An antique machine with $\binom{N}{3}$ switches capable of processing integers in the range $0..2^N-1$ has just been discovered. Each switch is associated to a distinct integer in $0..2^N-1$ with exactly three ones in its binary representation. By setting switches associated with number $X_0, X_1, \ldots, X_{M-1}$ to on, any integer Y passing through the machine will render a result of $Y \oplus X_0 \oplus X_1 \oplus \ldots \oplus X_{M-1}$ (here "$\oplus$" stands for bitwise-XOR).

We are interested in the number of configurations capable of transforming integer S into T with exactly K switches set to on. Could you write a program to help us?

Input

There are multiple test cases in the input file.

Each test case starts with two integers, N and K ($1 \le N \le 40$, $0 \le K \le \min\{20, \binom{N}{3}\}$) followed by two binary integers, S and T, each containing exactly N bits.

Two successive test cases are separated by a blank line. A case with $N = 0$ and $K = 0$ indicates the end of the input file, and should not be processed by your program.

Output

For each test case, please print a single integer, the total number of ways to transform the first integer into the second one. Since the answer could be quite large, you are only required to find the result *module* 10007.

Sample Input

```
4 3
1101
1001

3 1
101
010

5 3
11010
10111

0 0
```

Sample Output

```
Case #1: 1
Case #2: 1
Case #3: 6
```