Trie

Task 1

Implement the basic operations of the ‘Trie’ data structure by implementing the following functions:

- void **insert**(): Inserts a string in a trie
- boolean **search**(): Returns if the query string is a valid word.
- void **display**(): Shows all the words that are stored in the Trie in lexicographically sorted order.

As for the Trie data structure itself, you can use either structs or classes. It should look similar to an 26-ary tree, where each node in the tree has 26 child pointers corresponding to 26 letters of the English alphabet. There should be a boolean variable called “is_word” in each node which will be TRUE if the letters starting from the root to this node correspond to an actual word in the dictionary.

Next, test your implemented Trie data structure. The first line of input contains two integers N and Q. The next line of input contains N space-separated words that need to be inserted in the Trie. Once the words are inserted, display all of them.

The following line contains Q query words. Print **T/F** based on their presence/absence.

Sample Input	**Sample Output**
9 6 toy algo algorithm to tom also algae tommy toyota	 algae algo algorithm also to tom tommy toy toyota
toy toys al also algorithm algorithmic	T F F T T F

Task 2

Suppose a set of words is stored in a Dictionary. Given a *prefix*, your task is to find out how many words start with it.

The first input line will contain *N* and *Q*, where *N* represents the number of words in the dictionary, and Q is the number of queries. Print the number of words starting with each corresponding prefix.

Sample Input	**Sample Output**
10 10 Beauty Beast Beautiful Amazing Amsterdam Beautify Banana Xray Beauty Glorifying A Am AM Beauty Beaut Beast Ing AMS Be B	 2 2 2 1 3 1 0 1 4 5

Note: Convert every string/prefix in lowercase before storing/ searching. Don’t forget to handle duplicate entries.

Task 3

You are given a set of ‘products’ and a string ‘searchWord’. Design a solution that suggests **at most three** products after each character of searchWord is typed. Suggested products should have a common prefix with searchWord. If there are more than three products with a common prefix, follow the lexicographical order. If the search word matches with no products then print “Null”.

The first line of input contains space-separated strings indicating the set of products. The second line contains a single search word.

Input (products)	**Output**	**Explanation (searchWord)**
mobile mouse moneypot monitor mousepad mouse	mobile moneypot monitor mobile moneypot monitor mouse mousepad mouse mousepad mouse mousepad	**‘m’** **‘mo’** **‘mou’** (only 2 matches) **‘mous’** (only 2 matches) **‘mouse’** (only 2 matches)
havana havana	havana havana havana havana havana havana	**‘h’** **‘ha’** **‘hav’** **‘hava’** **‘havan’** **‘havana’**
juice jeerapani icecream jelly jam jackfruit jalapeno jeans	jackfruit jalapeno jam Jelly jeerapani Null Null Null	**‘J’**: 6 words matched. Printed only the first 3 in lexicographical order. **‘Je’**: 2 matches No match found for **‘jea’**, **‘jean’**, **‘jeans’**. Hence null.