Name: Gaurav Kisan Pawar

Class: SE-III (Q-Batch)

Roll No.: 27

**Code:**

#include <iostream>

#include <string>

using namespace std;

struct Node {

    string k;

    string m;

    Node\* left;

    Node\* right;

};

class Dictionary {

public:

    Node\* root;

    Dictionary(){

        root = NULL;

    }

    Node\* createNode(string k, string m) {

        Node\* newNode = new Node;

        newNode->k = k;

        newNode->m = m;

        newNode->left = newNode->right = NULL;

        return newNode;

    }

    Node\* insert(Node\* root, string k, string m) {

        if (!root)

            return createNode(k, m);

        if (k < root->k)

            root->left = insert(root->left, k, m);

        else if (k > root->k)

            root->right = insert(root->right, k, m);

        return root;

    }

    Node\* findMin(Node\* node) {

        while (node->left != NULL)

            node = node->left;

        return node;

    }

    Node\* deleteNode(Node\* root, string k) {

        if (!root)

            return root;

        if (k < root->k)

            root->left = deleteNode(root->left, k);

        else if (k > root->k)

            root->right = deleteNode(root->right, k);

        else {

            if (!root->left) {

                Node\* temp = root->right;

                delete root;

                return temp;

            } else if (!root->right) {

                Node\* temp = root->left;

                delete root;

                return temp;

            }

            Node\* temp = findMin(root->right);

            root->k = temp->k;

            root->m = temp->m;

            root->right = deleteNode(root->right, temp->k);

        }

        return root;

    }

    void display(Node\* root) {

       if (root->left != NULL) {

           display(root->left);

       }

       cout << "Keyword: " << root->k << " | Meaning: " << root->m << endl;

       if (root->right != NULL) {

           display(root->right);

       }

   }

    int maxComparisonsUtil(Node\* root, string k, int count) {

        if (!root)

            return count;

        if (root->k == k)

            return count + 1;

        else if (k < root->k)

            return maxComparisonsUtil(root->left, k, count + 1);

        else

            return maxComparisonsUtil(root->right, k, count + 1);

    }

};

int main() {

    Dictionary dict;

    int choice;

    string k, m;

    do {

        cout << "\nMenu:\n";

        cout << "1. Add\n";

        cout << "2. Delete\n";

        cout << "3. Update\n";

        cout << "4. Display\n";

        cout << "5. Search\n";

        cout << "6. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter key: ";

                cin >> k;

                cout << "Enter meaning: ";

                cin>> m;

                dict.root = dict.insert(dict.root, k, m);

                cout << "Entry added successfully.\n";

                break;

            case 2:

                cout << "Enter key to delete: ";

                cin >> k;

                dict.root = dict.deleteNode(dict.root, k);

                cout << "Entry deleted successfully.\n";

                break;

            case 3:

                cout << "Enter key to update: ";

                cin >> k;

                cout << "Enter new meaning: ";

                cin>>m;

                dict.root = dict.deleteNode(dict.root, k);

                dict.root = dict.insert(dict.root, k, m);

                cout << "Entry updated successfully.\n";

                break;

            case 4:

                cout<<"\n";

                dict.display(dict.root);

                break;

            case 5:

                cout << "Enter key to find maximum comparisons: ";

                cin >> k;

                cout << "Maximum comparisons required: " << dict.maxComparisonsUtil(dict.root, k, 0) << endl;

                break;

            case 6:

               cout << "Thank You...\n";

                break;

            default:

                cout << "Invalid choice...\n";

        }

    } while (choice != 6);

    return 0;

}

**Output:**

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 1

Enter key: d

Enter meaning: dog

Entry added successfully.

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 1

Enter key: b

Enter meaning: bag

Entry added successfully.

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 1

Enter key: f

Enter meaning: ferarri

Entry added successfully.

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 1

Enter key: h

Enter meaning: house

Entry added successfully.

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 4

Keyword: b | Meaning: bag

Keyword: d | Meaning: dog

Keyword: f | Meaning: ferarri

Keyword: h | Meaning: house

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 3

Enter key to update: b

Enter new meaning: ball

Entry updated successfully.

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 4

Keyword: b | Meaning: ball

Keyword: d | Meaning: dog

Keyword: f | Meaning: ferarri

Keyword: h | Meaning: house

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 5

Enter key to find maximum comparisons: h

Maximum comparisons required: 3

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 2

Enter key to delete: h

Entry deleted successfully.

Menu:

1. Add

2. Delete

3. Update

4. Display

5. Search

6. Exit

Enter your choice: 4

Keyword: b | Meaning: ball

Keyword: d | Meaning: dog

Keyword: f | Meaning: ferarri