Final Exam

This exam is:

- open notes/books/any printed resources

Online resources that you are allowed to access:

- Gradescope
- course website
- language documentation:
  - Java: https://docs.oracle.com/javase/10/docs/api/
  - C++: https://www.cplusplus.com/reference/ and/or https://en.cppreference.com/w/
- anything else you want since we cannot restrict it, but keep in mind that this will take the time away from the time you have to solve the problems
- Ed discussion forum, for asking questions during the exam.

Instructions:

Solve four out of the five problems given on the next pages. You will **not** get extra credit for solving all five problems. If you do attempt five problems, we will pick the four with the highest scores. For each problem, your last submission counts. If that last submission is made after your due time, it will not count and you will not get credit for that problem.

Grading:

Every exam problem is graded out of 10 points. The total exam grade is the weighted sum computed as follows (assume *scoreN* is a score for a particular problem with *score1 >= score2 >= score3 >= score4*):

*exam = 4 * score1 + 3 * score2 + 2 * score3 + 1 * score4*

The total score for a problem is determined by the maximum of zero and the sum of the scores for individual tests based on their results. The maximum score for each test is determined by *max_score = 10/number_of_tests*.

test outcome	test score
passed test	*max_score*
wrong answer	*- 0.5 max_score*
runtime error	*- 0.5 max_score*
timeout error	*- 0.5 max_score*
presentation error	*0.75 max_score*

Edit Distance

Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other.

In this problem, we consider a single operation: replace a single character with another character.

For example, we can transform "stock" to "spice" with the following steps:

"stock" -> "stick" -> "slick" -> "slice" -> "spice"

Each successive word differs from the previous word in only a single character position while the word length remains the same. The added restriction is to use the words from a given dictionary in each step of the transformation.

Given a dictionary and a list of word-pairs, write a program to determine the minimum number of steps in the shortest possible transformation for each pair.

Input

The first line contains an integer `n`, the number of words in the dictionary, `1 <= n <= 200`. Each of the following `n` lines contains a word in the dictionary. All words will be alphabetic and in lower case. And no word will be longer than 10 characters.

The following line contains an integer `m`, the number of queries. Each of the following `m` lines contain a pair of word separated by space. They represent the starting word and the ending word. All pairs are guaranteed to have a transformation using the dictionary given.

Output `m` lines of numbers: the i-th line represents the minimum number of steps in the transformation for the i-th pair of words.

Example 1

```
Input:
11
dip
lip
mad
map
may
pad
pip
pod
pop
sap
sip
4
may mad
map map
pip pod
mad dip


Output:
1
0
2
4
```

Follow the Path

You are programming a new robot called Frank to navigate the path through a grid. Frank should be able to adjust to direction it is moving in rather than being pre-programmed. This particular robot needs to navigate through a 2D grid. At each grid location Frank receives the instructions for the instructions for its next move. The possible moves are:

- N - move North (or up)
- S - move South (or down)
- E - move East (or right)
- W - move West (or left)

Your task now is to calculate what path Frank should take and verify that it actually took that path (otherwise you have a bug in the code that directs Frank through the grid).

Example 1:

```
              F
  N  E  E  S<-W  E
           v
<-W<-W  W  E->S  S
     ^        v
  S  N<-W<-W<-W  W
```

If Frank enters this grid at the grid location just below of the position of F, it will move west, then south, then east, then south, then west three times, then north and west two more times and finally it will exit the grid.

Example 2:

```
  F
  S  E  S<-W  E
  v     v  ^
  E->E->S  N<-W
        v     ^
  N  W  E->E->N

  N  W  S  W  N
```

If Frank enters this grid at the grid location just below of the position of F, it will move south, then east twice, and then it will enter a loop consisting of 8 moves. Frank will never leave this grid.

Input

On the first line are three integers separated by blanks: the number of rows in the grid, the number of columns in the grid, and the number of the column in which the robot enters from the north. The possible entry columns are numbered starting with one at the left.

Then come the rows of the direction instructions. Each grid will have at least one and at most 10 rows and columns of instructions. The lines of instructions contain only the characters N, S, E, or W with no blanks.

Output

There should be one line of output. Either the robot follows a certain number of instructions and exits the grid on any one the four sides or else the robot follows the instructions on a certain number of locations once, and then the instructions on some number of locations repeatedly. The sample input below corresponds to the two grids above and illustrates the two forms of output. The word “step” is always immediately followed by “(s)” whether or not the number before it is 1.

Example 1

```
Input:

3 6 5
NEESWE
WWWESS
SNWWWW


Output:

10 step(s) to exit
```

Example 2

```
Input:

4 5 1
SESWE
EESNW
NWEEN
EWSEN

Output:

3 step(s) before a loop of 8 step(s)
```

Desolate Number

Given two integers A and B, a desolate number N is defined as follows:

- N is an (A+B)-bit integer;
- the binary representation of N has exactly A 1's and B 0's (leading zeroes are ok);
- N has the maximum number of 1's adjacent to at least one 0 in its binary representation.

Your task is to find the smallest desolate number N given the values of A and B.

Input

The input consists of a single line, containing two non-negative integers A and B (1 <= A + B <= 50).

Output

Print one line, containing the smallest desolate number.

Example 1

```
Input:
1 1

Output:
1
```

Since we have two bits and one of them has to be zero and the other one has to be one, then the two options are `01` and `10`. They both have the same number of 1-bits adjacent to the 0-bits. But the first one is smaller, so the answer is 1.

Example 2

```
Input:
4 3

Output:
45
```

45 is binary using 7 bits is `0101101`. In this sequence every 1-bit is adjacent to at least one 0 bit sot this constraint is maximized. The only way to make this value smaller is to shift some of the 1-bits to the right, but this would decrease the number of 1-bits adjacent to the 0-bits.

Example 3

```
Input:
4 1

Output:
23
```

23 in binary using 5 bits is `10111`. With a single zero available at most two 1-bits can be adjacent to a 0-bit. If we move the 0-bit any further to the right, the value would increase.

HINT: You do not need to try every possible combination of the bits to solve this problem.

Definite Prime

A definite prime is a prime number for which we can rearrange the digits in any way and the result is also a prime. For example, 113, 131 and 311 are all prime, so we may call 131 a definite prime (and so are 113 and 311).

Given a positive number N, find the smallest definite prime that is larger than N and smaller than the next power of 10 greater than N.

Input

The input consists of only one line containing a single integer N (1 <= N < 10,000,000) as described above.

Output

Print a single line containing the definite prime or 0 if one does not exist.

Example 1

```
Input:
10

Output:
11
```

Example 2

```
Input:
16

Output:
17
```

Example 3

```
Input:
500

Output:
733
```

733 is the smallest definite prime larger than 500 and smaller than 1000. There are others, but they are greater than 733. All numbers 337, 373, 733 are prime.

Example 3

```
Input:
9990

Output:
0
```

There are no primes between 9,990 and 10,000.

Example 4

```
Input:
```

```
9000

Output:
0
```

None of the primes between 9,000 and 10,000 are definite.

CIMS Printer

For to some historical reasons there is only one printer still working at Courant Institute of Mathematical Sciences. There are tons of printing jobs queuing for that printer and you may have to wait for a very very long time to get your paper printed.

Printing jobs are assigned a priority between 1 and 9 (inclusive) where 9 is highest priority and 1 is the lowest. The printing system manages the jobs as follows.

- Select the first job from the queue.
- If there is some job in the queue with higher priority than the selected job then move the selected job to the end of the queue without printing it.
- Otherwise, print that job.

The problem is that this makes it tricky to know when your paper will be printed.

Your task is to write a program to figure out when a given job is printed assuming some knowledge of all the jobs sent to the printer. The program is given the current printing queue and the position of a job of interest in the queue. It should output how long it will take until that job is printed. Assume that

- there are no additional print jobs added to the queue,
- printing a single job takes exactly one minute (regardless of its size), and
- the queue operations happen instantly (i.e., there is no additional time spent manage the queue).

Input

The first line of input contains two integers `N` and `M` where `N` stands for the number of jobs in the print queue (`1 <= N <= 100`) and `M` is the position of the job `(0 <= M <= N - 1` ; with 0 being the first position in the queue). The second line has `N` integers in the range, giving the priorities of jobs in the queue from the first position to the last.

Output

You should print **one line** with a single number, the number of minutes until the job is printed.

Example 1

```
Input:
1 0
5

Output:
1
```

There is only one job and the printing is completed in one minute.

Example 2

```
Input:
4 2
1 2 3 4

Output:
2
```

The job of interest is the job with priority 3.

Queue operations: The first job has priority 1, so it is moved to the back of the queue (since there are jobs with higher priority in the queue). The next job has priority 2, so it is moved to the back of the queue. The next job has priority 3 and it is moved to the back of the queue. Finally, the next job has priority 4 and it will be printed.

Printing takes 1 minute.

Queue operations: The first job has priority 1, so it is moved to the back of the queue. The next job has priority 2, so it is moved to the back of the queue. Finally, the next job has priority 3 (this is the job of interest) and it will be printed.

Printing takes 1 minute.

In total, two minutes have passed for the job to be printed.

Example 3

```
Input:
6 0
1 1 9 1 1 1

Output:
5
```