1- Write a C++ program to create a binary search tree (BST) based on user input scores and then print only the even numbers present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم قم بطباعة الأرقام الزوجية الموجودة في شجرة البحث الثنائية فقط.

Input & Output

```
1) Add score
2) Print even numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Print even numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Print even numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Print even numbers
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Print even numbers
3) Exit
Enter your choice: 2
Even numbers in the BST: 2 4
1) Add score
2) Print even numbers
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to print only the even numbers in the BST
void printEvenNumbers(Node* root) {
    if (root == nullptr)
        return;
    if (root->score % 2 == 0)
        cout << root->score << " ";
    printEvenNumbers(root->left);
    printEvenNumbers(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print even numbers\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                cout << "Even numbers in the BST: ";
                printEvenNumbers(root);
                cout << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

2- Write a C++ program to create a binary search tree (BST) based on user input scores, increase all the numbers by one, and then print all the numbers present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، وقم بزيادة جميع الأرقام بمقدار واحد، ثم قم بطباعة جميع الأرقام الموجودة في BST.

Input & Output

```
1) Add score
2) Increase all numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Increase all numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Increase all numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Increase all numbers by one and print
3) Exit
Enter your choice: 2
Numbers after increasing by one: 2 3 4
1) Add score
2) Increase all numbers by one and print
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to increase all numbers by one in the BST
void increaseByOne(Node* root) {
    if (root == nullptr)
        return;
    root->score += 1;
    increaseByOne(root->left);
    increaseByOne(root->right);
}

// Function to print all numbers in the BST
void printNumbers(Node* root) {
    if (root == nullptr)
        return;
    cout << root->score << " ";
    printNumbers(root->left);
    printNumbers(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Increase all numbers by one and print\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                increaseByOne(root);
                cout << "Numbers after increasing by one: ";
                printNumbers(root);
                cout << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

3- Write a C++ program to create a binary search tree (BST) based on user input scores, increase only the even numbers by one, and then print all the numbers present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، وقم بزيادة الأرقام الزوجية بمقدار واحد فقط، ثم اطبع جميع الأرقام الموجودة في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Increase even numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Increase even numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Increase even numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Increase even numbers by one and print
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Increase even numbers by one and print
3) Exit
Enter your choice: 2
Numbers after increasing even numbers by one: 1 3 3 5
1) Add score
2) Increase even numbers by one and print
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to increase even numbers by one in the BST
void increaseEvenByOne(Node* root) {
    if (root == nullptr)
        return;
    if (root->score % 2 == 0)
        root->score += 1;
    increaseEvenByOne(root->left);
    increaseEvenByOne(root->right);
}

// Function to print all numbers in the BST
void printNumbers(Node* root) {
    if (root == nullptr)
        return;
    cout << root->score << " ";
    printNumbers(root->left);
    printNumbers(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Increase even numbers by one and print\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                increaseEvenByOne(root);
                cout << "Numbers after increasing even numbers by one: ";
                printNumbers(root);
                cout << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

4- Write a C++ program to create a binary search tree (BST) based on user input scores, then print all the numbers present in the BST from smallest to largest.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات مدخلات المستخدم، ثم اطبع جميع الأرقام الموجودة في شجرة البحث الثنائية من الأصغر إلى الأكبر.

Input & Output

```
1) Add score
2) Print scores in ascending order
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Print scores in ascending order
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Print scores in ascending order
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Print scores in ascending order
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Print scores in ascending order
3) Exit
Enter your choice: 2
Scores in ascending order: 80 85 90 95
1) Add score
2) Print scores in ascending order
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to print numbers in BST in ascending order
void printAscending(Node* root) {
    if (root == nullptr)
        return;
    printAscending(root->left);
    cout << root->score << " ";
    printAscending(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print scores in ascending order\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                cout << "Scores in ascending order: ";
                printAscending(root);
                cout << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

5- Write a C++ program to create a binary search tree (BST) based on user input scores, then print all the numbers present in the BST from largest to smallest.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم، ثم اطبع جميع الأرقام الموجودة في شجرة البحث الثنائية من الأكبر إلى الأصغر.

Input & Output

```
1) Add score
2) Print scores in descending order
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Print scores in descending order
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Print scores in descending order
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Print scores in descending order
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Print scores in descending order
3) Exit
Enter your choice: 2
Scores in descending order: 95 90 85 80
1) Add score
2) Print scores in descending order
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to print numbers in BST in descending order
void printDescending(Node* root) {
    if (root == nullptr)
        return;
    printDescending(root->right);
    cout << root->score << " ";
    printDescending(root->left);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print scores in descending order\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                cout << "Scores in descending order: ";
                printDescending(root);
                cout << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

6- Write a C++ program to create a binary search tree (BST) based on user input scores, then search for a specific score in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم، ثم ابحث عن نتيجة محددة في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Search for a score
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Search for a score
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Search for a score
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Search for a score
3) Exit
Enter your choice: 1
Enter score: 7
1) Add score
2) Search for a score
3) Exit
Enter your choice: 2
Enter the score to search for: 3
Score 3 found in the BST.
1) Add score
2) Search for a score
3) Exit
Enter your choice: 2
Enter the score to search for: 11
Score 11 not found in the BST.
1) Add score
2) Search for a score
3) Exit
Enter your choice: 3
Exiting the program...
```


Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to search for a score in the binary search tree (BST)
bool search(Node* root, int target) {
    if (root == nullptr)
        return false;
    if (root->score == target)
        return true;
    if (target < root->score)
        return search(root->left, target);
    return search(root->right, target);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Search for a score\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                int target;
                cout << "Enter the score to search for: ";
                cin >> target;
                if (search(root, target))
                    cout << "Score " << target << " found in the BST." << endl;
                else
                    cout << "Score " << target << " not found in the BST." << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

7- Write a C++ program to create a binary search tree (BST) based on user input scores, then print all the leaf nodes in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم اطبع جميع العقد الطرفية في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Print leaf nodes
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Print leaf nodes
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Print leaf nodes
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Print leaf nodes
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Print leaf nodes
3) Exit
Enter your choice: 2
Leaf nodes: 85 95
1) Add score
2) Print leaf nodes
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to print all leaf nodes of the BST
void printLeafNodes(Node* root) {
    if (root == nullptr)
        return;
    if (root->left == nullptr && root->right == nullptr)
        cout << root->score << " ";
    printLeafNodes(root->left);
    printLeafNodes(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print leaf nodes\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                cout << "Leaf nodes: ";
                printLeafNodes(root);
                cout << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

8- Write a C++ program to create a binary search tree (BST) based on user input scores, then calculate and print the sum of all leaf nodes in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم قم بحساب وطباعة مجموع جميع العقد الطرفية في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Calculate sum of leaf nodes
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Calculate sum of leaf nodes
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Calculate sum of leaf nodes
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Calculate sum of leaf nodes
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Calculate sum of leaf nodes
3) Exit
Enter your choice: 2
Sum of leaf nodes: 180
1) Add score
2) Calculate sum of leaf nodes
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(int s) {
        score = s;
        left = nullptr;
        right = nullptr;
    }
};

// Function to add a score to the binary search tree (BST)
Node* add(Node* root, int score) {
    if (root == nullptr) {
        root = new Node(score);
        return root;
    }
    if (score < root->score)
        root->left = add(root->left, score);
    else if (score > root->score)
        root->right = add(root->right, score);

    return root;
}

// Function to calculate the sum of leaf nodes in BST
int sumOfLeafNodes(Node* root) {
    if (root == nullptr)
        return 0;
    if (root->left == nullptr && root->right == nullptr)
        return root->score;
    return sumOfLeafNodes(root->left) + sumOfLeafNodes(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Calculate sum of leaf nodes\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                int score;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, score);
                break;
            case 2:
                if (root == nullptr) {
                    cout << "BST is empty. Please add scores first." << endl;
                    break;
                }
                cout << "Sum of leaf nodes: " << sumOfLeafNodes(root) << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

9- Write a C++ program to implement operations on a binary search tree (BST). The program should allow adding nodes, showing the tree, and deleting nodes based on the name of the node. The nodes of the tree contain a name and a score.

كتابة برنامج C++ لتنفيذ العمليات على شجرة البحث الثنائية (BST). يجب أن يسمح البرنامج بإضافة العقد وإظهار الشجرة وحذف العقد بناءً على اسم العقدة. تحتوي عقد الشجرة على اسم ودرجة.

Input & Output

Solution

```cpp
// www.gammal.tech

#include <iostream>
#include <string>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    string name;
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(string n, int s) {
        name = n;
        score = s;
        left = nullptr;
        right = nullptr;
    }

    // Function to print node details
    void print() {
        cout << name << " " << score << endl;
    }
};

// Function to add a node to the BST
Node* add(Node* root, string n, int s) {
    if (root == nullptr) {
        root = new Node(n, s);
        return root;
    }
    if (s < root->score)
        root->left = add(root->left, n, s);
    if (s > root->score)
        root->right = add(root->right, n, s);
    return root;
}

// Function to show the tree
void show(Node* root) {
    if (root == nullptr)
        return;
    root->print();
    show(root->left);
    show(root->right);
}

// Function to delete a node from the BST
Node* del(Node* root) {
    Node* r = root->right;
    Node* L = root->left;
    if (r != nullptr) {
        Node* prev = r;
        while (r->left != nullptr) {
            prev = r;
            r = r->left;
        }
        if (prev != r)
            prev->left = r->right;
        r->left = root->left;
        if (root->right != r)
            r->right = root->right;
        delete (root);
        return r;
    }
    if (L != nullptr) {
        Node* prev = L;
        while (L->right != nullptr) {
            prev = L;
            L = L->right;
        }
        if (prev != L)
            prev->right = L->left;
        L->right = root->right;
        if (root->left != L)
            L->left = root->left;
        delete (root);
        return L;
    }
    delete (root);
    return nullptr;
}
```

```cpp
// Function to delete a node by name from the BST
Node* del_name(Node* root, string n) {
    if (root == nullptr)
        return nullptr;
    if (root->name == n)
        return del(root);
    root->right = del_name(root->right, n);
    root->left = del_name(root->left, n);
    return root;
}

int main() {
    Node* root = nullptr;
    int answer;
    do {
        cout << "1) Add\n";
        cout << "2) Show\n";
        cout << "3) Delete\n";
        cout << "4) Exit\n";
        cout << "What would you like to do? ";
        cin >> answer;
        if (answer == 1) {
            string n;
            int s;
            cout << "Name: ";
            cin >> n;
            cout << "Score: ";
            cin >> s;
            root = add(root, n, s);
        } else if (answer == 2) {
            show(root);
        } else if (answer == 3) {
            string n;
            cout << "Name: ";
            cin >> n;
            root = del_name(root, n);
        }
    } while (answer >= 1 && answer <= 3);
    return 0;
}
```

10- Write a C++ program to implement a binary search tree (BST) with the ability to add nodes and display the tree. The program should provide options for the user to add nodes to the BST and display the BST in ascending or descending order based on node values.

كتابة برنامج C++ لتنفيذ شجرة بحث ثنائية (BST) مع إمكانية إضافة العقد وعرض الشجرة. يجب أن يوفر البرنامج خيارات للمستخدم لإضافة العقد إلى BST وعرض BST بترتيب تصاعدي أو تنازلي بناءً على قيم العقد.

Input & Output

```
1) Add
2) Display Ascending
3) Display Descending
4) Exit
Choose an option: 1
Enter name: ahmed
Enter score: 20
1) Add
2) Display Ascending
3) Display Descending
4) Exit
Choose an option: 1
Enter name: aly
Enter score: 25
1) Add
2) Display Ascending
3) Display Descending
4) Exit
Choose an option: 1
Enter name: amr
Enter score: 15
1) Add
2) Display Ascending
3) Display Descending
4) Exit
Choose an option: 2
BST in ascending order:
amr 15
ahmed   20
aly 25
1) Add
2) Display Ascending
3) Display Descending
4) Exit
Choose an option: 3
BST in descending order:
aly 25
ahmed   20
amr 15
1) Add
2) Display Ascending
3) Display Descending
4) Exit
Choose an option: 4
Exiting the program.
```

Solution

```cpp
// www.gammal.tech

#include <iostream>
#include <string>
using namespace std;

// Structure definition for binary search tree (BST) node
struct Node {
    string name;
    int score;
    Node* left;
    Node* right;

    // Constructor
    Node(string n, int s) {
        name = n;
        score = s;
        left = nullptr;
        right = nullptr;
    }

    // Function to print node data
    void print() {
        cout << name << "\t" << score << endl;
    }
};

// Function to add a node to the BST
Node* add(Node* root, string name, int score) {
    if (root == nullptr) {
        root = new Node(name, score);
        return root;
    }

    if (score < root->score)
        root->left = add(root->left, name, score);
    else
        root->right = add(root->right, name, score);

    return root;
}

// Function to display the BST in ascending order
void displayAscending(Node* root) {
    if (root == nullptr) return;

    displayAscending(root->left);
    root->print();
    displayAscending(root->right);
}

// Function to display the BST in descending order
void displayDescending(Node* root) {
    if (root == nullptr) return;

    displayDescending(root->right);
    root->print();
    displayDescending(root->left);
}

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

    do {
        cout << "1) Add\n";
        cout << "2) Display Ascending\n";
        cout << "3) Display Descending\n";
        cout << "4) Exit\n";
        cout << "Choose an option: ";
        cin >> answer;

        switch (answer) {
            case 1: {
                string name;
                int score;
                cout << "Enter name: ";
                cin >> name;
                cout << "Enter score: ";
                cin >> score;
                root = add(root, name, score);
                break;
            }
            case 2: {
                cout << "BST in ascending order:\n";
                displayAscending(root);
                break;
            }
            case 3: {
                cout << "BST in descending order:\n";
                displayDescending(root);
                break;
            }
            case 4: {
                cout << "Exiting the program.\n";
                break;
            }
            default:
                cout << "Invalid option. Please choose again.\n";
        }
    } while (answer != 4);

    return 0;
}
```