Goodifying Strings

FWMC receives a string s of length n, consisting of characters $s_0, s_1, \ldots, s_{n-1}$, as a birthday gift. Initially, every character in s is the letter '`A`'.

FWMC then starts playing around with the string. The process can be described as a sequence of q queries, each is one of the following two types:

- `1 l r c` — FWMC assigns all positions i in the range $[l, r]$ to the character c (that is, set $s_i = c$ for all $l \le i \le r$).
- `2 k` — FWMC dislikes long sequences of consecutive identical characters. FWMV considers a string **good** if there is no consecutive substring[1] of more than k identical characters. Considering the current string s, FWMC wants to determine the minimum number of characters that must be deleted to make s a **good** string.

  Note that FWMC just wants to know the minimum number of deleted characters, but does not make any changes to s.

Your task is to help FWMC determine the answer to each query of the second type.

Input

The first line contains two integers n and q ($1 \le n, q \le 300\,000$) — the length of the string and the number of queries.

Each of the next q lines describes a query. The first integer is t ($1 \le t \le 2$) —— the type of the query.

- If $t = 1$, two integers l and r ($0 \le l \le r < n$) and an uppercase Latin letter c are given.
- If $t = 2$, an integer k ($1 \le k \le n$) is given.

Output

For each query of the second type, output a single integer — the minimum number of characters that need to be deleted to make the string **good**.

[1] A string α is a substring of a string β if α can be obtained from β by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Sample Input 1	Sample Output 1
11 10 1 0 1 F 1 2 10 C 1 3 8 W 1 9 9 M 1 4 6 M 2 2 2 1 2 5 1 5 7 C 2 1	1 4 0 3

Sample Input 2	Sample Output 2
12 5 1 3 7 G 1 7 11 G 1 4 5 C 1 0 6 C 2 2	8

Sample Explanation

The following is the explanation for the first test case.
In the first five queries, the string changes as follows:

$$\texttt{AAAAAAAAAA} \to \texttt{FFAAAAAAAA}$$
$$\texttt{FFAAAAAAAA} \to \texttt{FFCCCCCCCC}$$
$$\texttt{FFCCCCCCCC} \to \texttt{FFCWWWWWWC}$$
$$\texttt{FFCWWWWWWC} \to \texttt{FFCWWWWWWMC}$$
$$\texttt{FFCWWWWWWMC} \to \texttt{FFCWMMMWWMC}$$

In the sixth query, it is sufficient to delete one character:

$$\texttt{FFCWMMMWWMC} \to \texttt{FFCWMMWWMC}$$

The groups of consecutive identical characters are FF | C | W | M | MM | WW | M | C, none of which contain more than $k = 2$ identical characters, thus the final string is **good**.
In the seventh query, with $k = 1$, one possible solution to delete 4 characters is:

$$\texttt{FFCWMMMWWMC} \to \texttt{FCWMWMC}$$

It can be shown that there exists no solution that delete fewer characters.
In the eighth query, with $k = 5$, the current string s is already a good string, so no deletions are needed.
In the ninth query, the string changes as follows:

$$\texttt{FFCWMMMWWMC} \to \texttt{FFCWMCCCWMC}$$

In the tenth query, with $k = 1$, one possible solution to delete 3 characters is:

$$\texttt{FFCWMCCCWMC} \to \texttt{FCWMCWMC}$$

It can be shown that there exists no solution that delete fewer characters.