|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: SY | Division: B | | Roll No: 272028 |
| Semester: IV | | Academic Year: 2022-2023 | |
| Subject Name & Code: Advance Data Structure: ADUA22202 | | | |
| Title of Assignment: Assignment based on Tries | | | |

**ASSIGNMENT NO. 10**

**PROGRAM AND OUTPUT:**

#include <iostream>

#include <unordered\_map>

using namespace std;

class TrieNode

{

public:

    unordered\_map<char, TrieNode \*> children;

    bool isEndOfWord;

    TrieNode()

    {

        isEndOfWord = false;

    }

};

class Trie

{

public:

    TrieNode \*root;

    Trie()

    {

        root = new TrieNode();

    }

    void insert(string word)

    {

        TrieNode \*curr = root;

        for (char c : word)

        {

            if (curr->children.count(c) == 0)

            {

                curr->children[c] = new TrieNode();

            }

            curr = curr->children[c];

        }

        curr->isEndOfWord = true;

    }

    bool search(string word)

    {

        TrieNode \*curr = root;

        for (char c : word)

        {

            if (curr->children.count(c) == 0)

            {

                return false;

            }

            curr = curr->children[c];

        }

        return curr->isEndOfWord;

    }

    bool deleteWord(string word, TrieNode \*curr, int idx)

    {

        if (idx == word.length())

        {

            if (curr->isEndOfWord == false)

            {

                return false;

            }

            curr->isEndOfWord = false;

            return curr->children.empty();

        }

        char c = word[idx];

        if (curr->children.count(c) == 0)

        {

            return false;

        }

        bool shouldDeleteNode = deleteWord(word, curr->children[c], idx + 1);

        if (shouldDeleteNode)

        {

            curr->children.erase(c);

            return curr->children.empty();

        }

        return false;

    }

};

int main()

{

    Trie trie;

    trie.insert("hello");

    trie.insert("world");

    trie.insert("hi");

    trie.insert("bye");

    cout << "Searching hello in tree : " << trie.search("hello") << endl;

    cout << "Searching world in tree : " << trie.search("world") << endl;

    cout << "Searching morning in tree : " << trie.search("morning") << endl;

    cout << "Searching bye in tree : " << trie.search("bye") << endl;

    cout << "Searching hey in tree : " << trie.search("hey") << endl;

    cout << "-----------------------------------------------------------------------------------------------------------" << endl;

    cout << "Deleting hello from tree : " << trie.deleteWord("hello", trie.root, 0) << endl;

    cout << "Deleting hi from tree : " << trie.deleteWord("hi", trie.root, 0) << endl;

    cout << "-----------------------------------------------------------------------------------------------------------" << endl;

    cout << "Search for deleted word hello : " << trie.search("hello") << endl;

    cout << "Search for deleted word hi : " << trie.search("hi") << endl;

    cout << "-----------------------------------------------------------------------------------------------------------" << endl;

    cout << "Search for remaining word world : " << trie.search("world") << endl;

    cout << "Search for remaining word bye : " << trie.search("bye") << endl;

    return 0;

