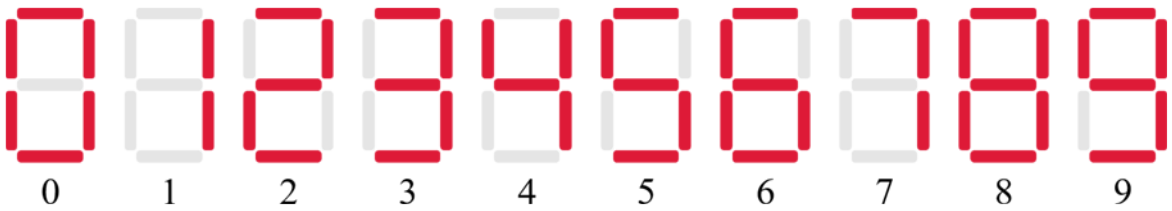
LED Display Renovation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You have an LED display capable of showing an n-digit integer, where each digit is represented using the standard 7-segment layout.



Initially, each of the 7 digit segments in every digit block may be in one of three states:

- Functional (denoted by '`W`'): behaves correctly, i.e., it lights up or turns off as required by the digit being displayed;
- Stuck ON (denoted by '`1`'): always lit regardless of the digit being displayed;
- Stuck OFF (denoted by '`0`'): always unlit regardless of the digit being displayed.

You can renovate at most k digit segments, i.e., convert at most k digit segments into functional. You need to maximize the number of integers that can be correctly displayed and find the number of ways to achieve the maximum. Note that you can renovate digit segments that are initially functional, and each digit segment can be renovated at most once. Two ways of renovation are considered different if there is at least one digit segment that is renovated in one way but not in the other.

An integer is correctly displayed if and only if:

- It has no leading zeros, unless the integer is exactly 0;
- It is right-aligned on the display: if the integer has d ($1 \le d \le n$) digits, then only the rightmost d digit blocks are used, and the leftmost $(n - d)$ blocks should be left blank and show nothing;
- For each of the d used digit blocks, all 7 digit segments match the target digit's display pattern when functional segments show the correct state for that digit and faulty segments remain stuck in their state.

Input

The first line of the input contains an integer T ($1 \le T \le 100$), denoting the number of test cases. For each test case:

The first line contains two integers n ($1 \le n \le 9$) and k ($0 < k < 7n$), denoting the number of digits the LED display can show and the number of digit segments you can repair.

The next 7 lines each contain a string of exactly $(5n - 1)$ characters, describing the initial state of the LED display in the form of an ASCII image.

The display consists of n digit blocks, each occupying 4 columns, separated by single-column gaps. All digit segments are represented by two identical characters of '`W`', '`0`', or '`1`', and all other positions are filled with the character '`.`' that are purely for formatting. See the sample input and the notes for details.

Output

For each test case, output a line containing two integers, denoting the maximum number of integers that can be correctly displayed by renovating at most k digit segments, as well as the number of ways to achieve the maximum.

Example

standard input
2 3 2 .00...00...00. 0..0.0..0.0..1 0..0.0..0.0..1 .00...00...00. 0..W.0..0.0..1 0..W.0..0.0..1 .00...00...00. 9 62 .WW...WW...WW...WW...WW...WW...WW...WW...WW. W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W .WW...WW...WW...WW...WW...WW...WW...WW...WW. W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W.W..W .WW...WW...WW...WW...WW...WW...WW...WW...WW.
standard output
2 23 1000000000 9223372036854775807

Note

In each digit block of the LED display in the form of an ASCII image, the digit segments are as follows:

Segment	Part	Positions (row, column)
0	Top horizontal	$(1,2)$, $(1,3)$
1	Upper-left vertical	$(2,1)$, $(3,1)$
2	Upper-right vertical	$(2,4)$, $(3,4)$
3	Middle horizontal	$(4,2)$, $(4,3)$
4	Lower-left vertical	$(5,1)$, $(6,1)$
5	Lower-right vertical	$(5,4)$, $(6,4)$
6	Bottom horizontal	$(7,2)$, $(7,3)$

The display patterns for digits $0, 1, 2, \ldots, 9$ (`1` = lit, `0` = unlit) are as follows:

Digit	**Pattern (segments** $0, 1, 2, \ldots, 6$**)**	**Explanation**
0	`1110111`	All except middle
1	`0010010`	Upper-right and lower-right
2	`1011101`	Top, upper-right, middle, lower-left, and bottom
3	`1011011`	Top, upper-right, middle, lower-right, and bottom
4	`0111010`	Upper-left, upper-right, middle, and lower-right
5	`1101011`	Top, upper-left, middle, lower-right, and bottom
6	`1101111`	All except upper-right
7	`1010010`	Top, upper-right, and lower-right
8	`1111111`	All segments
9	`1111011`	All except lower-left