1553 Buffer Manager

DBMS (Data Base Management System) development team has been successful in designing efficient Lock Manager and is going to proceed further. As a part of the team you will be responsible for the **Buffer Manager**.

Data blocks being read by DBMS from the hard drive are stored in the main memory in a fixed number of pre-allocated **buffers**. Each buffer can hold one data block. Each buffer can be either **free** (does not contain any useful information) or **occupied** by some data. When DBMS is going to read data block from the hard drive it has to decide which buffer to use for data storing. If there are any free buffers, then one of them is used for that purpose. If there are no free buffers, then one of the occupied buffers has to be flushed to become free, unless it was **locked** by some part of DBMS.

The choice of the buffer to flush is critical to DBMS performance. A lot of different algorithms were developed, LRU (Least Recently Used) algorithm being the one used most often. However, your DBMS is going to implement the Advanced Buffer Management algorithm which takes advantage of the fact that maximal performance is achieved when a number of consecutive data blocks from the hard drive are read into consecutive memory buffers.

Buffers are numbered from 1 to N, where N ($1 \leq N \leq 100000$) is a total number of buffers. Each buffer can be in any one of the following states: free, occupied or locked. Each occupied buffer is assigned an integer number from 1 to 9 - the **worthiness** of the currently stored information in that buffer. The worthiness of free buffers is considered to be zero. Locked buffers cannot be neither used nor flushed and their worthiness is undefined.

Having received the request to read K ($1 \leq K \leq 10000$) data blocks from the hard drive, Buffer Manager has to choose K consecutive non-locked buffers numbered from L to $L + K - 1$ that have minimal possible sum of their worthiness, or to report that it is impossible to find K consecutive non-locked buffers. The latter can also happen if total number of buffers is less than K.

Your task is to write a program that models the processing of one request to Buffer Manager using the above algorithm.

Input

Input consists of several datasets. The first line of each dataset contains two integers, N and K, separated by a space.

Starting from the second line there is a description of a buffers' state. The state of each buffer is represented by a single character:

- `0` — when the corresponding buffer is free.
- `1` — when the corresponding buffer is occupied and has worthiness of 1.
- `2` — when the corresponding buffer is occupied and has worthiness of 2.
- `...`
- `9` — when the corresponding buffer is occupied and has worthiness of 9.
- `*` — when the corresponding buffer is locked.

Those characters are situated on the consecutive lines grouped by 80 characters per line without any spaces. Thus, each line starting from the second one contains exactly 80 characters with a possible exception for the last line.

Output

For each test case, write to the output the single integer number L in a single line. This number gives the buffer number where first of the K blocks from the hard drive shall be read to ensure the minimal possible total worthiness of the blocks that have to be flushed. If there are more than one such value for L, then write the smallest one.

Write to the output file a single number '0' if it's impossible to find K consecutive non-locked buffers.

Sample Input

```
100 53
21657452160918534777558003938597858072075231699543 41**7363*9*94664808*4777717089
09825185827659480548
100 10
21657452160918534777558003938597858072075231699543 41**7363*9*94664808*4777717089
09825185827659480548
```

Sample Output

```
0
36
```