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

struct node {
    int data;
    node* left;
    node* right;
};

node* createNode(int value) {
    node* newNode = new node;
    newNode->data = value;
    newNode->left = nullptr;
    newNode->right = nullptr;
    return newNode;
}

void insert(node*& root, node* temp) {
    if (root == nullptr) {
        root = temp;
    } else {
        if (root->data > temp->data) {
            if (root->left == nullptr) {
                root->left = temp;
            } else {
                insert(root->left, temp);
            }
        } else if (temp->data > root->data) {
            if (root->right == nullptr) {
                root->right = temp;
            } else {
                insert(root->right, temp);
            }
        } else {
            cout << "Duplicate element not inserted" << endl;
            delete temp;
        }
    }
}

void display(node* root) {
    if (root != nullptr) {
        display(root->left);
        cout << root->data << " ";
        display(root->right);
    }
}

node* search(node* root, int t) {
    if (root == nullptr || root->data == t) {
        return root;
    }

    if (t < root->data) {
        return search(root->left, t);
    } else {
        return search(root->right, t);
    }
}

int longestPath(node* root) {
    if (root == nullptr) {
        return 0;
    }

    int L = longestPath(root->left);
    int R = longestPath(root->right);

    return max(L, R) + 1;
}




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

    do {
        cout << "\nMenu\n";
        cout << "1) Insert  \n2) Display  \n3) Search  \n4) Longest Path  \n5) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int value;
                cout << "Enter the value to insert: ";
                cin >> value;
                node* newNode = createNode(value);
                insert(root, newNode);
                break;
            }
            case 2: {
                cout << "BST Elements: ";
                display(root);
                cout << endl;
                break;
            }
            case 3: {
                int searchValue;
                cout << "Enter the value to search: ";
                cin >> searchValue;
                node* searchResult = search(root, searchValue);
                if (searchResult != nullptr) {
                    cout << "Value " << searchValue << " found in the tree." << endl;
                } else {
                    cout << "Value " << searchValue << " not found in the tree." <<
endl;
                }
                break;
            }
            case 4: {
                int pathLength = longestPath(root);
                cout << "Number of nodes in the longest path: " << pathLength <<
endl;
                break;
            }

            case 5: {
                cout << "Exiting the program." << endl;
                break;
            }
            default:
                cout << "Invalid choice. Please enter a valid option." << endl;
        }
    } while (choice != 5);

    return 0;
```

```
}
```