**Implement Trie ll**

#include <bits/stdc++.h>

struct Node{

    Node\* links[26];

    bool flag = false;

    int count=0;

    int endWith =0;

    bool contains(char ch ){

        return(links[ch-'a']!= NULL);

    }

    void put (char ch , Node \* node){

        links[ch-'a']= node;

    }

    Node \* get(char ch ){

        return links[ch -'a'];

    }

    void increaseEnd() { endWith++; }

    void increasePrefix(){count++;

    }

    void deleteend(){

        endWith--;

    }

    void reducePrefix(){

        count--;

    }

    int getend(){

        return endWith;

    }

    int getPrefix(){

        return count;

    }

}

;

class Trie{

    private:

    Node\* root;

    public:

    Trie(){

        // Write your code here.

        root=  new Node();

    }

    void insert(string &word){

        // Write your code here

        Node \* node = root;

        for (int i =0;i< word.length(); i++){

            if (!node-> contains(word[i])){

                node-> put(word[i],new Node());

            }

            node = node-> get (word[i]);

            node->increasePrefix();

        }

        node->increaseEnd();

    }

    int countWordsEqualTo(string &word){

        // Write your code here.

        Node \* node = root;

        for(int i =0; i< word.length(); i++){

            if (node-> contains(word[i])){

                node= node-> get(word[i]);

            }

            else{

                return 0;

            }

        }

        node->getend();

    }

    int countWordsStartingWith(string &word){

        // Write your code here.

        Node\*node= root;

        for (int i =0; i< word.length(); i++){

            if (node-> contains(word[i])){

                node = node-> get(word[i]);

            }

            else{

                return 0;

            }

        }

        return node-> getPrefix();

    }

    void erase(string &word){

        // Write your code here.

        Node \*node = root;

        for (int i =0; i< word.length(); i++){

            if ( node-> contains(word[i])){

                node= node-> get(word[i]);

                node-> reducePrefix();

            }

            else{

                return ;

            }

        }

        node-> deleteend();

    }

};