**TRIE IMPLEMENTATION**

public class Trie {

    static final int ALPHABET\_SIZE = 26;

    // trie node

    static class TrieNode

    {

        TrieNode[] children = new TrieNode[ALPHABET\_SIZE];

        // isEndOfWord is true if the node represents

        // end of a word

        boolean isEndOfWord;

        TrieNode(){

            isEndOfWord = false;

            for (int i = 0; i < ALPHABET\_SIZE; i++)

                children[i] = null;

        }

    };

    static TrieNode root;

    // If not present, inserts key into trie

    // If the key is prefix of trie node,

    // just marks leaf node

    static void insert(String key)

    {

        int level;

        int length = key.length();

        int index;

        TrieNode pCrawl = root;

        for (level = 0; level < length; level++)

        {

            index = key.charAt(level) - 'a';

            if (pCrawl.children[index] == null)

                pCrawl.children[index] = new TrieNode();

            pCrawl = pCrawl.children[index];

        }

        // mark last node as leaf

        pCrawl.isEndOfWord = true;

    }

// Returns true if key presents in trie, else false

    static boolean search(String key)

    {

        int level;

        int length = key.length();

        int index;

        TrieNode pCrawl = root;

        for (level = 0; level < length; level++)

        {

            index = key.charAt(level) - 'a';

            if (pCrawl.children[index] == null)

                return false;

            pCrawl = pCrawl.children[index];

        }

        return (pCrawl != null && pCrawl.isEndOfWord);

    }

        else System.out.println("thaw --- " + output[0]);

    }

}