101
0010
0110
1101
1100
1101
0111
0101
1010
1111
1111
1010
1111
0011
1110
1101
0101
1101
1000
1101
0000
0100
1001
0011
```

output.txt

```
7 10 0
2 6 5
9 1 8
```

For this **second sample case** the final "mosaic of mosaics" is shown in Figure 1.

Server Provisioning (server)

William is in charge of server provisioning for the Italian Team Olympiads, which essentially means that he has to make sure there are enough resources to sustain the load of users that connect during a contest. Today there is a contest (starting at second 0 and ending at second T) and it's important to spend as little as possible, so William wants to increase the server's power *only when needed.*

We will assume for simplicity that:

- When a user connects to the server, they will *stay connected until the end* of the contest.
- If we pay k euro cent per second, then the server can sustain a load of at most k users per second.
- At any integer second we can decide to increase the power (and the payment will increase from k euro cent per second to some other amount, greater than k).

Since William will also have to supervise the contest, he will ask Giorgio to upgrade the server everytime an upgrade is needed.

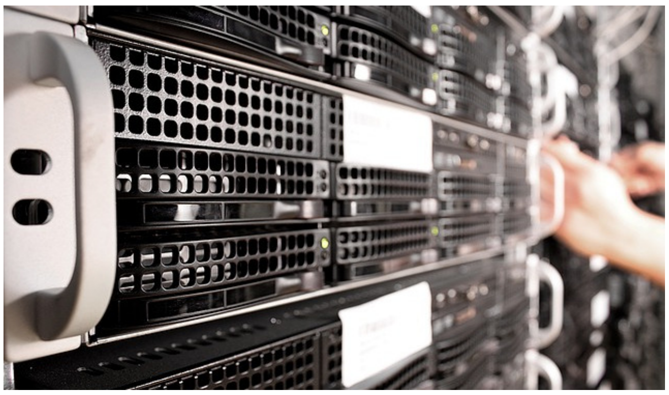


Figure 1: In this picture you can almost see Giorgio, while he is performing a server upgrade.

Upgrading the server is a very difficult and tiring operation. Giorgio will be able to upgrade the server K times **at most**. Given the log of all the N user connections that will happen, help William find the *least possible server cost* by helping him schedule the K server upgrades.

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

Input

The first line contains two space-separated integers T and K, respectively: the duration of the contest and the maximum number of server upgrades that Giorgio can perform. The second line contains a single integer N, the number of users that will connect to the server. The third line contains N space-separated **sorted** integers t_i, representing the time when the i-th user will connect to the server (in seconds).

Output

You need to write a single line with an integer: the least possible total server cost (in euro cents).

Constraints

- $1 \leq K \leq N \leq 200$.
- $1 \leq T \leq 3600$, that is: the contest lasts for 1 hour at most.
- $0 \leq t_0 < t_1 < \cdots < t_{N-1} \leq T - 1$.

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** [**5 points**]: Examples.
- **Subtask 2** [**25 points**]: $K = N$.
- **Subtask 3** [**40 points**]: $N \leq 10$.
- **Subtask 4** [**30 points**]: No additional limitations.

Examples

input.txt	output.txt

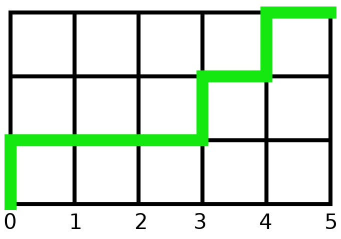
5 2
3
0 3 4
	9

5 2
3
0 1 4
	11

1000 1
1
600
	400

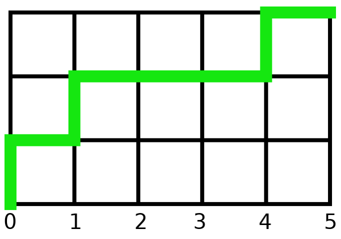
Explanation

In the **first sample case** the load of connected users looks like this:



The best we can do in this case is to increase the power by $+1$ at $t = 0$ and then by $+2$ at $t = 3$.

In the **second sample case** the load of connected users looks like this:



The best we can do in this case is to increase the power by $+2$ at $t = 0$ and then by $+1$ at $t = 4$.