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

struct Node {
   char key[50];
   char info[100];
   Node *left, *right;

   Node(const char* k, const char* m) {
      strcpy(key, k);
      strcpy(info, m);
      left = right = nullptr;
   }
};

void insert(Node* &root, Node* t) {
   if (root == nullptr) {
      root = t;
   } else {
      int l1 = strlen(root->key);
      int l2 = strlen(t->key);
      int minLen = min(l1, l2);
      int i = 0;
      while (i < minLen && root->key[i] == t->key[i]) {
         i++;
      }
      if (i < minLen && root->key[i] > t->key[i]) {
```

```
            insert(root->left, t);
        } else {
            insert(root->right, t);
        }
    }
}

void preorder(Node *temp) {
    if (temp != nullptr) {
        cout << temp->key << ": " << temp->info << " ";
        preorder(temp->left);
        preorder(temp->right);
    }
}

void postorder(Node *temp) {
    if (temp != nullptr) {
        postorder(temp->left);
        postorder(temp->right);
        cout << temp->key << ": " << temp->info << " ";
    }
}

void inorder(Node *temp) {
    if (temp != nullptr) {
        inorder(temp->left);
        cout << temp->key << ": " << temp->info << " ";
        inorder(temp->right);
```

```
    }
}

void dorder(Node *temp) {
    if (temp != nullptr) {
        dorder(temp->right);
        cout << temp->key << ": " << temp->info << " ";
        dorder(temp->left);
    }
}

Node* search(Node *root, char *k) {
    while (root != nullptr) {
        int l1 = strlen(root->key);
        int l2 = strlen(k);
        int minLen = min(l1, l2);
        int i = 0;
        while (i < minLen && root->key[i] == k[i]) {
            i++;
        }
        if (i == l2 && strcmp(root->key, k) == 0) {
            return root;
        } else if (i < l2 && root->key[i] > k[i]) {
            root = root->left;
        } else {
            root = root->right;
        }
    }
```

```
    return nullptr;
}

int main() {
    Node* root = nullptr;
    int choice;

    do {
        cout << "\nMenu:\n";
        cout << "1. Insert a new node\n";
        cout << "2. Preorder Traversal\n";
        cout << "3. Postorder Traversal\n";
        cout << "4. Inorder Traversal / Ascending order\n";
        cout << "5. Search\n";
        cout << "6. Descending order\n";
        cout << "7. Update\n";
        cout << "8. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                char key[50], info[100];
                cout << "Enter key: ";
                cin >> key;
                cout << "Enter info: ";
                cin >> info;
                Node* newNode = new Node(key, info);
```

```
    insert(root, newNode);
    break;
}
case 2: {
    cout << "Preorder traversal: ";
    preorder(root);
    cout << endl;
    break;
}
case 3: {
    cout << "Postorder traversal: ";
    postorder(root);
    cout << endl;
    break;
}
case 4: {
    cout << "Ascending order: ";
    inorder(root);
    cout << endl;
    break;
}
case 5: {
    char key[50];
    cout << "Enter the key you have to search: ";
    cin >> key;
    Node *temp = search(root, key);
    if (temp != nullptr) {
        cout << temp->key << ": " << temp->info << " ";
```

```
        cout << "Key is found";
    } else {
        cout << "Key not found";
    }
    cout << endl;
    break;
}
case 6: {
    cout << "Descending order: ";
    dorder(root);
    cout << endl;
    break;
}
case 7: {
    char key[50];
    cout << "Enter the value of key which needs to be updated: ";
    cin >> key;
    char info[100];
    cout << "Enter the updated information: ";
    cin >> info;
    Node *temp = search(root, key);
    if (temp != nullptr) {
        strcpy(temp->info, info);
        cout << "Information updated." << endl;
    } else {
        cout << "Key not found." << endl;
    }
    break;
```

```
            }
            case 8: {
                cout << "Exiting program." << endl;
                break;
            }
            default:
                cout << "Invalid choice." << endl;
        }

    } while (choice != 8);

    return 0;
}
```