1- Write a program to manage scores using a Binary Search Tree (BST). The program should allow users to add scores along with names and display them in preorder traversal. Each score along with its corresponding name should be represented as a node in the BST.

اكتب برنامجًا لإدارة النتائج باستخدام شجرة البحث الثنائية (BST). يجب أن يسمح البرنامج للمستخدمين بإضافة النتائج مع الأسماء وعرضها في اجتياز الطلب المسبق. يجب أن يتم تمثيل كل درجة مع اسمها المقابل كعقدة في BST.

Input & Output

```
1) Add
2) Show
3) Exit
Enter your choice: 1
Name: aly
Score: 90
1) Add
2) Show
3) Exit
Enter your choice: 1
Name: ahmed
Score: 95
1) Add
2) Show
3) Exit
Enter your choice: 1
Name: amr
Score: 80
1) Add
2) Show
3) Exit
Enter your choice: 2
Scores in preorder traversal:
aly 90
amr 80
ahmed   95
1) Add
2) Show
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

```cpp
// www.gammal.tech

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

struct Node {
    string name;
    int score;
    Node* left;
    Node* right;

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

    void print() {
        cout << name << "\t" << score << endl;
    }
};

Node* add(Node* root, string name, int score) {
    if (root == NULL) {
        root = new Node(name, score);
        return root;
    }

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

    return root;
}

void show(Node* root) {
    if (root == NULL)
        return;
    root->print();
    show(root->left);
    show(root->right);
}

int main() {
    Node* root = NULL;
    int answer;
    do {
        cout << "1) Add\n2) Show\n3) Exit\n";
        cout << "Enter your choice: ";
        cin >> answer;
        switch (answer) {
            case 1: {
                string name;
                int score;
                cout << "Name: ";
                cin >> name;
                cout << "Score: ";
                cin >> score;
                root = add(root, name, score);
                break;
            }
            case 2: {
                cout << "Scores in preorder traversal:" << endl;
                show(root);
                break;
            }
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (answer != 3);
    return 0;
}
```

2- Write a C++ program to construct a binary search tree (BST) based on user input scores and then find the average of all numbers in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم ابحث عن متوسط جميع الأرقام في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Average of Numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Average of Numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Average of Numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Average of Numbers
3) Exit
Enter your choice: 2
Average of all numbers in the BST: 2
1) Add score
2) Average of 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 calculate the sum of all numbers in the BST
int sumOfNumbers(Node* root) {
    if (root == nullptr)
        return 0;
    return root->score + sumOfNumbers(root->left) + sumOfNumbers(root->right);
}

// Function to count the number of nodes in the BST
int countNodes(Node* root) {
    if (root == nullptr)
        return 0;
    return 1 + countNodes(root->left) + countNodes(root->right);
}

// Function to calculate the average of all numbers in the BST
double averageOfNumbers(Node* root) {
    int sum = sumOfNumbers(root);
    int count = countNodes(root);
    if (count == 0)
        return 0;
    return static_cast<double>(sum) / count;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Average of 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 << "Average of all numbers in the BST: " << averageOfNumbers(root) << 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 construct a binary search tree (BST) based on user input scores and then check whether the number 5 is found in the BST or not.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم ثم تحقق مما إذا كان الرقم 5 موجودًا في شجرة البحث الثنائية أم لا.

Input & Output

```
1) Add score
2) Search for number 5
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Search for number 5
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Search for number 5
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Search for number 5
3) Exit
Enter your choice: 2
Number 5 found in the BST.
1) Add score
2) Search for number 5
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 number in the BST
bool search(Node* root, int key) {
    if (root == nullptr)
        return false;
    if (root->score == key)
        return true;
    if (key < root->score)
        return search(root->left, key);
    else
        return search(root->right, key);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Search for number 5\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 (search(root, 5))
                    cout << "Number 5 found in the BST." << endl;
                else
                    cout << "Number 5 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;
}
```

4- Write a C++ program to construct a binary search tree (BST) based on user input scores and then print all numbers between 5 and 10 in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم، ثم اطبع جميع الأرقام بين 5 و10 في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Print numbers between 5 and 10
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Print numbers between 5 and 10
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Print numbers between 5 and 10
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Print numbers between 5 and 10
3) Exit
Enter your choice: 1
Enter score: 6
1) Add score
2) Print numbers between 5 and 10
3) Exit
Enter your choice: 1
Enter score: 11
1) Add score
2) Print numbers between 5 and 10
3) Exit
Enter your choice: 2
Numbers between 5 and 10 in the BST: 5 6
1) Add score
2) Print numbers between 5 and 10
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 numbers between low and high in the BST
void printInRange(Node* root, int low, int high) {
    if (root == nullptr)
        return;
    if (root->score > low)
        printInRange(root->left, low, high);
    if (root->score >= low && root->score <= high)
        cout << root->score << " ";
    if (root->score < high)
        printInRange(root->right, low, high);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print numbers between 5 and 10\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 << "Numbers between 5 and 10 in the BST: ";
                printInRange(root, 5, 10);
                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 construct a binary search tree (BST) based on user input scores and then print the sum of all even numbers in the BST.

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

Input & Output

```
1) Add score
2) Print sum of even numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Print sum of even numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Print sum of even numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Print sum of even numbers
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Print sum of even numbers
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Print sum of even numbers
3) Exit
Enter your choice: 2
Sum of even numbers in the BST: 6
1) Add score
2) Print sum of 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 calculate the sum of all even numbers in the BST
int sumEvenNumbers(Node* root) {
    if (root == nullptr)
        return 0;
    int sum = 0;
    if (root->score % 2 == 0)
        sum += root->score;
    sum += sumEvenNumbers(root->left);
    sum += sumEvenNumbers(root->right);
    return sum;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print sum of 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 << "Sum of even numbers in the BST: " << sumEvenNumbers(root) << 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 construct a binary search tree (BST) based on user input scores and then print the sum of all odd numbers in the BST.

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

Input & Output

```
1) Add score
2) Print sum of odd numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Print sum of odd numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Print sum of odd numbers
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Print sum of odd numbers
3) Exit
Enter your choice: 2
Sum of odd numbers in the BST: 8
1) Add score
2) Print sum of odd 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 calculate the sum of all odd numbers in the BST
int sumOddNumbers(Node* root) {
    if (root == nullptr)
        return 0;
    int sum = 0;
    if (root->score % 2 != 0)
        sum += root->score;
    sum += sumOddNumbers(root->left);
    sum += sumOddNumbers(root->right);
    return sum;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print sum of odd 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 << "Sum of odd numbers in the BST: " << sumOddNumbers(root) << 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 construct a binary search tree (BST) based on user input scores and then print all numbers smaller than 10 present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم اطبع جميع الأرقام الأصغر من 10 الموجودة في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Print numbers smaller than 10
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Print numbers smaller than 10
3) Exit
Enter your choice: 1
Enter score: 12
1) Add score
2) Print numbers smaller than 10
3) Exit
Enter your choice: 1
Enter score: 13
1) Add score
2) Print numbers smaller than 10
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Print numbers smaller than 10
3) Exit
Enter your choice: 2
Numbers smaller than 10 in the BST: 1 5
1) Add score
2) Print numbers smaller than 10
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 numbers smaller than a given value in the BST
void printNumbersSmallerThan(Node* root, int target) {
    if (root == nullptr)
        return;
    if (root->score < target)
        cout << root->score << " ";
    printNumbersSmallerThan(root->left, target);
    printNumbersSmallerThan(root->right, target);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print numbers smaller than 10\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 << "Numbers smaller than 10 in the BST: ";
                printNumbersSmallerThan(root, 10);
                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 construct a binary search tree (BST) based on user input scores and then print all numbers larger than 10 present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم اطبع جميع الأرقام الأكبر من 10 الموجودة في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Print numbers larger than 10
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Print numbers larger than 10
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Print numbers larger than 10
3) Exit
Enter your choice: 1
Enter score: 15
1) Add score
2) Print numbers larger than 10
3) Exit
Enter your choice: 1
Enter score: 20
1) Add score
2) Print numbers larger than 10
3) Exit
Enter your choice: 1
Enter score: 10
1) Add score
2) Print numbers larger than 10
3) Exit
Enter your choice: 2
Numbers larger than 10 in the BST: 15 20
1) Add score
2) Print numbers larger than 10
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 numbers larger than a given value in the BST
void printNumbersLargerThan(Node* root, int target) {
    if (root == nullptr)
        return;
    if (root->score > target)
        cout << root->score << " ";
    printNumbersLargerThan(root->left, target);
    printNumbersLargerThan(root->right, target);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print numbers larger than 10\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 << "Numbers larger than 10 in the BST: ";
                printNumbersLargerThan(root, 10);
                cout << 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 construct a binary search tree (BST) based on user input scores and then print the sum of all numbers larger than 20 present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم اطبع مجموع جميع الأرقام الأكبر من 20 الموجودة في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Print sum of numbers larger than 20
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Print sum of numbers larger than 20
3) Exit
Enter your choice: 1
Enter score: 20
1) Add score
2) Print sum of numbers larger than 20
3) Exit
Enter your choice: 1
Enter score: 50
1) Add score
2) Print sum of numbers larger than 20
3) Exit
Enter your choice: 1
Enter score: 40
1) Add score
2) Print sum of numbers larger than 20
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Print sum of numbers larger than 20
3) Exit
Enter your choice: 2
Sum of numbers larger than 20 in the BST: 90
1) Add score
2) Print sum of numbers larger than 20
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 numbers larger than a given value in the BST
int sumNumbersLargerThan(Node* root, int target) {
    if (root == nullptr)
        return 0;
    if (root->score > target)
        return root->score + sumNumbersLargerThan(root->left, target) + sumNumbersLargerThan(root-
>right, target);
    return sumNumbersLargerThan(root->right, target);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print sum of numbers larger than 20\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 << "Sum of numbers larger than 20 in the BST: ";
                cout << sumNumbersLargerThan(root, 20) << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```

10- Write a C++ program to create a binary search tree (BST) based on user input scores and then print the sum of all numbers smaller than 15 present in the BST.

اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم اطبع مجموع جميع الأرقام الأصغر من 15 الموجودة في شجرة البحث الثنائية.

Input & Output

```
1) Add score
2) Print sum of numbers smaller than 15
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Print sum of numbers smaller than 15
3) Exit
Enter your choice: 1
Enter score: 6
1) Add score
2) Print sum of numbers smaller than 15
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Print sum of numbers smaller than 15
3) Exit
Enter your choice: 1
Enter score: 16
1) Add score
2) Print sum of numbers smaller than 15
3) Exit
Enter your choice: 1
Enter score: 17
1) Add score
2) Print sum of numbers smaller than 15
3) Exit
Enter your choice: 2
Sum of numbers smaller than 15 in the BST: 12
1) Add score
2) Print sum of numbers smaller than 15
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 numbers smaller than a given value in the BST
int sumNumbersSmallerThan(Node* root, int target) {
    if (root == nullptr)
        return 0;
    if (root->score < target)
        return root->score + sumNumbersSmallerThan(root->left, target) +
sumNumbersSmallerThan(root->right, target);
    return sumNumbersSmallerThan(root->left, target);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Print sum of numbers smaller than 15\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 << "Sum of numbers smaller than 15 in the BST: ";
                cout << sumNumbersSmallerThan(root, 15) << endl;
                break;
            case 3:
                cout << "Exiting the program..." << endl;
                break;
            default:
                cout << "Invalid choice! Please enter again." << endl;
        }
    } while (choice != 3);
    return 0;
}
```