```cpp
#include <iostream>
#include <cstring>
using namespace std;

typedef struct Node {
    char k[20];
    char m[20];
    Node* lnode;
    Node* rnode;
} Node;

class Dictionary {
public:
    Node* root;
    void create();
    void display(Node*);
    void addRecord(Node*, Node*);
    int search(Node*, char[]);
    int update(Node*, char[]);
    Node* delNode(Node*, char[]);
    Node* min(Node*);
};

void Dictionary::create() {
    Node* temp;
    int ch;
    do {
        temp = new Node;
        cout << "\nEnter the Keyword : ";
        cin >> temp->k;
        cout << "Enter Meaning of the Keyword : ";
        cin >> temp->m;
        temp->lnode = NULL;
        temp->rnode = NULL;
        if (root == NULL) {
            root = temp;
        } else {
            addRecord(root, temp);
        }
        cout << "\nDo you want to add more records? (1 for yes, 0 for no): ";
        cin >> ch;
    } while (ch == 1);
}
```

```
void Dictionary::addRecord(Node* root, Node* temp) {
    if (strcmp(temp->k, root->k) < 0) {
        if (root->lnode == NULL)
            root->lnode = temp;
        else
            addRecord(root->lnode, temp);
    } else {
        if (root->rnode == NULL)
            root->rnode = temp;
        else
            addRecord(root->rnode, temp);
    }
}

void Dictionary::display(Node* root) {
    if (root != NULL) {
        display(root->lnode);
        cout << "\nKeyword: " << root->k;
        cout << "\tMeaning: " << root->m;
        display(root->rnode);
    }
}

int Dictionary::search(Node* root, char k[20]) {
    int c = 0;
    while (root != NULL) {
        c++;
        if (strcmp(k, root->k) == 0) {
            cout << "\nNumber of Comparisons: " << c;
            return 1;
        }
        if (strcmp(k, root->k) < 0)
            root = root->lnode;
        else
            root = root->rnode;
    }
    return -1;
}

int Dictionary::update(Node* root, char k[20]) {
    while (root != NULL) {
        if (strcmp(k, root->k) == 0) {
```

```
            cout << "\nEnter New Meaning of Keyword " << root->k << ": ";
            cin >> root->m;
            return 1;
        }
        if (strcmp(k, root->k) < 0)
            root = root->lnode;
        else
            root = root->rnode;
    }
    return -1;
}

Node* Dictionary::delNode(Node* root, char k[20]) {
    Node* temp;
    if (root == NULL) {
        cout << "\nElement Not Found";
        return root;
    }
    if (strcmp(k, root->k) < 0) {
        root->lnode = delNode(root->lnode, k);
        return root;
    }
    if (strcmp(k, root->k) > 0) {
        root->rnode = delNode(root->rnode, k);
        return root;
    }
    if (root->rnode == NULL && root->lnode == NULL) {
        temp = root;
        delete temp;
        return NULL;
    }
    if (root->rnode == NULL) {
        temp = root;
        root = root->lnode;
        delete temp;
        return root;
    }
    if (root->lnode == NULL) {
        temp = root;
        root = root->rnode;
        delete temp;
        return root;
    }
```

```
    temp = min(root->rnode);
    strcpy(root->k, temp->k);
    root->rnode = delNode(root->rnode, temp->k);
    return root;
}

Node* Dictionary::min(Node* q) {
    while (q->lnode != NULL) {
        q = q->lnode;
    }
    return q;
}

int main() {
    int ch;
    Dictionary d;
    d.root = NULL;
    do {
        cout << "\n1: Create\n2: Display\n3: Search\n4: Update\n5: Delete\n6:
        cout << "\nSelect your choice: ";
        cin >> ch;
        switch (ch) {
            case 1:
                d.create();
                break;
            case 2:
                if (d.root == NULL)
                    cout << "\nDictionary is empty.";
                else
                    d.display(d.root);
                break;
            case 3:
                if (d.root == NULL)
                    cout << "\nDictionary is empty.";
                else {
                    char k[20];
                    cout << "\nEnter Keyword to search: ";
                    cin >> k;
                    if (d.search(d.root, k))
                        cout << "\nKeyword Found";
                    else
                        cout << "\nKeyword Not Found";
                }
```

```
                break;
            case 4:
                if (d.root == NULL)
                    cout << "\nDictionary is empty.";
                else {
                    char k[20];
                    cout << "\nEnter Keyword which meaning you want to update
                    cin >> k;
                    if (d.update(d.root, k) == 1)
                        cout << "\nMeaning Updated";
                    else
                        cout << "\nMeaning Not Found";
                }
                break;
            case 5:
                if (d.root == NULL)
                    cout << "\nDictionary is empty.";
                else {
                    char k[20];
                    cout << "\nEnter Keyword you want to delete: ";
                    cin >> k;
                    d.root = d.delNode(d.root, k);
                }
                break;
            case 6:
                cout << "\nExiting...";
                break;
            default:
                cout << "\nInvalid choice";
        }
    } while (ch != 6);
    return 0;
}
```