**Trie**

**Problem-1:** There are four functionalities. Insert a new word, count number of words equal to particular word, count number of words start with particular word or prefix, erase a word.

struct TrieNode{

TrieNode\* links[26];

int count\_Ends\_with=0;

int count\_starts\_with=0;

**bool Contains\_key(char ch)**{ if(links[ch-'a']!=NULL) return true; else return false; }

**void Put\_New\_Node(char ch, TrieNode\* node)**{ links[ch-'a']=node; }

**TrieNode\* Get(char ch)** { return links[ch-'a']; }

**void increase\_end()**{ count\_Ends\_with++; }

**void increase\_prefix()** { count\_starts\_with++; }

**void delete\_end()**{ count\_Ends\_with--; }

**void reduce\_prefix()** { count\_starts\_with--; }

**int Get\_End()** { return count\_Ends\_with; }

**int Get\_Prefix()** { return count\_starts\_with; }

};

**class Trie**{

private: TrieNode\* root;

public:

Trie(){ root=new TrieNode(); //New root will be created } //Constructor. When a trie is created one root will be created by default.

**void insert(string &word)** { TrieNode\* node=root;

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

if(!node->Contains\_key(word[i])){ node->Put\_New\_Node(word[i], new TrieNode()); }

node=node->Get(word[i]);node->increase\_prefix();

} node->increase\_end(); }

**int words\_equal\_to(string &word)** { TrieNode\* node=root;

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

if(!node->Contains\_key(word[i])){ return 0;} node=node->Get(word[i]); }

return node->Get\_End(); }

**int words\_starts\_with(string &word**) { TrieNode\* node=root;

for(int i=0; i<word.size(); i++){ if(!node->Contains\_key(word[i])) return false;

node=node->Get(word[i]); } return node->Get\_Prefix(); }

**void erase(string &word)** {TrieNode\* node=root; for(int i=0; i<word.size(); i++){

if(node->Contains\_key(word[i])){ node->Get(word[i]); node->reduce\_prefix();} else return; }

node->delete\_end();} };