**CSE 4304-Data Structures Lab. Winter 23-24**

**Batch:** CSE 22

**Date**: November 20, 2024

**Target Group:** All

**Topic**: Trie

**Instructions**:

* Regardless of how you finish the lab tasks, you must submit the solutions in Google Classroom. In case I forget to upload the tasks there, CR should contact me. The deadline will always be 11:59 PM on the day the lab took place.
* Task naming format: fullID\_T01L01\_2A.c/cpp
* If you find any issues in the problem description/test cases, comment in the Google Classroom.
* If you find any tricky test cases that I didn’t include but that others might forget to handle, please comment! I’ll be happy to add them.
* Use appropriate comments in your code. This will help you recall the solution easily in the future.
* Obtained marks will vary based on the efficiency of the solution.
* Do not use <bits/stdc++.h> library.
* Modified sections will be marked with BLUE color.
* You can use the STL stack unless it’s specifically mentioned that you should use manual functions.

| **Group** | **Tasks** |
| --- | --- |
| 2A | 1 2 3 4 (15 marks) |
| 1B | 1 2 3 4 (15 marks) |
| 1A |  |
| 2B |  |
| **Assignments** | 2A/1B:  1A/2B: |

**Task 1**: Basic operations of Trie data structure

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.

The first line of input contains space-separated words that need to be inserted in the Trie. Once the words are inserted, display all of them.

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

| **Sample Input** | **Sample Output** |
| --- | --- |
| toy algo algorithm to tom also algea tommy toyota  toy toyo al also algorithm algorithmic | algea algo algorithm also to tom tommy toy toyota  T F F T T F |

**Task 2**: **Find the number of words starting with a certain prefix**

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**: Search Suggestions

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.

| **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 | **‘h’**  **‘ha’**  **‘hav’**  **‘hava’**  **‘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. |

**Task 4 (15 marks)**

You are given a collection of ‘n’ non-negative integers. Your task is to find the **maximum XOR value** between any two numbers in the collection. Implement an efficient solution to solve this problem.

| **Input** | **Output** | **Explanation** |
| --- | --- | --- |
| 3 10 5 25 2 8 | 28 | The maximum XOR is achieved between 5 (binary: 0101) and 25 (binary: 11001). The XOR result is 28 (binary: 11100). |
| 0 1 2 3 4 | 7 | Achieved between 3 (binary: 11) and 4 (binary: 100). The XOR result is 7 (binary: 111). |
| 8 1 2 15 | 14 | The maximum XOR is achieved between 1 (binary: 0001) and 15 (binary: 1111). The XOR result is 15 (binary: 1111). |
| 1 1 1 1 | 0 | All numbers are the same, so the XOR of any two numbers is 0. |
| 5 25 10 2 8 12 | 29 |  |
| 0 | 0 |  |
|  |  |  |

**Hint**:

* Trie can be used to find solution in O(N)
* Each node in the Trie represents a bit (0 or 1) of the binary representation of numbers.
* Insert each number into the Trie bit by bit, from the most significant bit (MSB) to the least significant bit (LSB).
* Finding Maximum XOR:
  + For each number, traverse the Trie to find the number that gives the maximum XOR with the current number. This is done by attempting to match opposite bits (e.g., if the current bit is 0, look for 1).