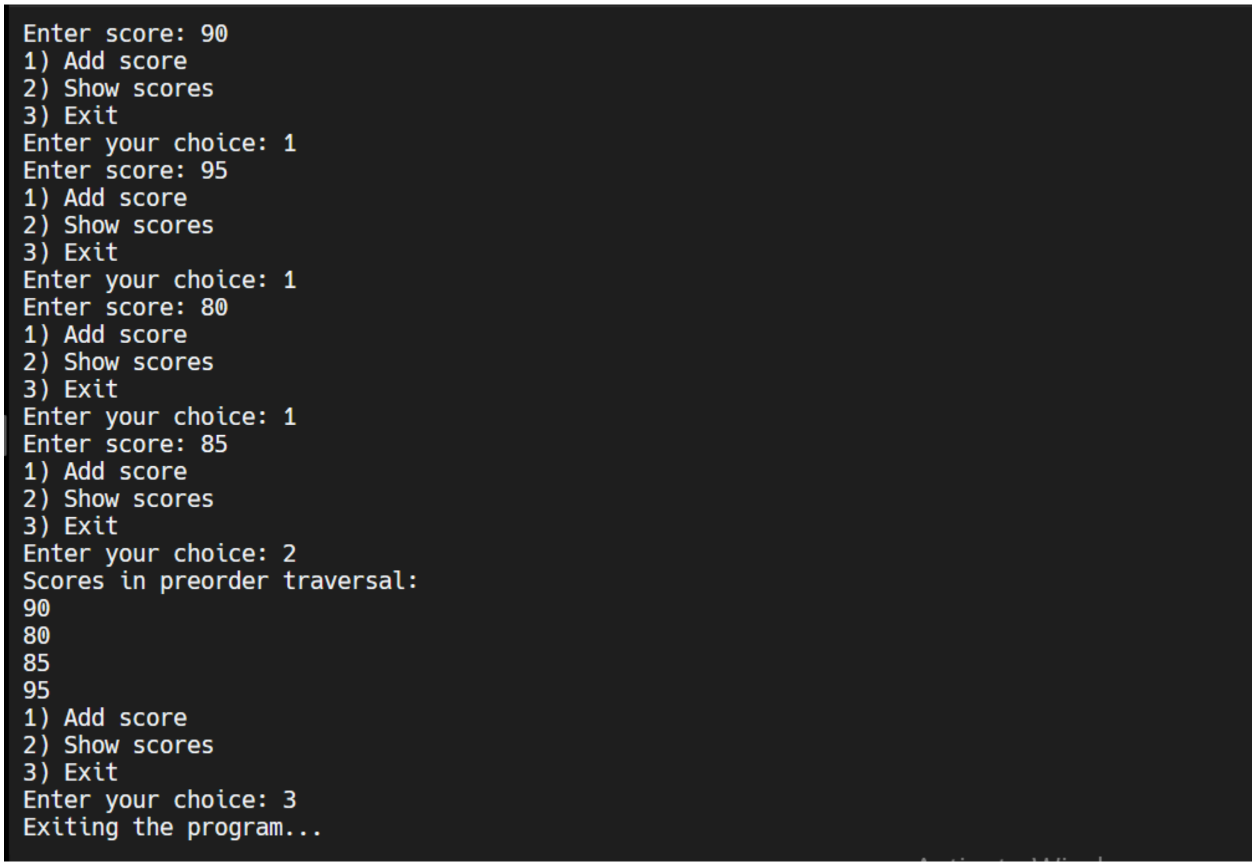
1- Write a C++ program to implement a binary search tree (BST) using structures and dynamic memory allocation. The program should provide options to add scores to the BST and display the scores in preorder traversal. The program should terminate when the user chooses to exit.

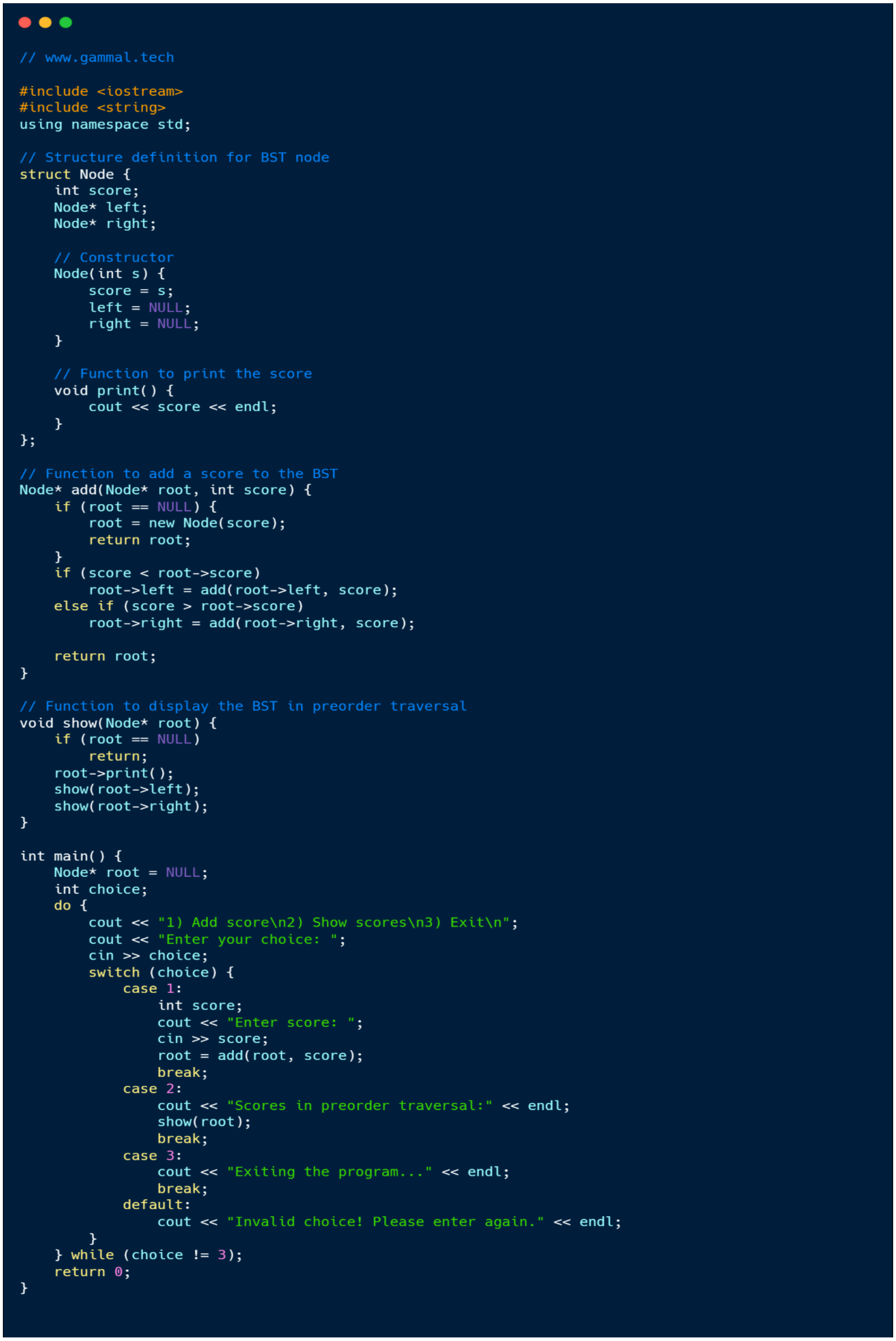
اكتب برنامج C++ لتنفيذ شجرة بحث ثنائية (BST) باستخدام structures وتخصيص الذاكرة الديناميكية. يجب أن يوفر البرنامج خيارات لإضافة الدرجات إلى BST وعرض الدرجات في اجتياز الطلب المسبق. يجب أن ينتهي البرنامج عندما يختار المستخدم الخروج.

Input & Output



```
Enter score: 90
1) Add score
2) Show scores
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Show scores
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Show scores
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Show scores
3) Exit
Enter your choice: 2
Scores in preorder traversal:
90
80
85
95
1) Add score
2) Show scores
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

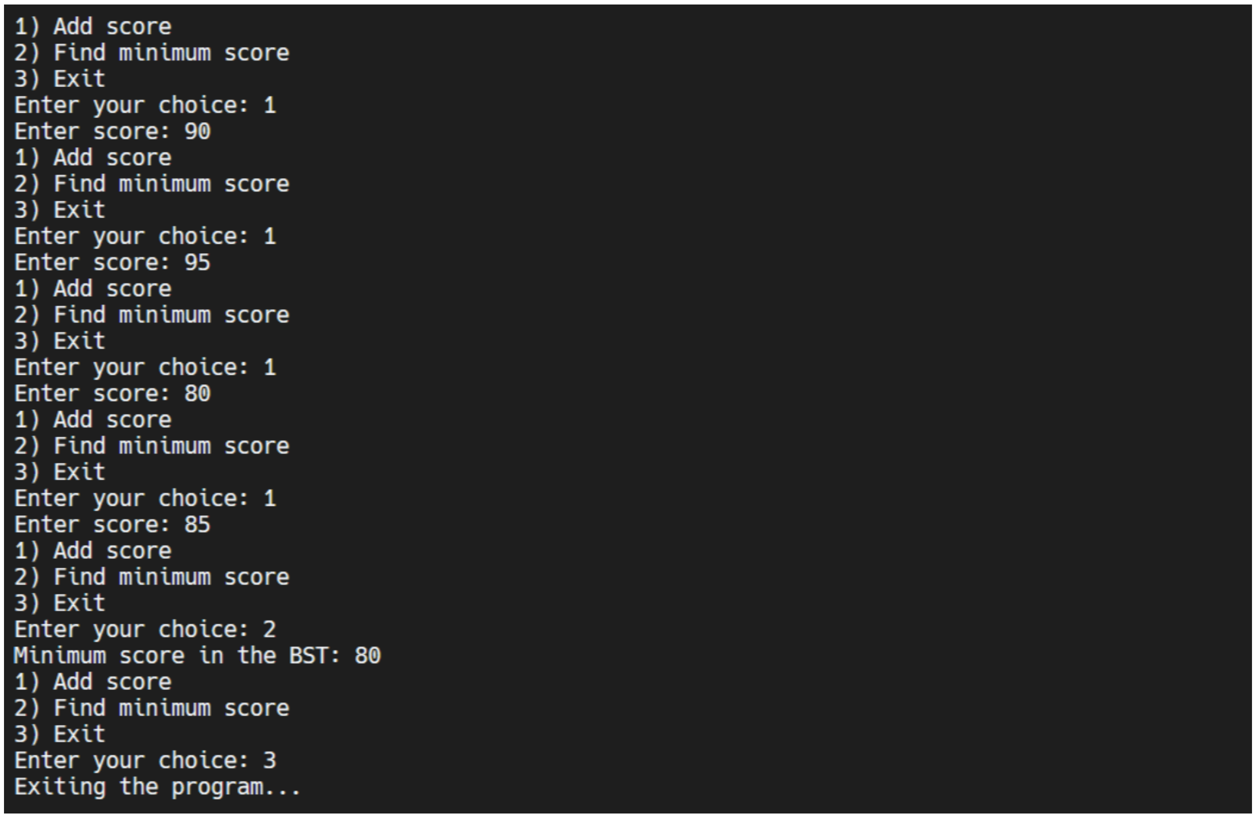


```cpp
// www.gammal.tech

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

// Structure definition for BST node
struct Node {
    int score;
    Node* left;
    Node* right;

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

    // Function to print the score
    void print() {
        cout << score << endl;
    }
};

// Function to add a score to the BST
Node* add(Node* root, int score) {
    if (root == NULL) {
        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 display the BST in preorder traversal
void show(Node* root) {
    if (root == NULL)
        return;
    root->print();
    show(root->left);
    show(root->right);
}

int main() {
    Node* root = NULL;
    int choice;
    do {
        cout << "1) Add score\n2) Show scores\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 << "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 (choice != 3);
    return 0;
}
```

2- Write a C++ program to find the minimum element in a binary search tree (BST). The program should prompt the user to input scores to build the BST. After constructing the BST, it should find and display the minimum score present in the tree.

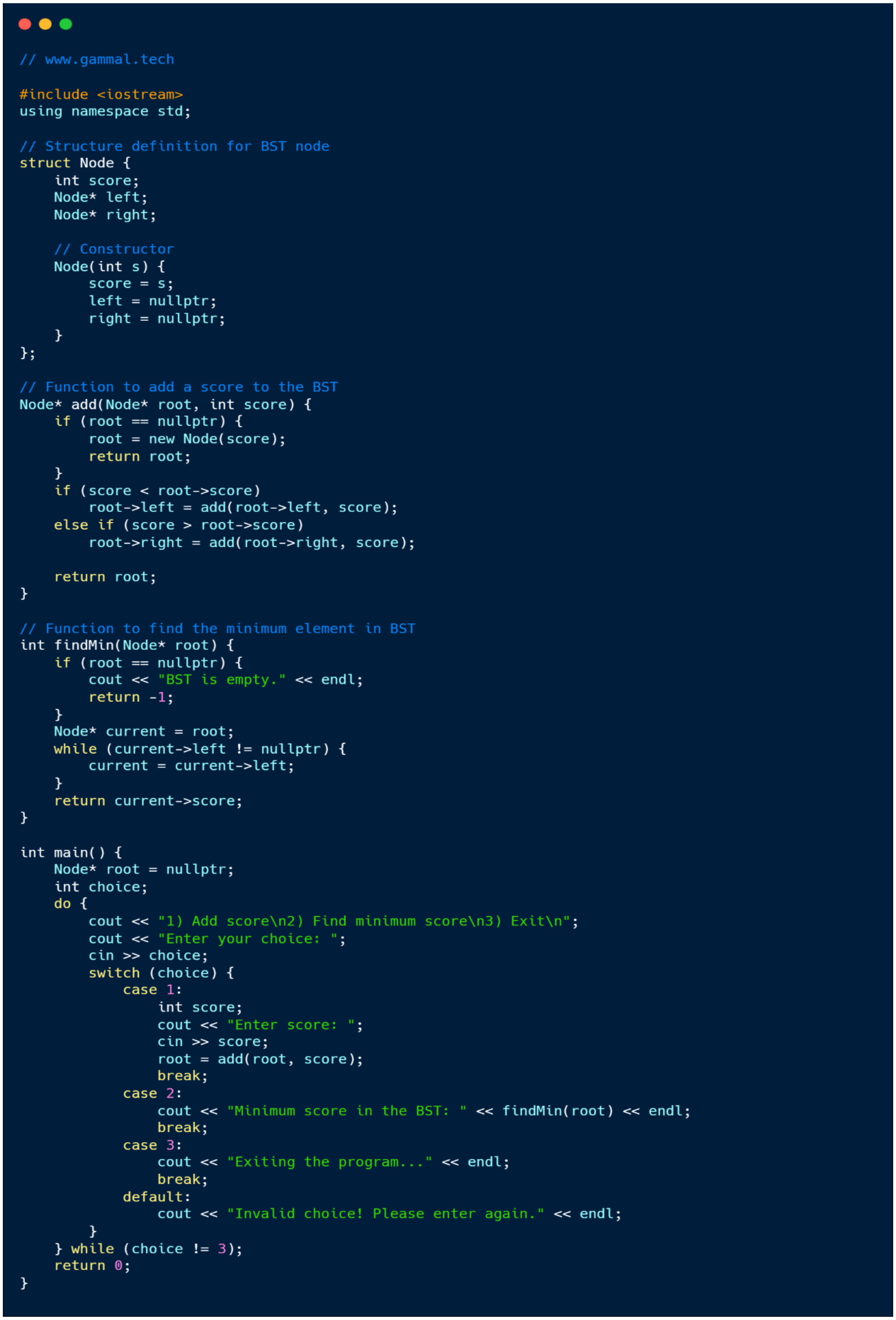
اكتب برنامج C++ للعثور على الحد الأدنى من العناصر في شجرة البحث الثنائية (BST). يجب أن يطالب البرنامج المستخدم بإدخال الدرجات لبناء BST. بعد إنشاء BST، يجب العثور على الحد الأدنى من الدرجات الموجودة في الشجرة وعرضها.

Input & Output



```
1) Add score
2) Find minimum score
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Find minimum score
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Find minimum score
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Find minimum score
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Find minimum score
3) Exit
Enter your choice: 2
Minimum score in the BST: 80
1) Add score
2) Find minimum score
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

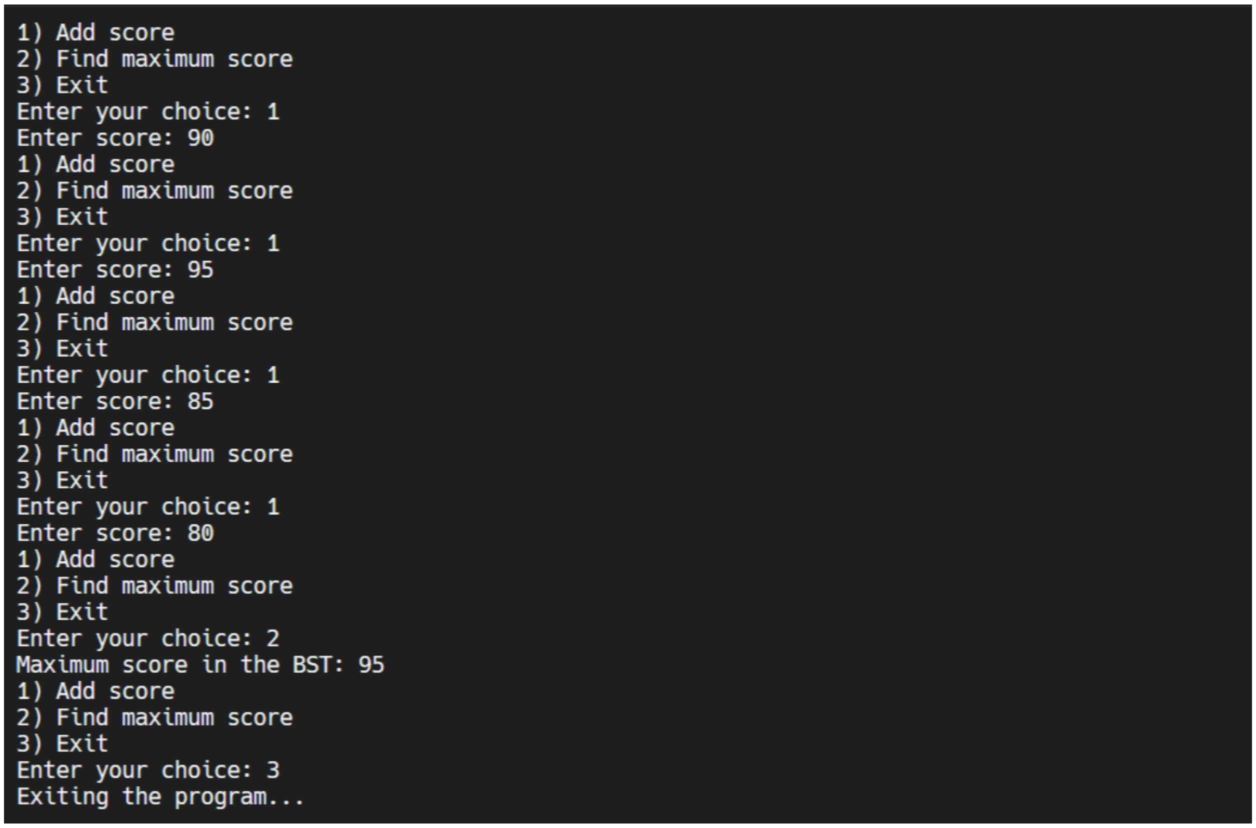


```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for 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 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 find the minimum element in BST
int findMin(Node* root) {
    if (root == nullptr) {
        cout << "BST is empty." << endl;
        return -1;
    }
    Node* current = root;
    while (current->left != nullptr) {
        current = current->left;
    }
    return current->score;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Find minimum 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:
                cout << "Minimum score in the BST: " << findMin(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 find the maximum element in a binary search tree (BST). The program should prompt the user to input scores to build the BST. After constructing the BST, it should find and display the maximum score present in the tree.

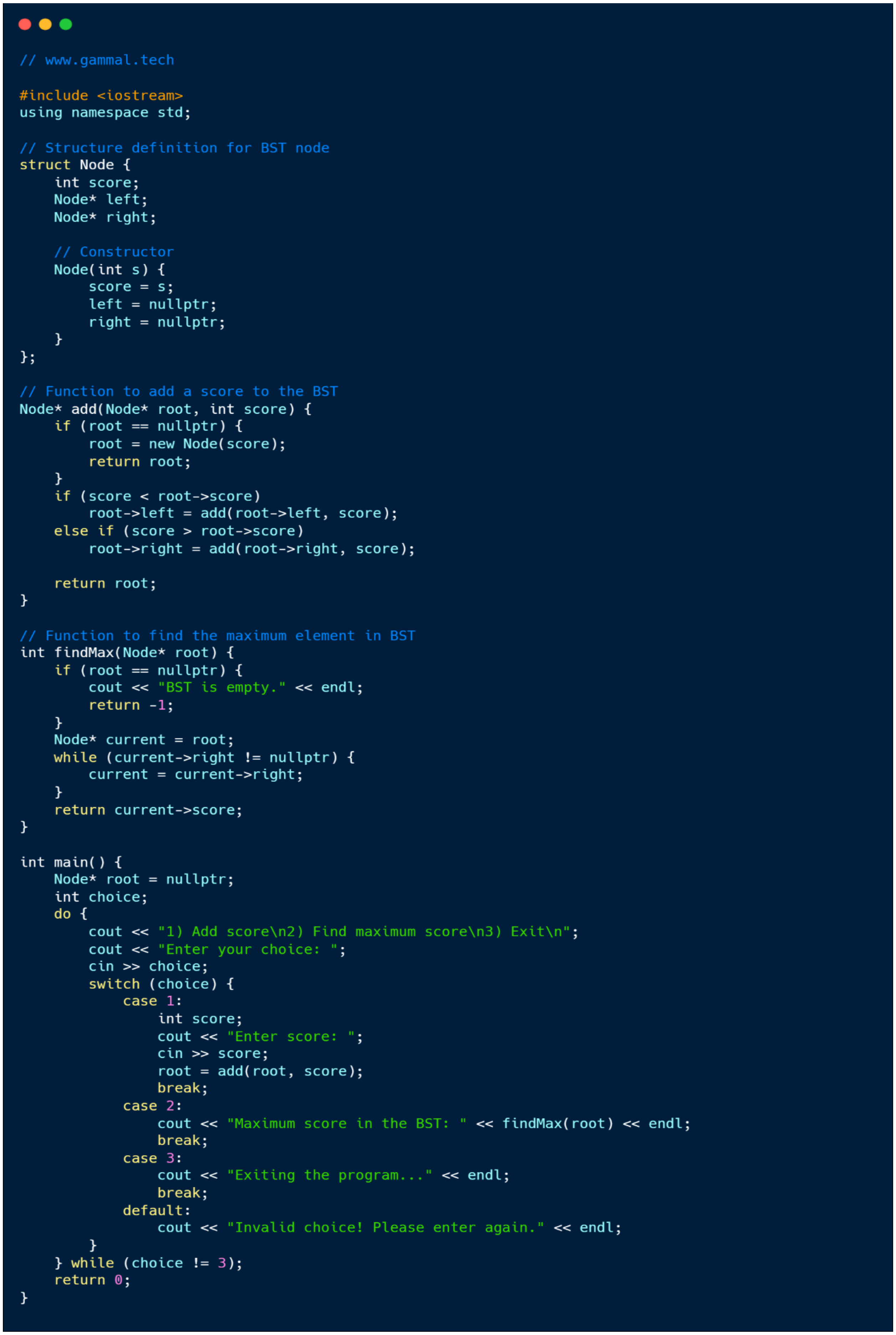
اكتب برنامج C++ للعثور على الحد الأقصى للعنصر في شجرة البحث الثنائية (BST). يجب أن يطالب البرنامج المستخدم بإدخال الدرجات لبناء BST. بعد إنشاء BST، يجب العثور على الحد الأقصى للدرجات الموجودة في الشجرة وعرضها.

Input & Output



```
1) Add score
2) Find maximum score
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Find maximum score
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Find maximum score
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Find maximum score
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Find maximum score
3) Exit
Enter your choice: 2
Maximum score in the BST: 95
1) Add score
2) Find maximum score
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

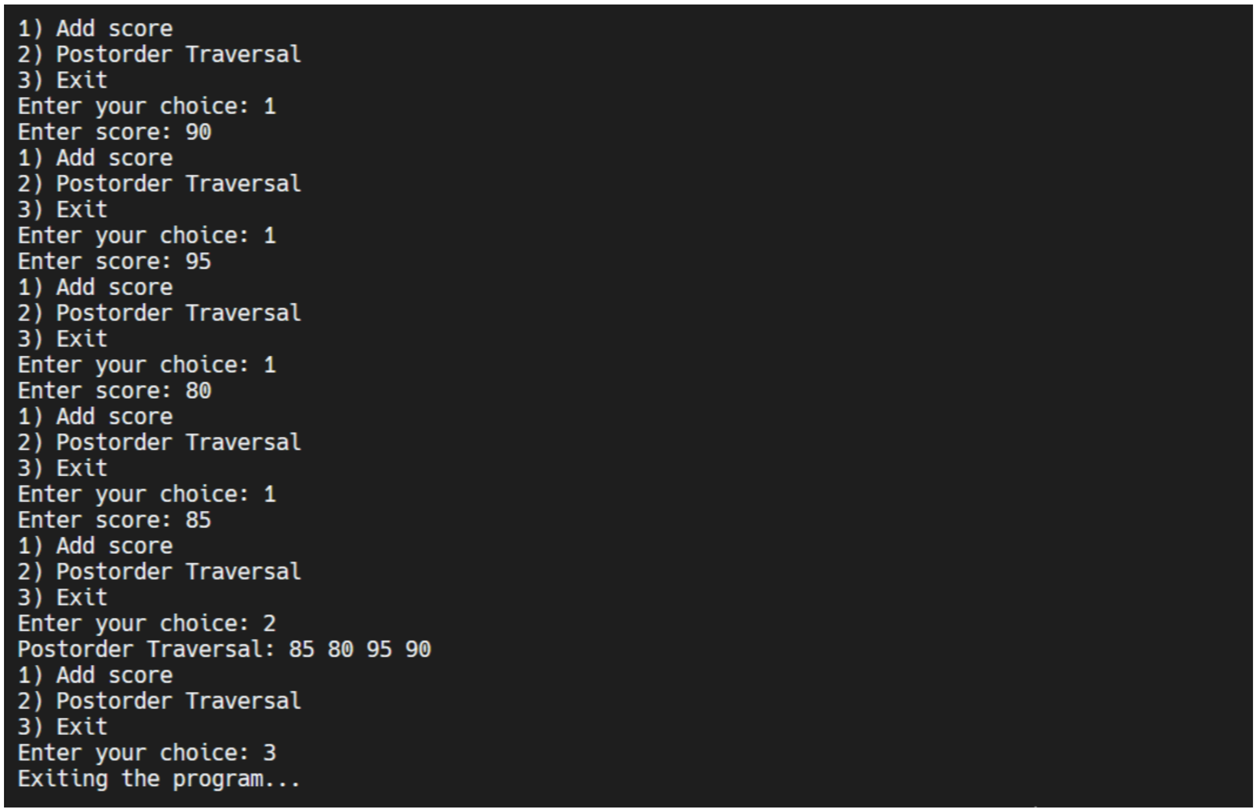


```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for 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 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 find the maximum element in BST
int findMax(Node* root) {
    if (root == nullptr) {
        cout << "BST is empty." << endl;
        return -1;
    }
    Node* current = root;
    while (current->right != nullptr) {
        current = current->right;
    }
    return current->score;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Find maximum 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:
                cout << "Maximum score in the BST: " << findMax(root) << 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 perform postorder traversal to display the elements of the tree in postorder sequence.

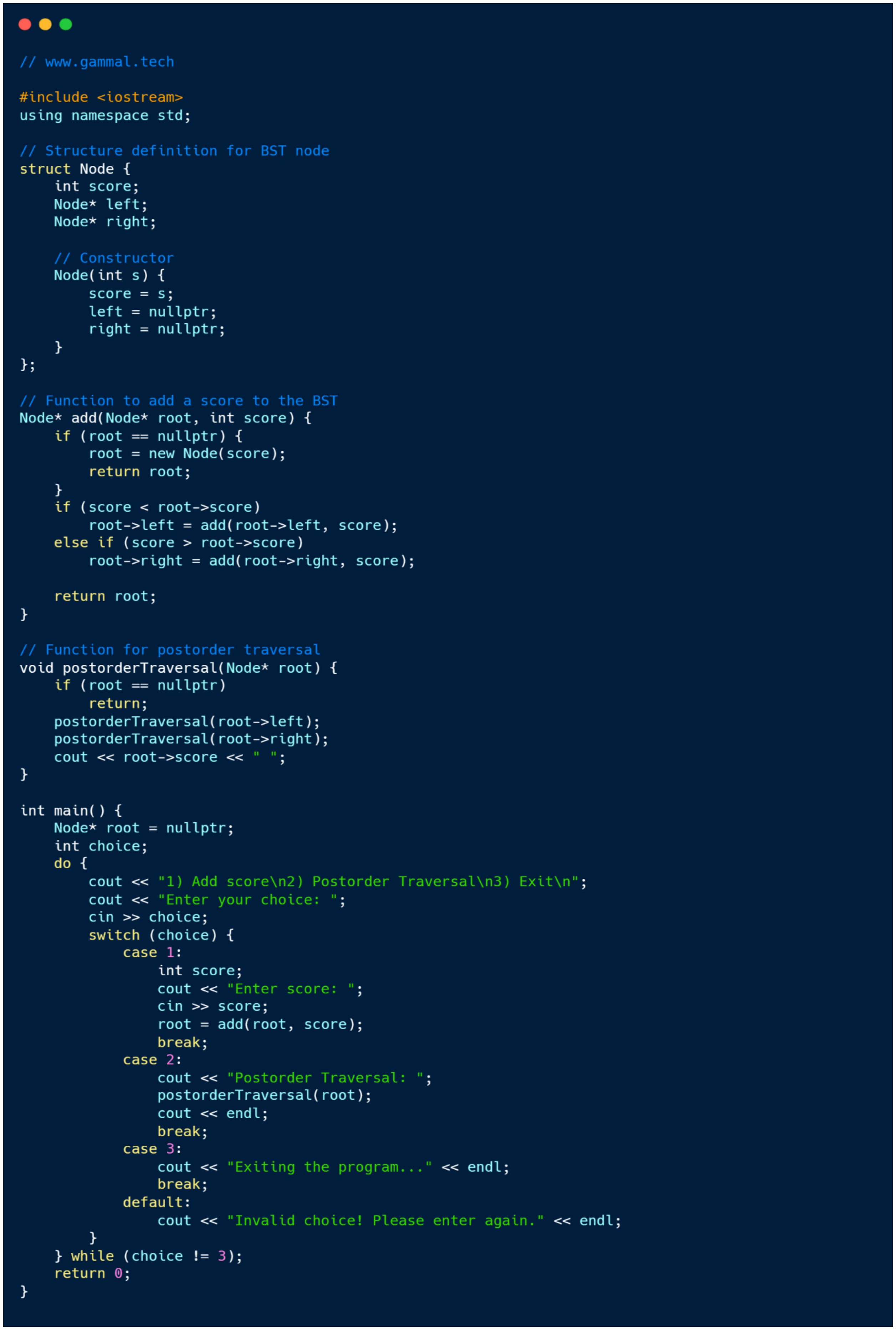
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم ثم قم بإجراء اجتياز الطلب اللاحق لعرض عناصر الشجرة في تسلسل ما بعد الطلب.

Input & Output



```
1) Add score
2) Postorder Traversal
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Postorder Traversal
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Postorder Traversal
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Postorder Traversal
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Postorder Traversal
3) Exit
Enter your choice: 2
Postorder Traversal: 85 80 95 90
1) Add score
2) Postorder Traversal
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

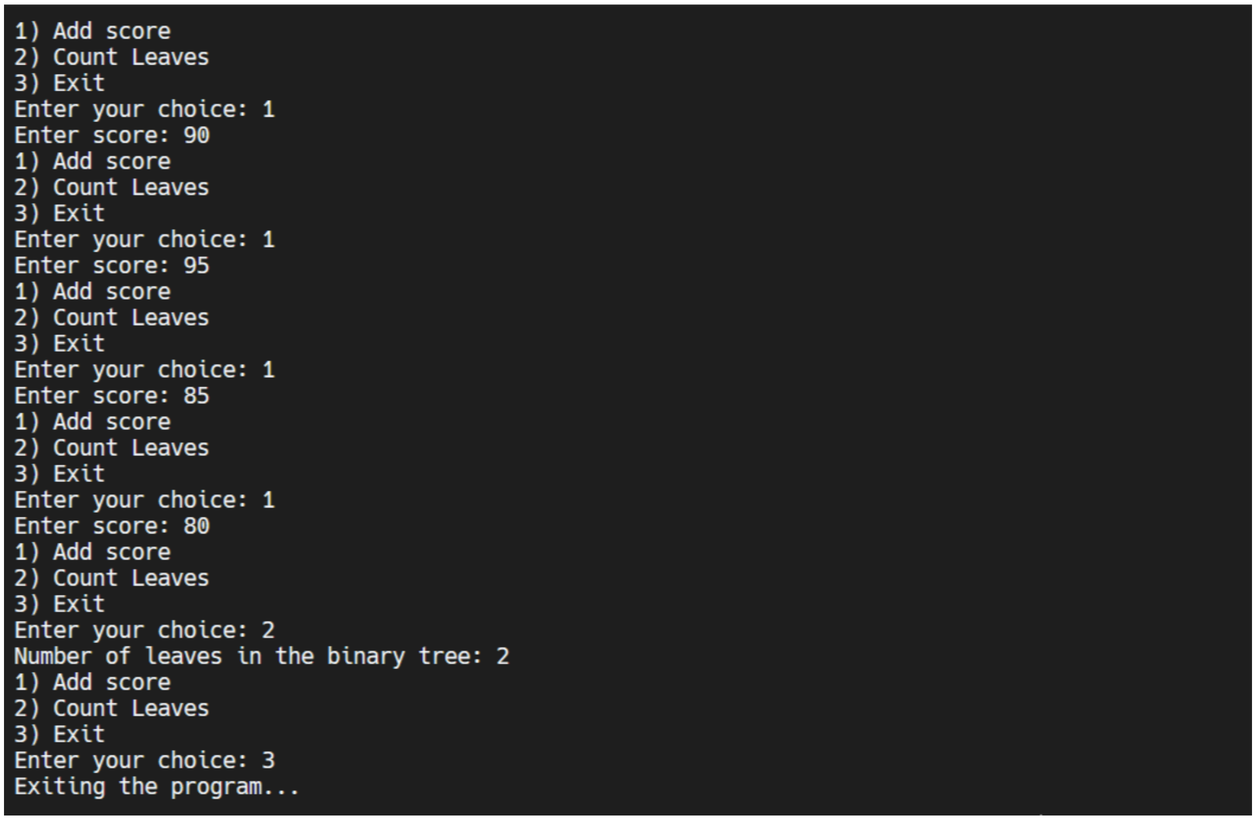


```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for 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 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 for postorder traversal
void postorderTraversal(Node* root) {
    if (root == nullptr)
        return;
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    cout << root->score << " ";
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Postorder Traversal\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 << "Postorder Traversal: ";
                postorderTraversal(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 construct a binary tree based on user input scores and then count the number of leaf nodes in the binary tree.

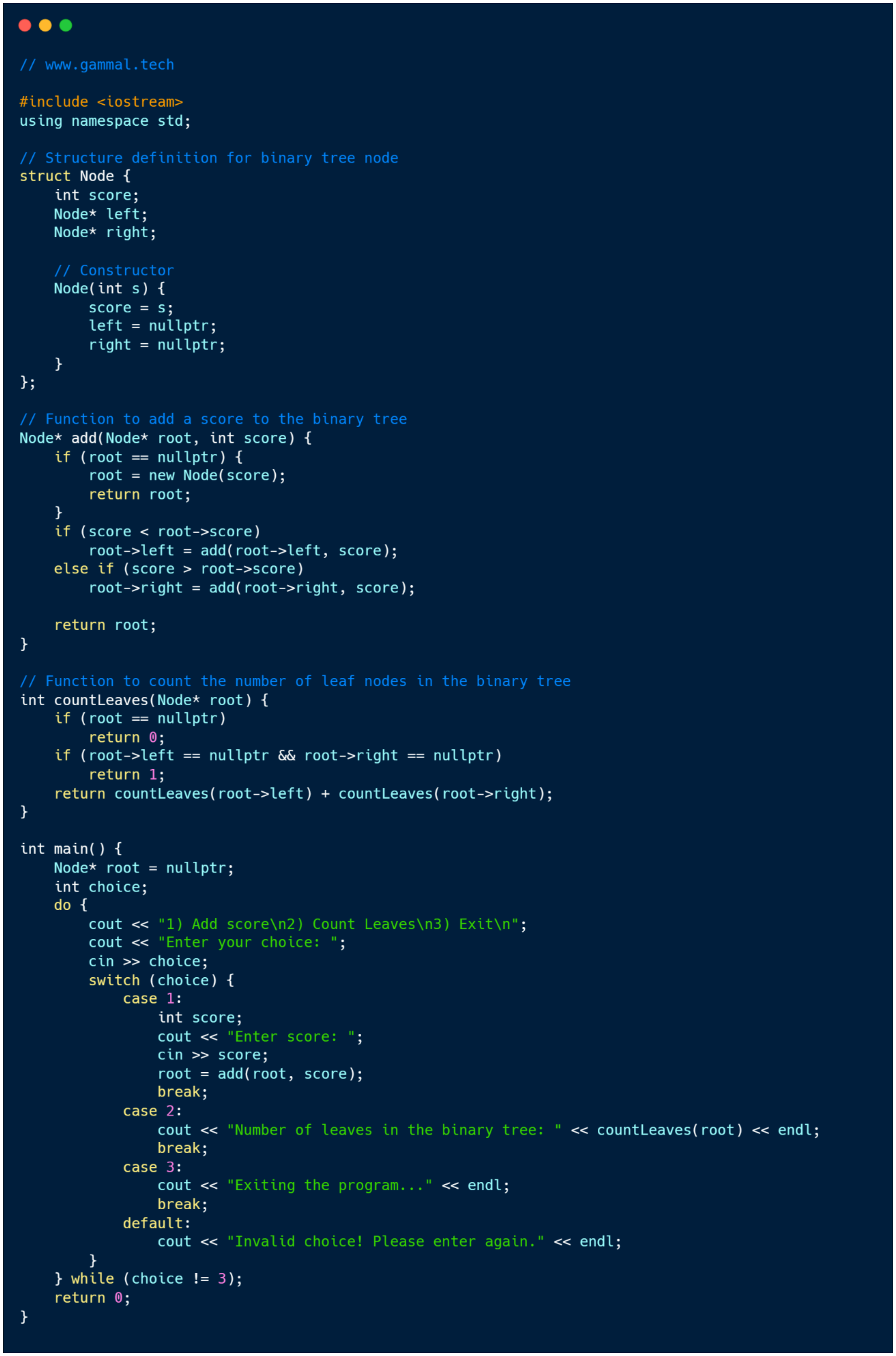
اكتب برنامج C++ لإنشاء شجرة ثنائية بناءً على درجات إدخال المستخدم ثم قم بحساب عدد العقد الورقية في الشجرة الثنائية.

Input & Output



```
1) Add score
2) Count Leaves
3) Exit
Enter your choice: 1
Enter score: 90
1) Add score
2) Count Leaves
3) Exit
Enter your choice: 1
Enter score: 95
1) Add score
2) Count Leaves
3) Exit
Enter your choice: 1
Enter score: 85
1) Add score
2) Count Leaves
3) Exit
Enter your choice: 1
Enter score: 80
1) Add score
2) Count Leaves
3) Exit
Enter your choice: 2
Number of leaves in the binary tree: 2
1) Add score
2) Count Leaves
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

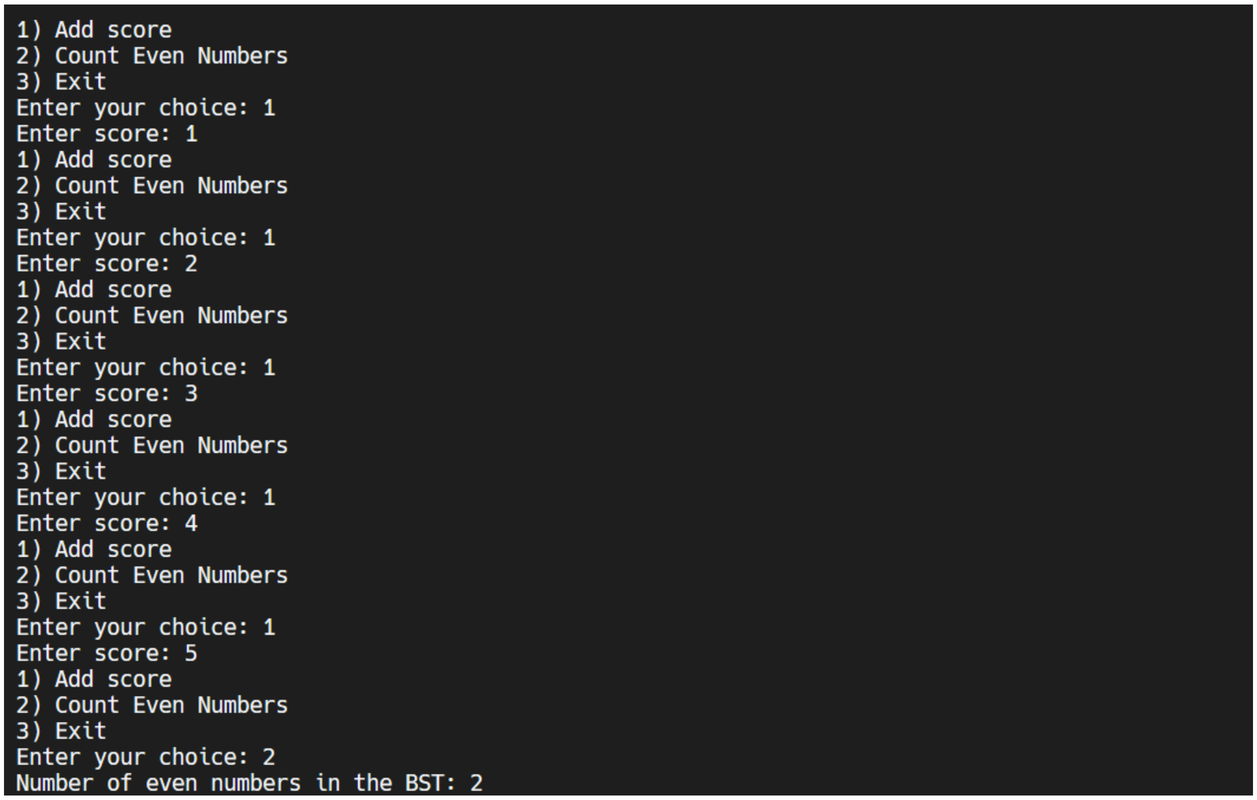


```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

// Structure definition for binary tree 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 tree
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 count the number of leaf nodes in the binary tree
int countLeaves(Node* root) {
    if (root == nullptr)
        return 0;
    if (root->left == nullptr && root->right == nullptr)
        return 1;
    return countLeaves(root->left) + countLeaves(root->right);
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Count Leaves\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 << "Number of leaves in the binary tree: " << countLeaves(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 count the number of even numbers in the BST.

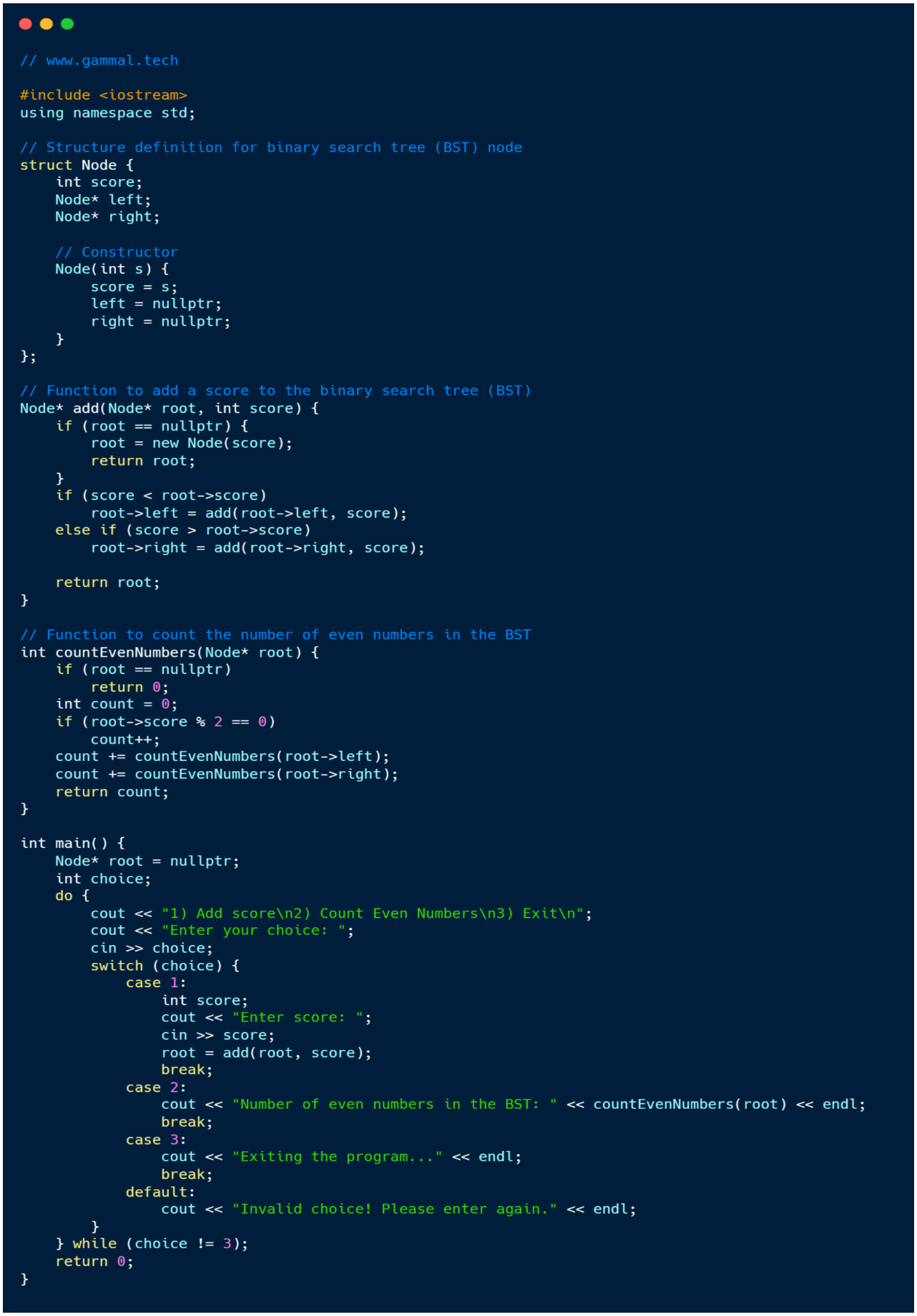
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم ثم قم بحساب عدد الأرقام الزوجية في شجرة البحث الثنائية.

Input & Output



```
1) Add score
2) Count Even Numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Count Even Numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Count Even Numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Count Even Numbers
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Count Even Numbers
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Count Even Numbers
3) Exit
Enter your choice: 2
Number of even numbers in the BST: 2
```

Solution

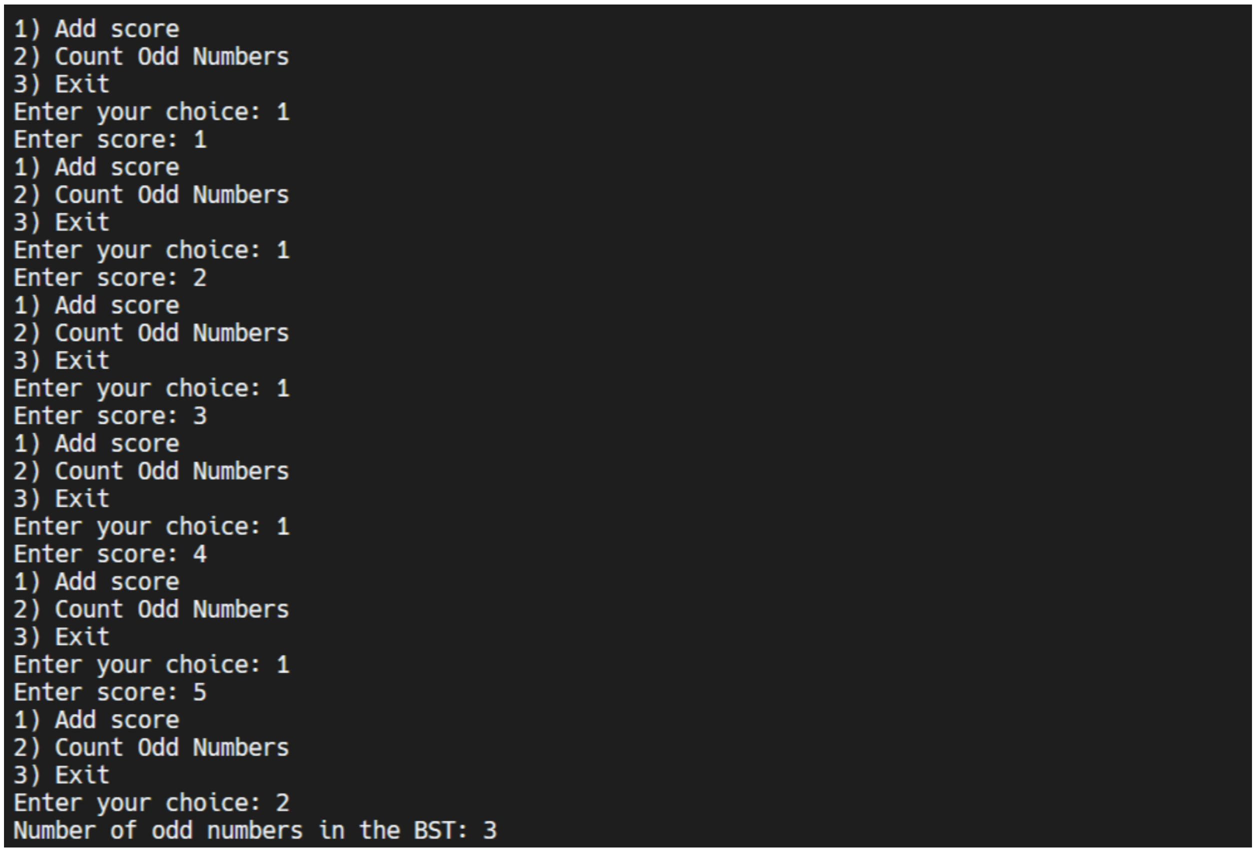


```cpp
// www.gammal.tech

#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 count the number of even numbers in the BST
int countEvenNumbers(Node* root) {
    if (root == nullptr)
        return 0;
    int count = 0;
    if (root->score % 2 == 0)
        count++;
    count += countEvenNumbers(root->left);
    count += countEvenNumbers(root->right);
    return count;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Count 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 << "Number of even numbers in the BST: " << countEvenNumbers(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 count the number of odd numbers in the BST.

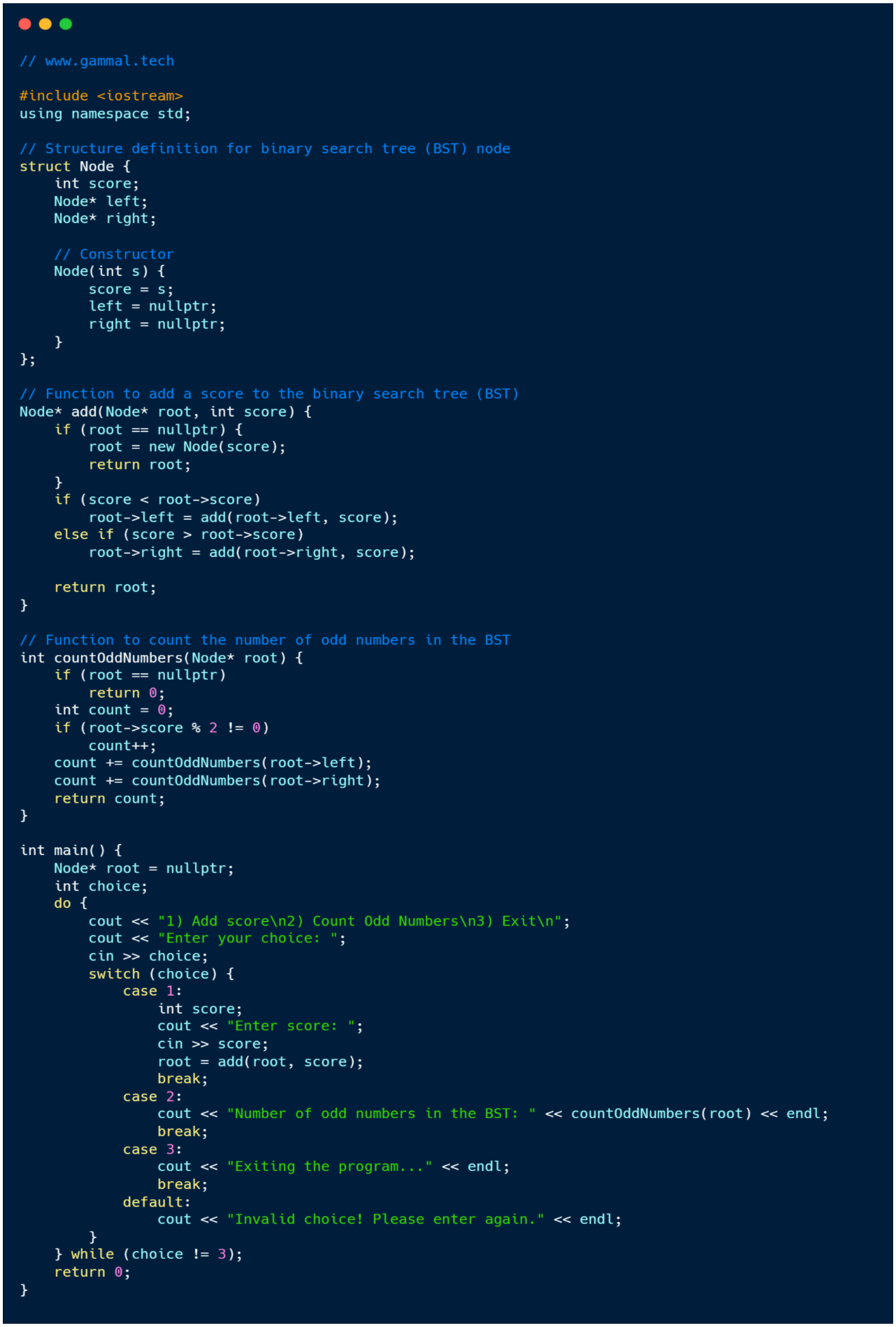
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم ثم قم بحساب عدد الأرقام الفردية في شجرة البحث الثنائية.

Input & Output



```
1) Add score
2) Count Odd Numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Count Odd Numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Count Odd Numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Count Odd Numbers
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Count Odd Numbers
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Count Odd Numbers
3) Exit
Enter your choice: 2
Number of odd numbers in the BST: 3
```

Solution

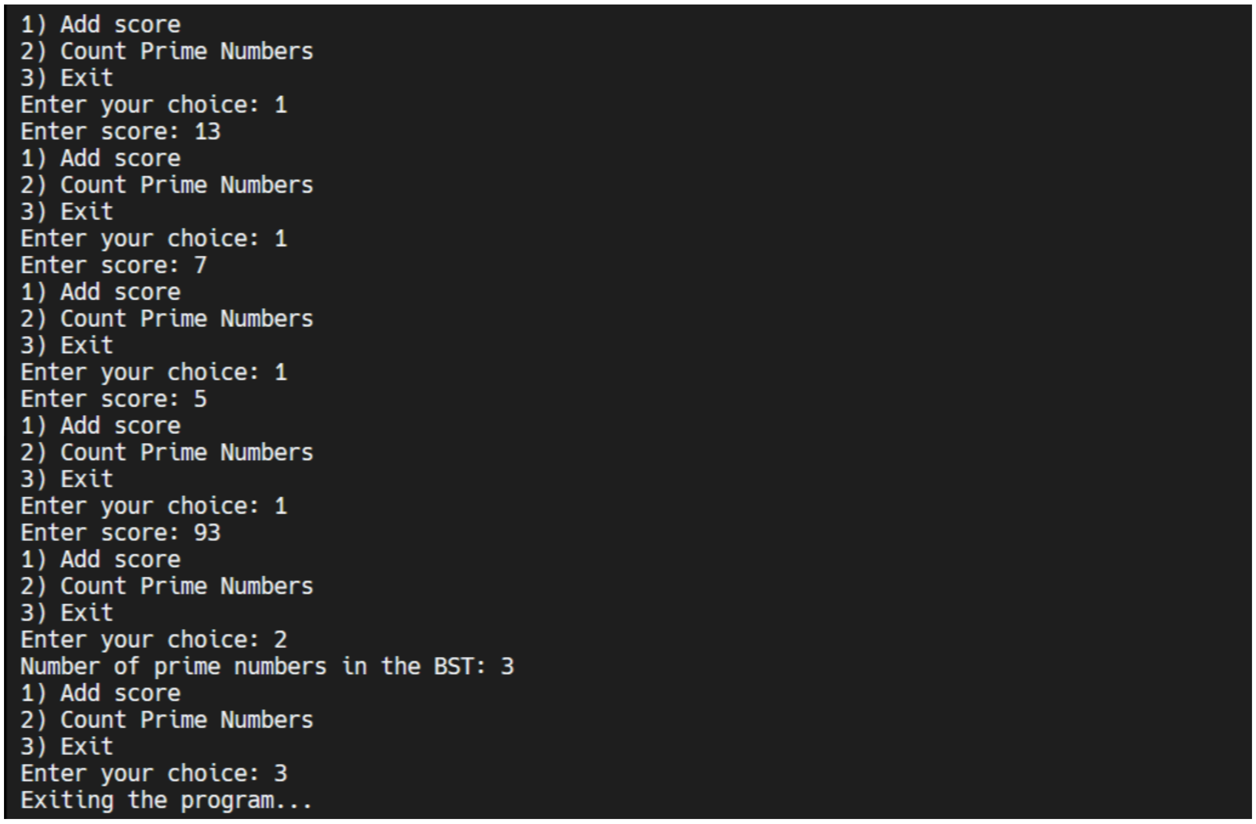


```cpp
// www.gammal.tech

#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 count the number of odd numbers in the BST
int countOddNumbers(Node* root) {
    if (root == nullptr)
        return 0;
    int count = 0;
    if (root->score % 2 != 0)
        count++;
    count += countOddNumbers(root->left);
    count += countOddNumbers(root->right);
    return count;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Count 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 << "Number of odd numbers in the BST: " << countOddNumbers(root) << 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 count the number of prime numbers in the BST.

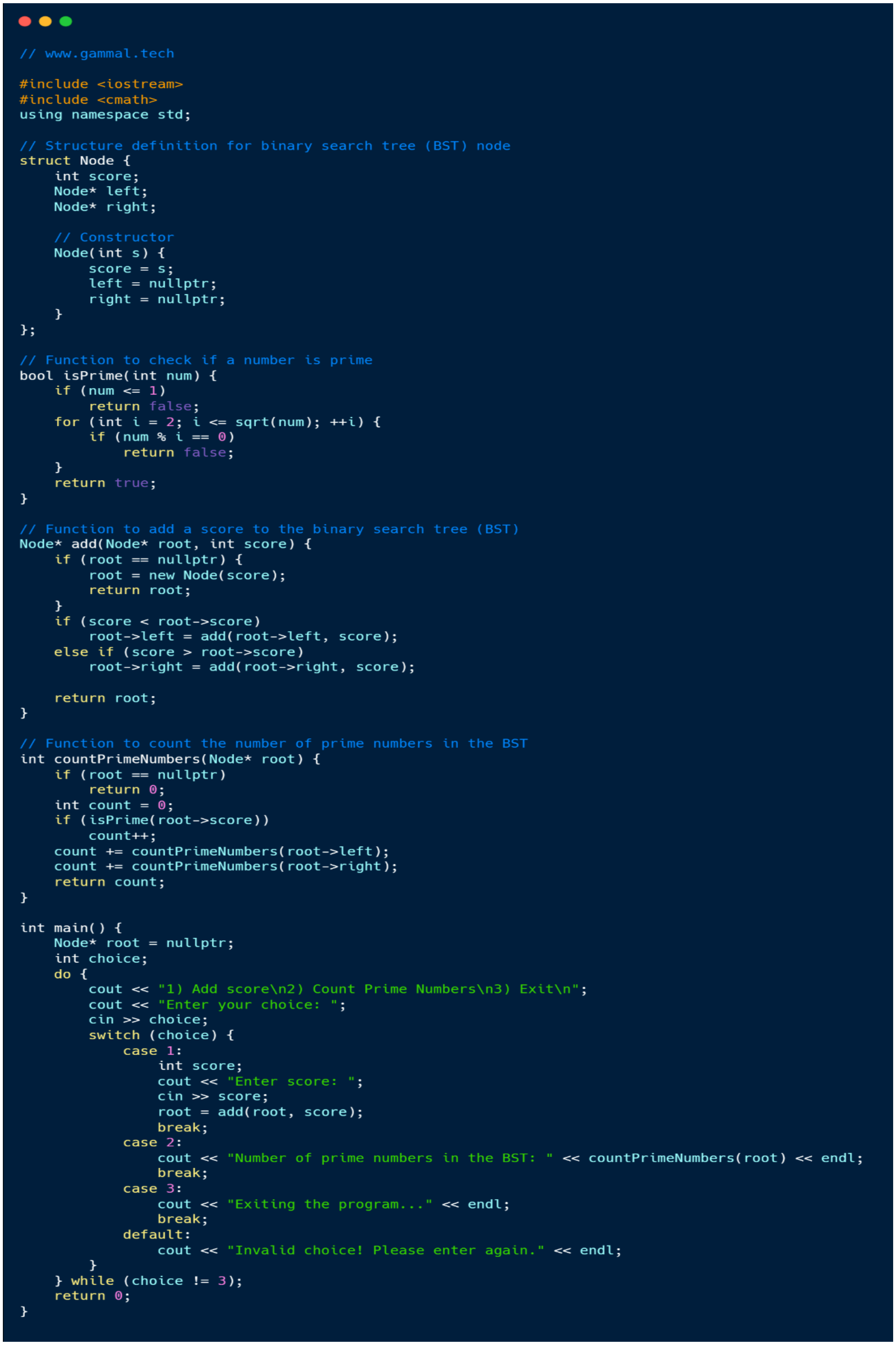
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم ثم قم بحساب عدد الأعداد الأولية في شجرة البحث الثنائية.

Input & Output



```
1) Add score
2) Count Prime Numbers
3) Exit
Enter your choice: 1
Enter score: 13
1) Add score
2) Count Prime Numbers
3) Exit
Enter your choice: 1
Enter score: 7
1) Add score
2) Count Prime Numbers
3) Exit
Enter your choice: 1
Enter score: 5
1) Add score
2) Count Prime Numbers
3) Exit
Enter your choice: 1
Enter score: 93
1) Add score
2) Count Prime Numbers
3) Exit
Enter your choice: 2
Number of prime numbers in the BST: 3
1) Add score
2) Count Prime Numbers
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

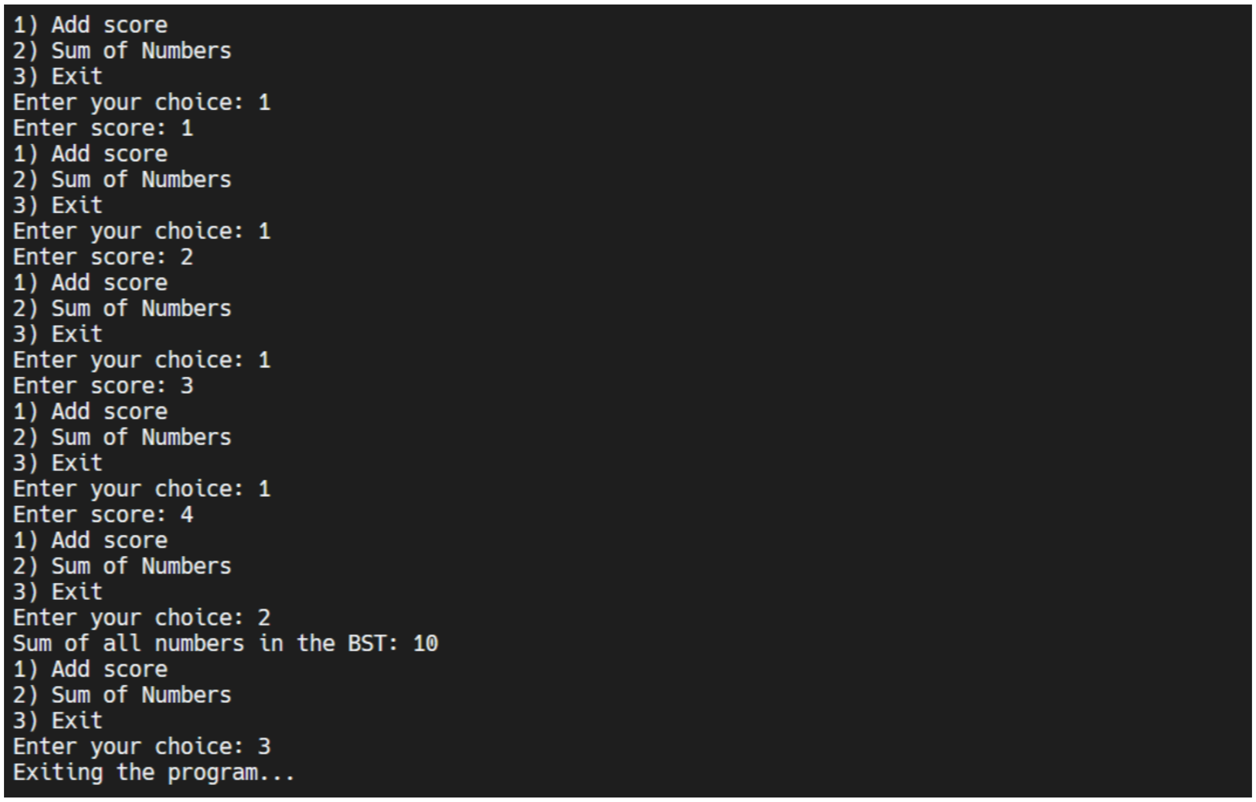


```cpp
// www.gammal.tech

#include <iostream>
#include <cmath>
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 check if a number is prime
bool isPrime(int num) {
    if (num <= 1)
        return false;
    for (int i = 2; i <= sqrt(num); ++i) {
        if (num % i == 0)
            return false;
    }
    return true;
}

// 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 count the number of prime numbers in the BST
int countPrimeNumbers(Node* root) {
    if (root == nullptr)
        return 0;
    int count = 0;
    if (isPrime(root->score))
        count++;
    count += countPrimeNumbers(root->left);
    count += countPrimeNumbers(root->right);
    return count;
}

int main() {
    Node* root = nullptr;
    int choice;
    do {
        cout << "1) Add score\n2) Count Prime 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 << "Number of prime numbers in the BST: " << countPrimeNumbers(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 construct a binary search tree (BST) based on user input scores and then find the sum of all numbers in the BST.

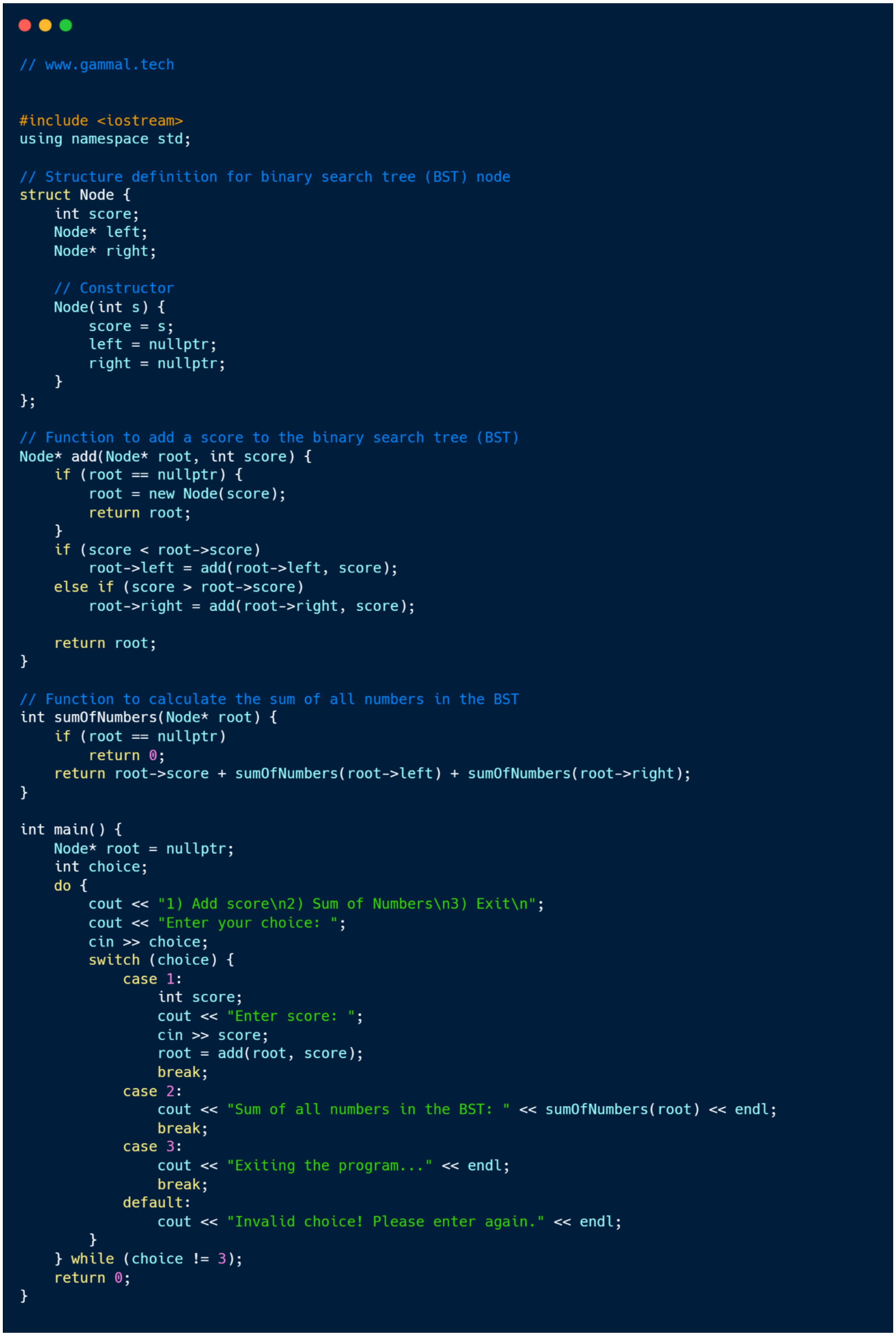
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم ابحث عن مجموع جميع الأرقام في شجرة البحث الثنائية.

Input & Output



```
1) Add score
2) Sum of Numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Sum of Numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Sum of Numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Sum of Numbers
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Sum of Numbers
3) Exit
Enter your choice: 2
Sum of all numbers in the BST: 10
1) Add score
2) Sum of Numbers
3) Exit
Enter your choice: 3
Exiting the program...
```

Solution

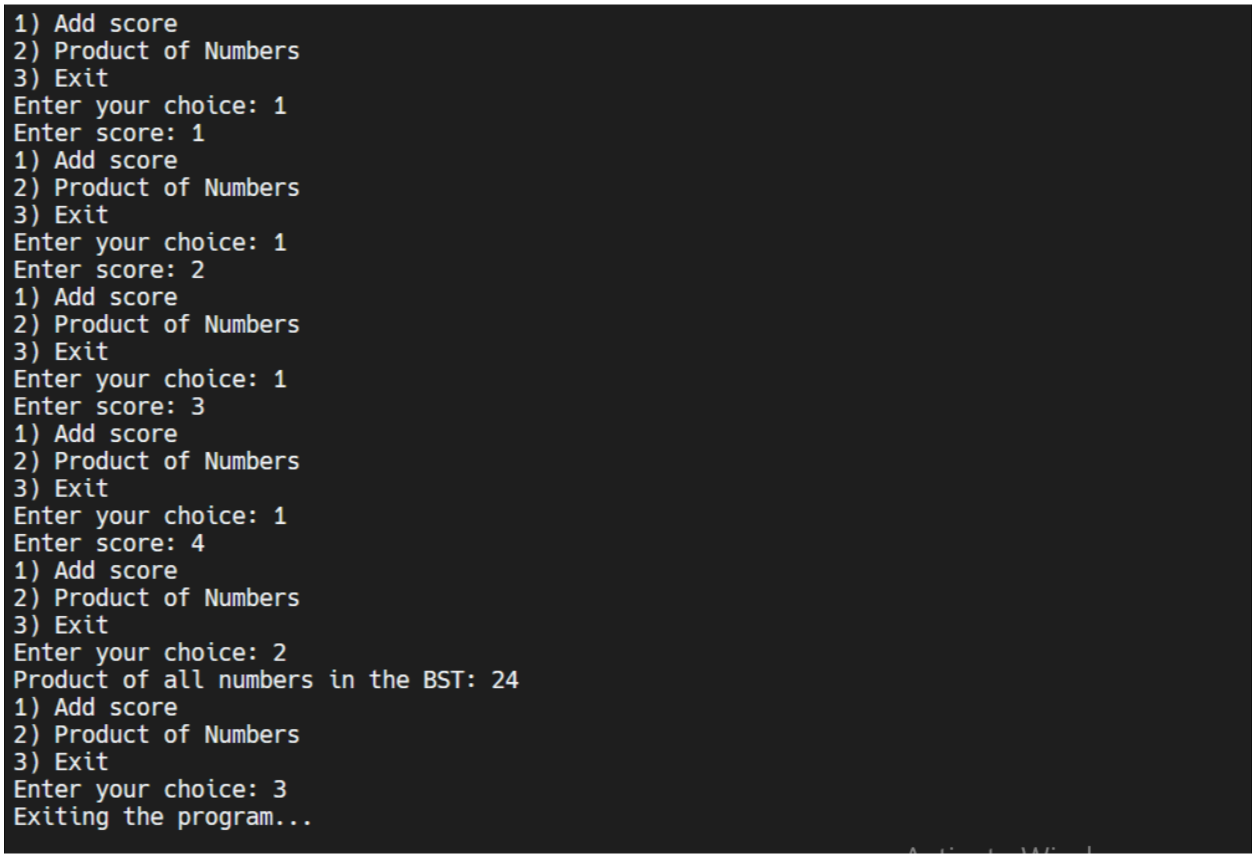


```cpp
// www.gammal.tech


#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);
}

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

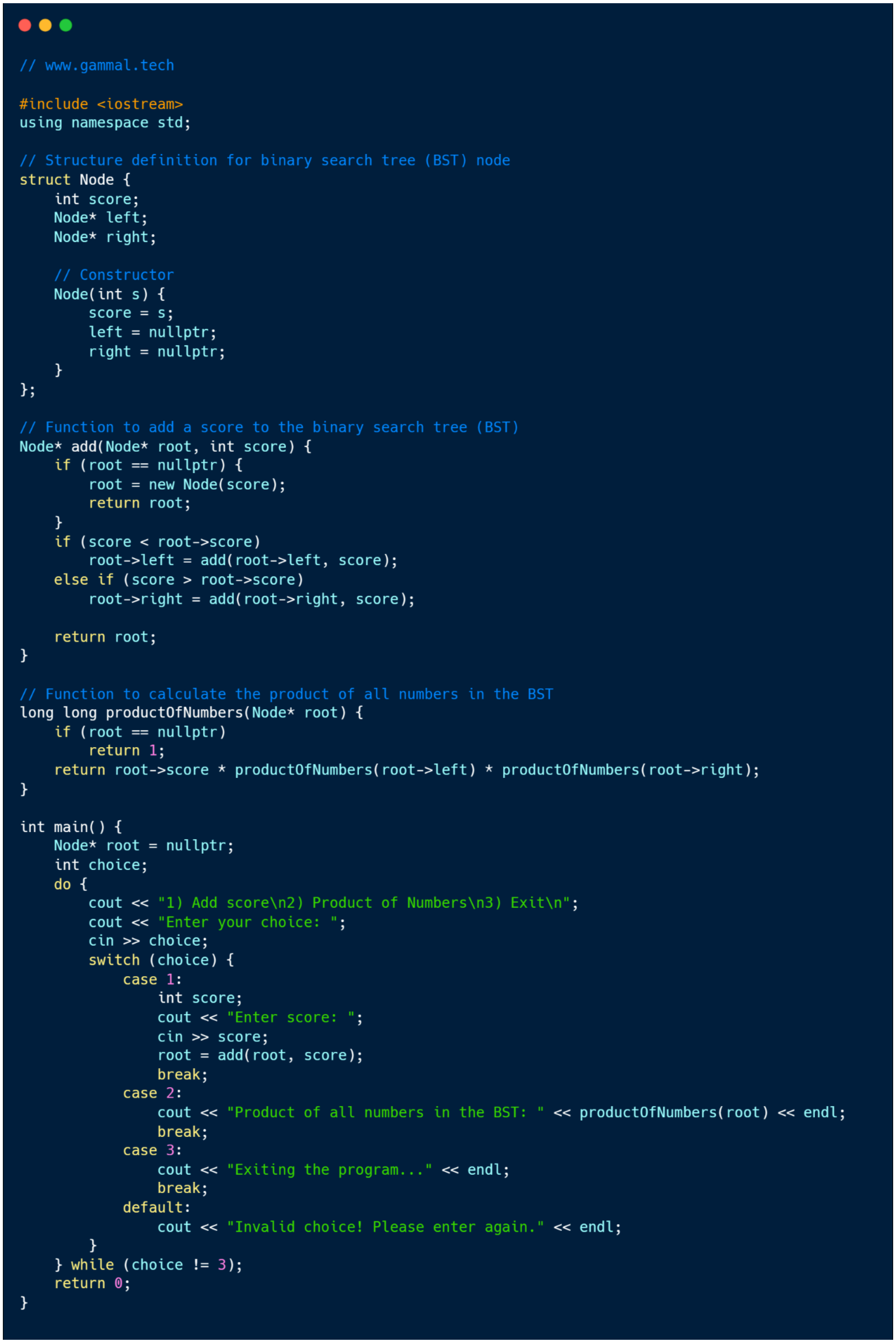
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم ابحث عن حاصل ضرب جميع الأرقام في شجرة البحث الثنائية.

Input & Output



```
1) Add score
2) Product of Numbers
3) Exit
Enter your choice: 1
Enter score: 1
1) Add score
2) Product of Numbers
3) Exit
Enter your choice: 1
Enter score: 2
1) Add score
2) Product of Numbers
3) Exit
Enter your choice: 1
Enter score: 3
1) Add score
2) Product of Numbers
3) Exit
Enter your choice: 1
Enter score: 4
1) Add score
2) Product of Numbers
3) Exit
Enter your choice: 2
Product of all numbers in the BST: 24
1) Add score
2) Product 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 product of all numbers in the BST
long long productOfNumbers(Node* root) {
    if (root == nullptr)
        return 1;
    return root->score * productOfNumbers(root->left) * productOfNumbers(root->right);
}

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