**SOLUTION**

class Trie {

Trie\* children[26];

bool isEnd;

public:

/\*\* Initialize your data structure here. \*/

Trie() {

isEnd = false;

for(int i=0; i<26 ; i++)

children[i] = nullptr;

}

/\*\* Inserts a word into the trie. \*/

void insert(string word) {

Trie\* curr = this;

for (char c : word){

if(!(curr->children[c - 'a']))

curr->children[c-'a'] = new Trie;

curr = curr->children[c-'a'];

}

curr->isEnd = true;

}

/\*\* Returns if the word is in the trie. \*/

bool search(string word) {

Trie\* curr = this;

for(char c : word){

curr = curr->children[c-'a'];

if(!curr)

return false;

}

if(curr->isEnd)

return true;

return false;

}

/\*\* Returns if there is any word in the trie that starts with the given prefix. \*/

bool startsWith(string prefix) {

Trie\* curr = this;

for(char c : prefix){

curr = curr->children[c-'a'];

if(!curr)

return false;

}

return true;

}

};

/\*\*

\* Your Trie object will be instantiated and called as such:

\* Trie\* obj = new Trie();

\* obj->insert(word);

\* bool param\_2 = obj->search(word);

\* bool param\_3 = obj->startsWith(prefix);

\*/

**TIME COMPLEXITY: O(N)**

**SPACE COMPLEXITY: O(1)**