A: Auto-Coin-o-Matic

Time Limit: 2 seconds, Memory limit: 2G

It's finally here! The day you unveil your new invention, the Auto-Coin-o-Matic! You watch with glee and anxiety as people insert their card into the machine, type in the amount they want, and get exact change out with the fewest number of coins.

But was it actually the fewest number of coins? That's how it was programmed, but what if you had a bug? It's okay, you're watching. You decide to randomly pick some transactions and double check that what the machine gave out is indeed the fewest number of coins possible. But, oh no, the machine is running out of certain types of coins! Will it still work correctly?

Input

The input starts with two integers n and m ($1 \leq n \leq 2000$, $1 \leq m \leq 10^5$).

The next line contains n integers, $d_1, d_2, \ldots, d_n$ ($1 \leq d_i \leq 10^5$) representing the denominations of coins available initially. It is guaranteed that all denominations are unique.

The next m lines contain a character c ($c \in \{\texttt{Q}, \texttt{X}\}$) and an integer v ($1 \leq v \leq 10^5$), where c is the type of event and v is the value of the event.

- If c is the character `Q`, this is a query and the output should be the minimum number of coins needed to give out exactly v. It is guaranteed that there will be at least one query.

  If it is not possible to make v exactly with the available denominations, output -1 instead.

- If c is the character `X`, this means the machine is out of coins of denomination v. All queries after this point cannot use this denomination. It is guaranteed that each `X` event corresponds to a denomination v which the machine currently has in stock.

Output

Output k lines, where k is the number of query events ($c = \texttt{Q}$). On the i^{th} line, output the fewest number of coins needed to give change for the i^{th} query, or -1 if this is impossible.

Sample Input 1	Sample Output 1
4 7 1 2 5 10 Q 23 X 1 Q 23 X 5 Q 23 X 10 Q 22	4 6 -1 11