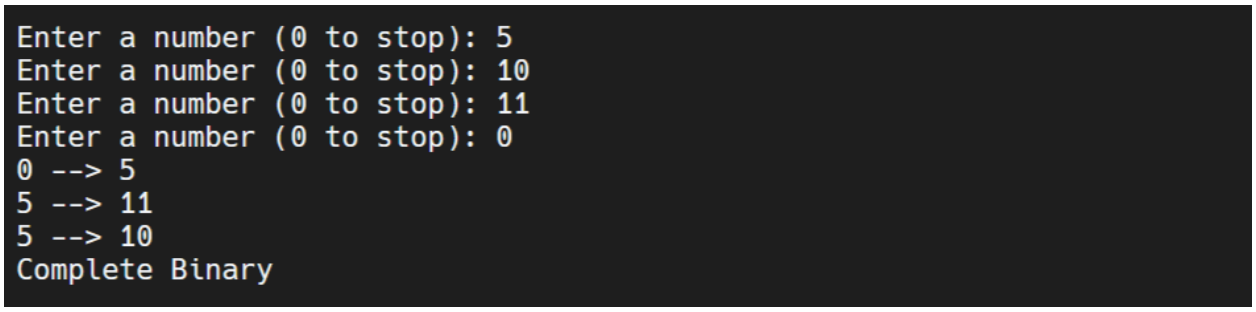
1- Write a C++ program to create a binary tree and check if it is a complete binary tree or not. The program should perform the following tasks:

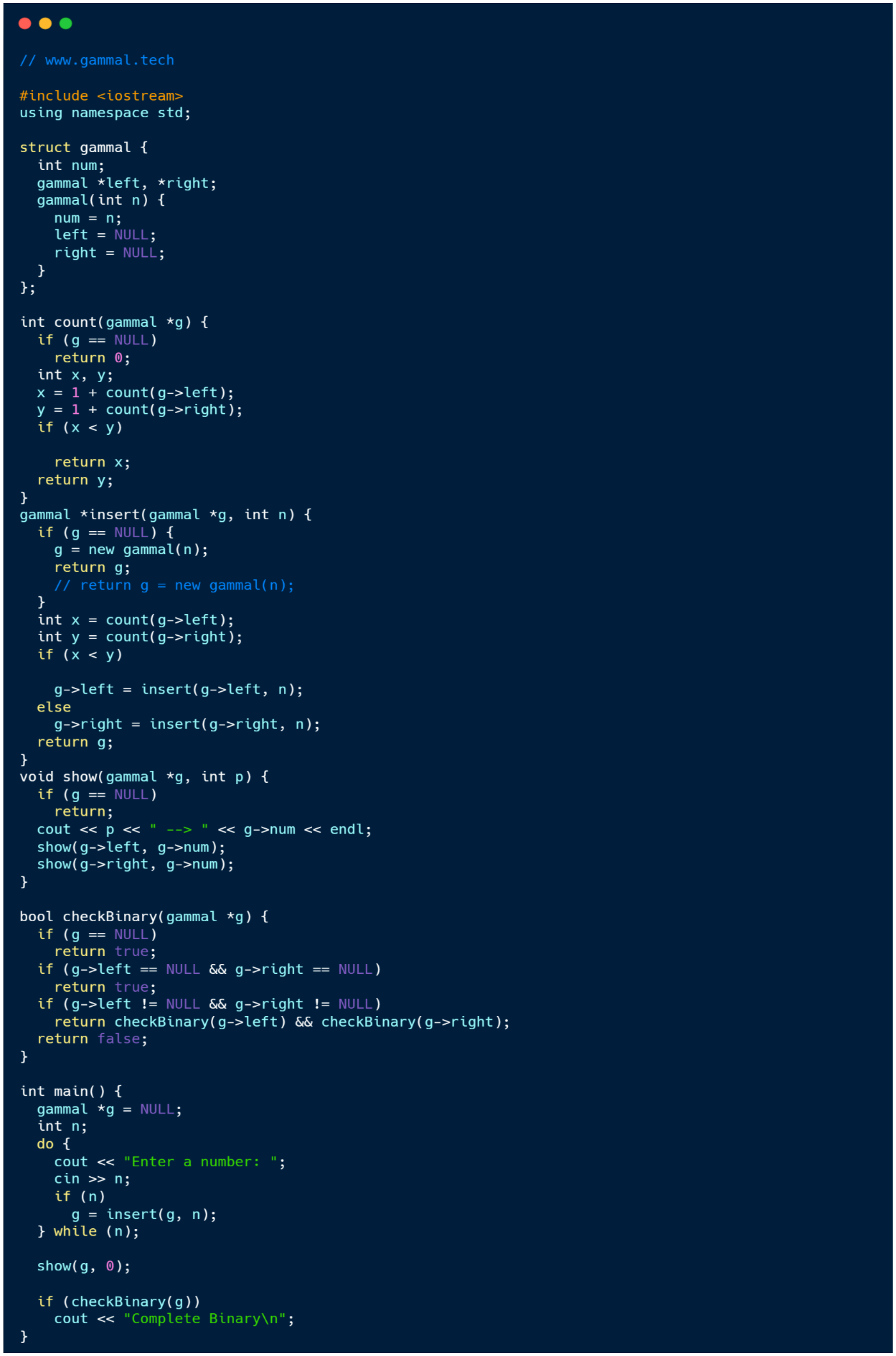
اكتب برنامج C++ لإنشاء شجرة ثنائية وتحقق مما إذا كانت شجرة ثنائية كاملة أم لا. يجب أن يقوم البرنامج بالمهام التالية:

Input & Output



```
Enter a number (0 to stop): 5
Enter a number (0 to stop): 10
Enter a number (0 to stop): 11
Enter a number (0 to stop): 0
0 --> 5
5 --> 11
5 --> 10
Complete Binary
```

Solution

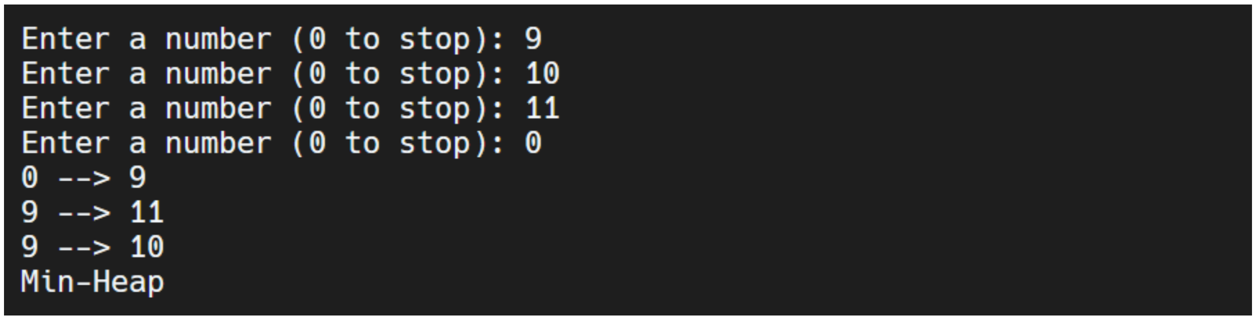


```
// www.gammal.tech

#include <iostream>
using namespace std;

struct gammal {
  int num;
  gammal *left, *right;
  gammal(int n) {
    num = n;
    left = NULL;
    right = NULL;
  }
};

int count(gammal *g) {
  if (g == NULL)
    return 0;
  int x, y;
  x = 1 + count(g->left);
  y = 1 + count(g->right);
  if (x < y)

    return x;
  return y;
}
gammal *insert(gammal *g, int n) {
  if (g == NULL) {
    g = new gammal(n);
    return g;
    // return g = new gammal(n);
  }
  int x = count(g->left);
  int y = count(g->right);
  if (x < y)

    g->left = insert(g->left, n);
  else
    g->right = insert(g->right, n);
  return g;
}
void show(gammal *g, int p) {
  if (g == NULL)
    return;
  cout << p << " --> " << g->num << endl;
  show(g->left, g->num);
  show(g->right, g->num);
}

bool checkBinary(gammal *g) {
  if (g == NULL)
    return true;
  if (g->left == NULL && g->right == NULL)
    return true;
  if (g->left != NULL && g->right != NULL)
    return checkBinary(g->left) && checkBinary(g->right);
  return false;
}

int main() {
  gammal *g = NULL;
  int n;
  do {
    cout << "Enter a number: ";
    cin >> n;
    if (n)
      g = insert(g, n);
  } while (n);

  show(g, 0);

  if (checkBinary(g))
    cout << "Complete Binary\n";
}
```

2- Write a C++ program to create a binary tree and check if it is a Min-heap or not.

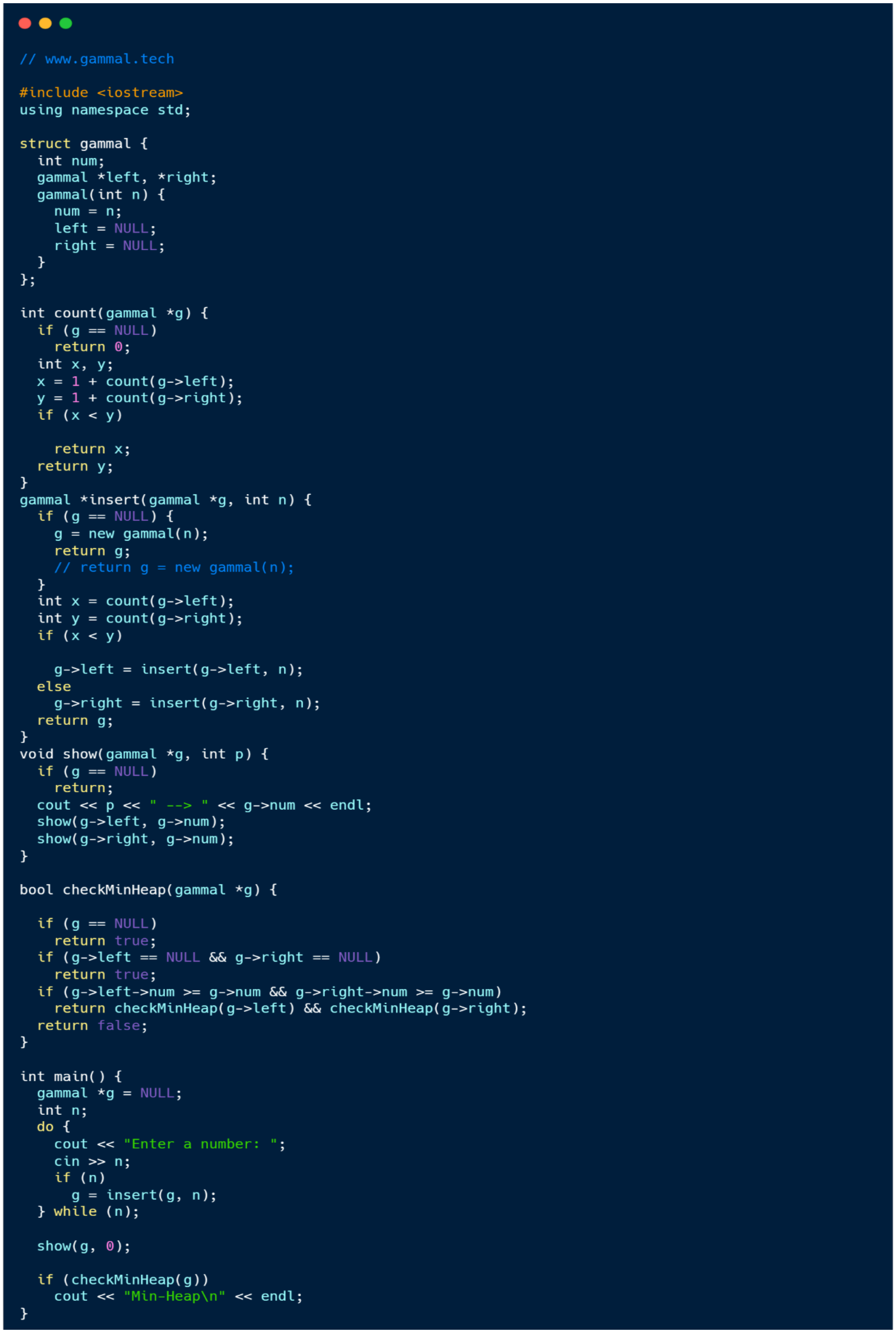
اكتب برنامج C++ لإنشاء شجرة ثنائية وتحقق مما إذا كانت Min-heap أم لا.

Input & Output



```
Enter a number (0 to stop): 9
Enter a number (0 to stop): 10
Enter a number (0 to stop): 11
Enter a number (0 to stop): 0
0 --> 9
9 --> 11
9 --> 10
Min-Heap
```

Solution

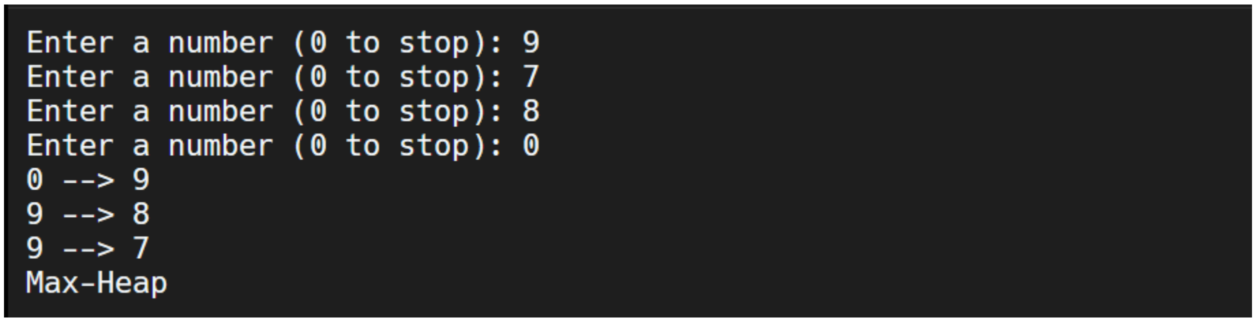


```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

struct gammal {
  int num;
  gammal *left, *right;
  gammal(int n) {
    num = n;
    left = NULL;
    right = NULL;
  }
};

int count(gammal *g) {
  if (g == NULL)
    return 0;
  int x, y;
  x = 1 + count(g->left);
  y = 1 + count(g->right);
  if (x < y)

    return x;
  return y;
}
gammal *insert(gammal *g, int n) {
  if (g == NULL) {
    g = new gammal(n);
    return g;
    // return g = new gammal(n);
  }
  int x = count(g->left);
  int y = count(g->right);
  if (x < y)

    g->left = insert(g->left, n);
  else
    g->right = insert(g->right, n);
  return g;
}
void show(gammal *g, int p) {
  if (g == NULL)
    return;
  cout << p << " --> " << g->num << endl;
  show(g->left, g->num);
  show(g->right, g->num);
}

bool checkMinHeap(gammal *g) {

  if (g == NULL)
    return true;
  if (g->left == NULL && g->right == NULL)
    return true;
  if (g->left->num >= g->num && g->right->num >= g->num)
    return checkMinHeap(g->left) && checkMinHeap(g->right);
  return false;
}

int main() {
  gammal *g = NULL;
  int n;
  do {
    cout << "Enter a number: ";
    cin >> n;
    if (n)
      g = insert(g, n);
  } while (n);

  show(g, 0);

  if (checkMinHeap(g))
    cout << "Min-Heap\n" << endl;
}
```

3- Write a C++ program to create a binary tree and check if it is a Max-heap or not.

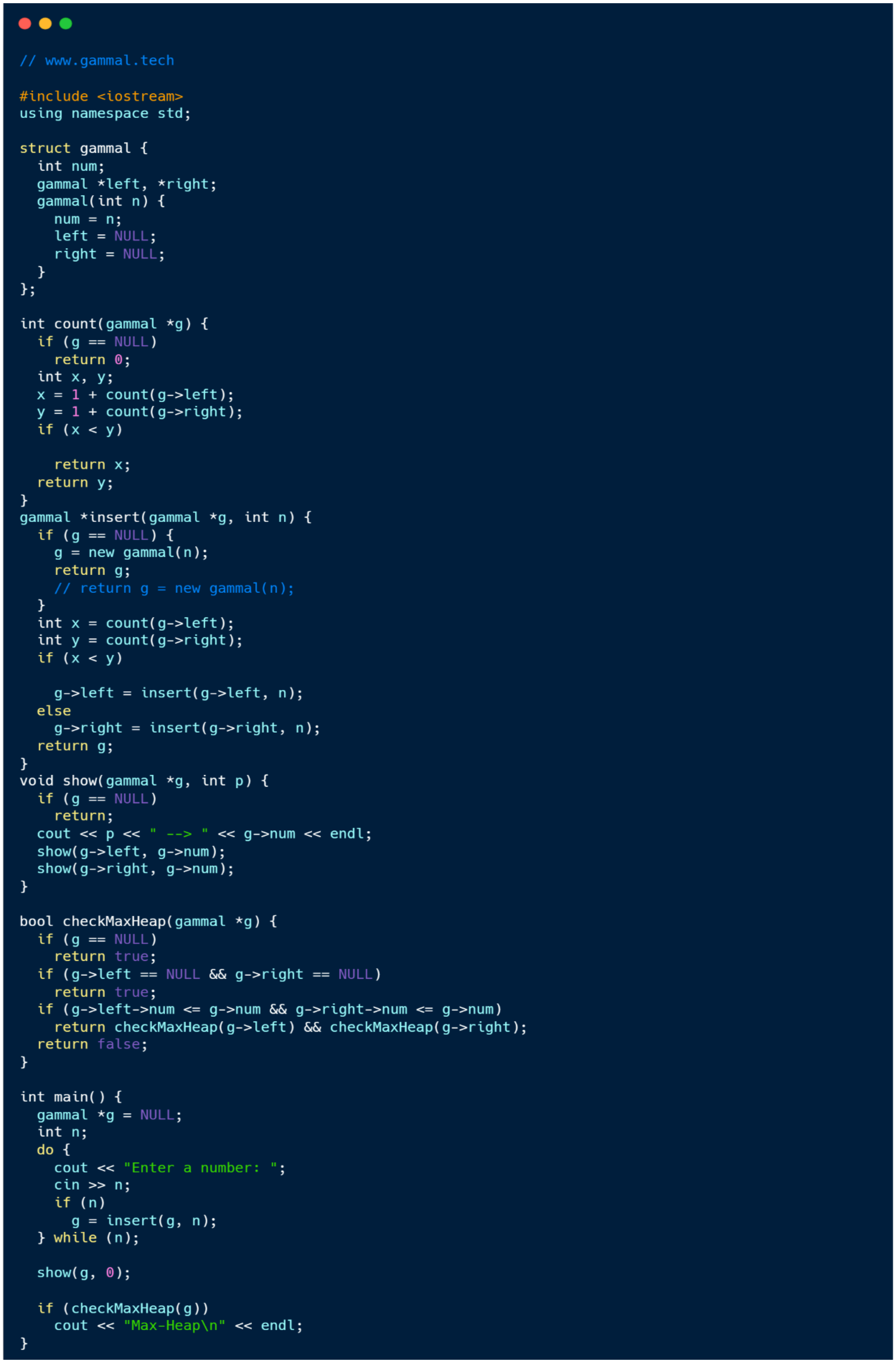
اكتب برنامج C++ لإنشاء شجرة ثنائية وتحقق مما إذا كانت Max-heap أم لا.

Input & Output



```
Enter a number (0 to stop): 9
Enter a number (0 to stop): 7
Enter a number (0 to stop): 8
Enter a number (0 to stop): 0
0 --> 9
9 --> 8
9 --> 7
Max-Heap
```

Solution

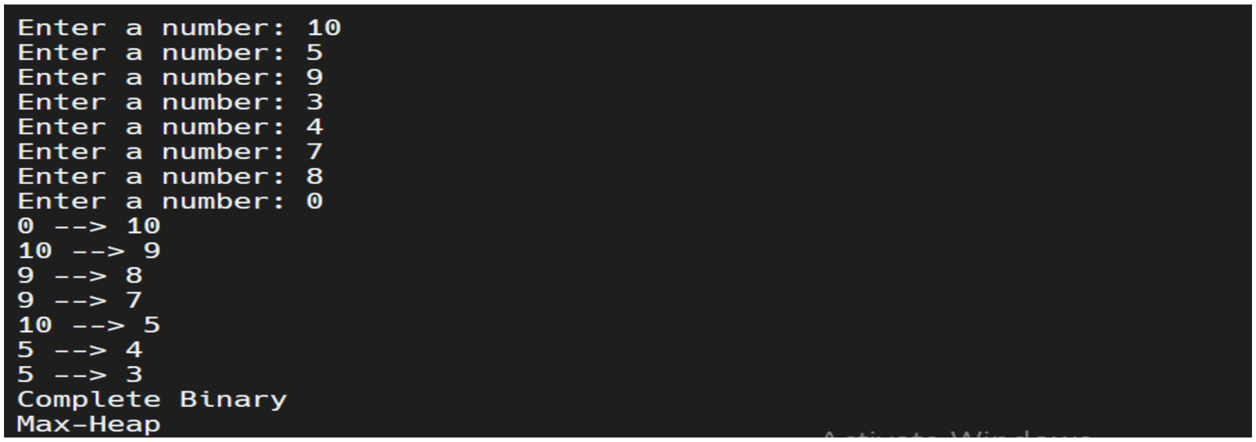


```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

struct gammal {
  int num;
  gammal *left, *right;
  gammal(int n) {
    num = n;
    left = NULL;
    right = NULL;
  }
};

int count(gammal *g) {
  if (g == NULL)
    return 0;
  int x, y;
  x = 1 + count(g->left);
  y = 1 + count(g->right);
  if (x < y)

    return x;
  return y;
}
gammal *insert(gammal *g, int n) {
  if (g == NULL) {
    g = new gammal(n);
    return g;
    // return g = new gammal(n);
  }
  int x = count(g->left);
  int y = count(g->right);
  if (x < y)

    g->left = insert(g->left, n);
  else
    g->right = insert(g->right, n);
  return g;
}
void show(gammal *g, int p) {
  if (g == NULL)
    return;
  cout << p << " --> " << g->num << endl;
  show(g->left, g->num);
  show(g->right, g->num);
}

bool checkMaxHeap(gammal *g) {
  if (g == NULL)
    return true;
  if (g->left == NULL && g->right == NULL)
    return true;
  if (g->left->num <= g->num && g->right->num <= g->num)
    return checkMaxHeap(g->left) && checkMaxHeap(g->right);
  return false;
}

int main() {
  gammal *g = NULL;
  int n;
  do {
    cout << "Enter a number: ";
    cin >> n;
    if (n)
      g = insert(g, n);
  } while (n);

  show(g, 0);

  if (checkMaxHeap(g))
    cout << "Max-Heap\n" << endl;
}
```

4- Write a C++ program to create a binary tree and check if it is a Min-heap or a Max-heap or a complete binary tree.

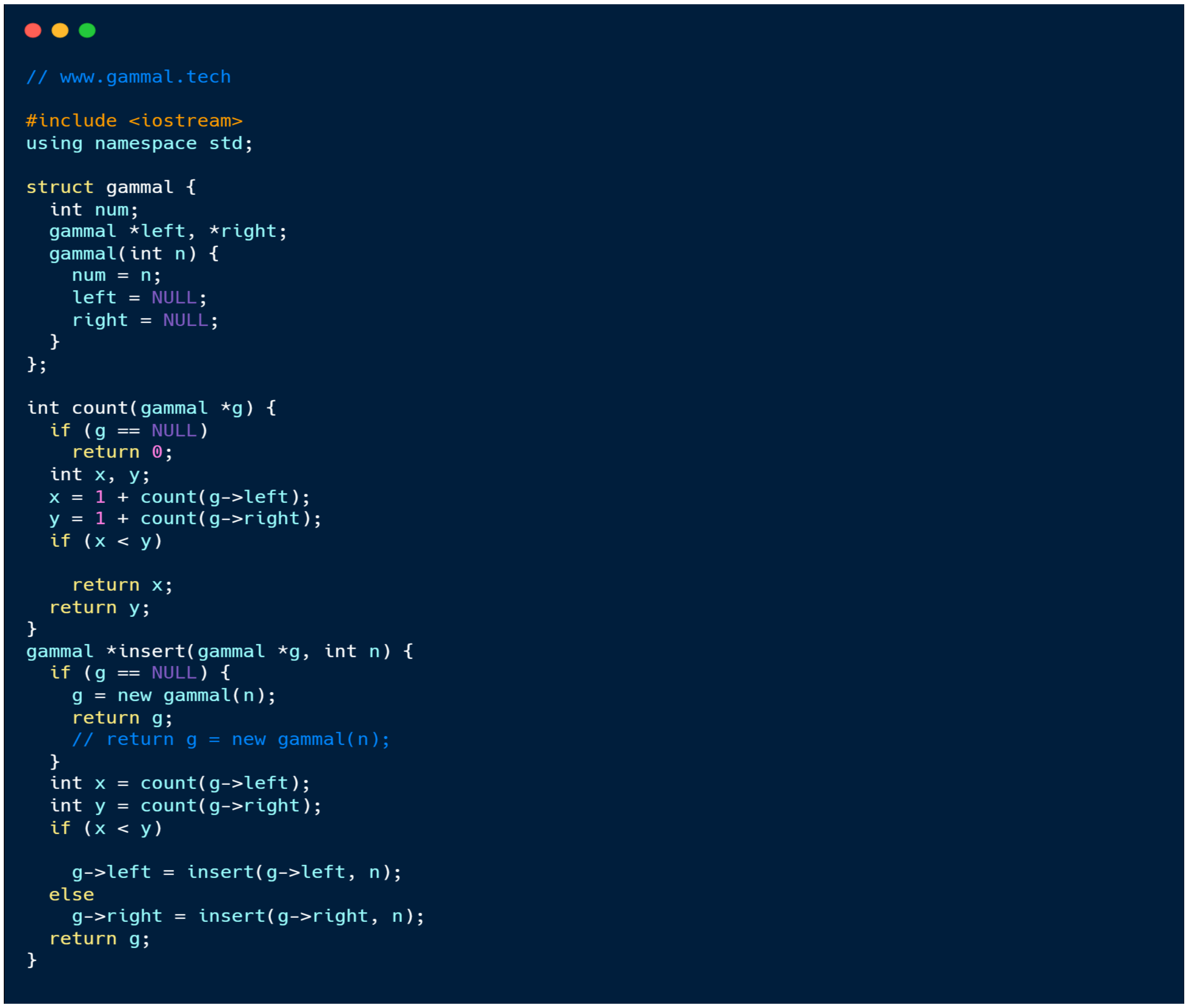
اكتب برنامج C++ لإنشاء شجرة ثنائية وتحقق مما إذا كانت Min-heap أو Max-heap أو شجرة ثنائية كاملة.

Input & Output

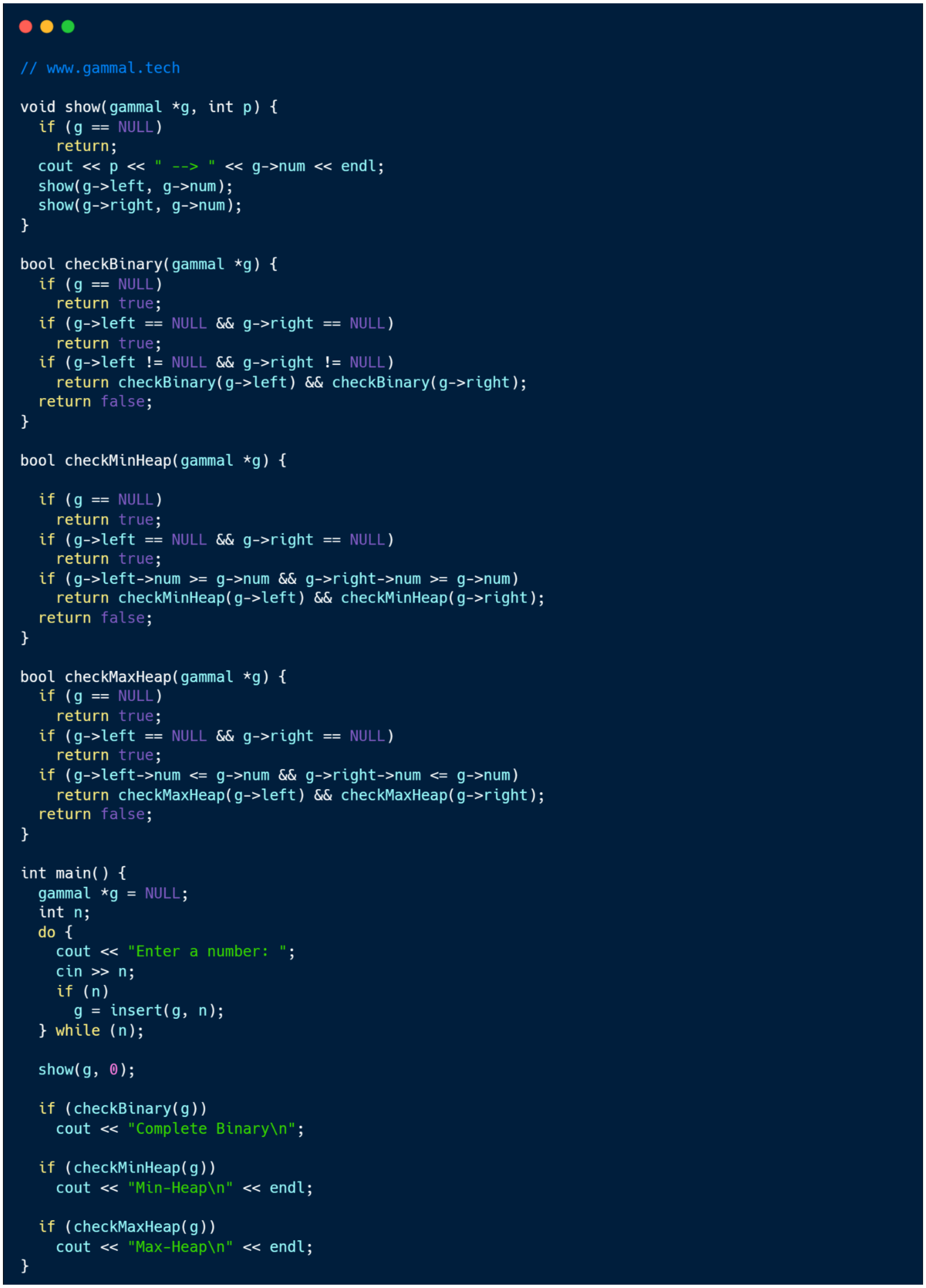


```
Enter a number: 10
Enter a number: 5
Enter a number: 9
Enter a number: 3
Enter a number: 4
Enter a number: 7
Enter a number: 8
Enter a number: 0
0 --> 10
10 --> 9
9 --> 8
9 --> 7
10 --> 5
5 --> 4
5 --> 3
Complete Binary
Max-Heap
```

Solution



```cpp
// www.gammal.tech

#include <iostream>
using namespace std;

struct gammal {
  int num;
  gammal *left, *right;
  gammal(int n) {
    num = n;
    left = NULL;
    right = NULL;
  }
};

int count(gammal *g) {
  if (g == NULL)
    return 0;
  int x, y;
  x = 1 + count(g->left);
  y = 1 + count(g->right);
  if (x < y)

    return x;
  return y;
}
gammal *insert(gammal *g, int n) {
  if (g == NULL) {
    g = new gammal(n);
    return g;
    // return g = new gammal(n);
  }
  int x = count(g->left);
  int y = count(g->right);
  if (x < y)

    g->left = insert(g->left, n);
  else
    g->right = insert(g->right, n);
  return g;
}
```

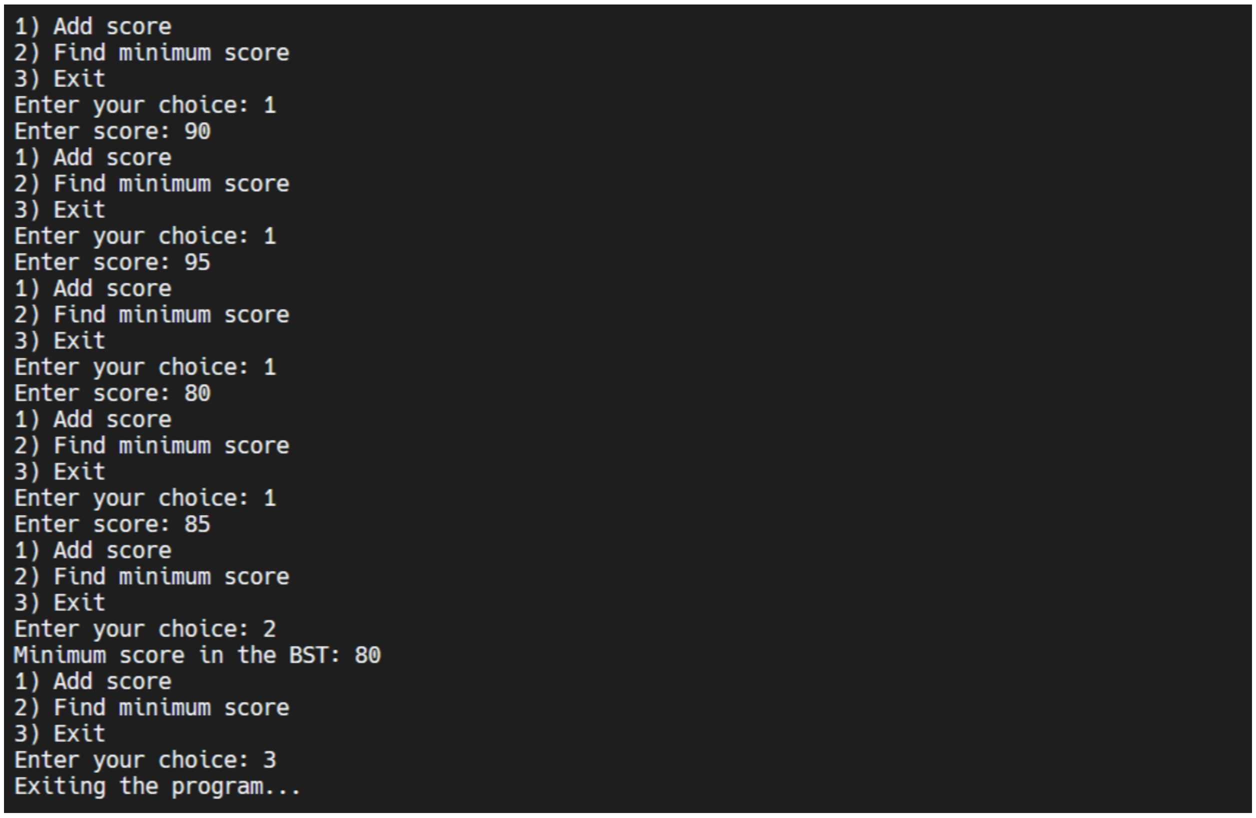


```cpp
// www.gammal.tech

void show(gammal *g, int p) {
  if (g == NULL)
    return;
  cout << p << " --> " << g->num << endl;
  show(g->left, g->num);
  show(g->right, g->num);
}

bool checkBinary(gammal *g) {
  if (g == NULL)
    return true;
  if (g->left == NULL && g->right == NULL)
    return true;
  if (g->left != NULL && g->right != NULL)
    return checkBinary(g->left) && checkBinary(g->right);
  return false;
}

bool checkMinHeap(gammal *g) {

  if (g == NULL)
    return true;
  if (g->left == NULL && g->right == NULL)
    return true;
  if (g->left->num >= g->num && g->right->num >= g->num)
    return checkMinHeap(g->left) && checkMinHeap(g->right);
  return false;
}

bool checkMaxHeap(gammal *g) {
  if (g == NULL)
    return true;
  if (g->left == NULL && g->right == NULL)
    return true;
  if (g->left->num <= g->num && g->right->num <= g->num)
    return checkMaxHeap(g->left) && checkMaxHeap(g->right);
  return false;
}

int main() {
  gammal *g = NULL;
  int n;
  do {
    cout << "Enter a number: ";
    cin >> n;
    if (n)
      g = insert(g, n);
  } while (n);

  show(g, 0);

  if (checkBinary(g))
    cout << "Complete Binary\n";

  if (checkMinHeap(g))
    cout << "Min-Heap\n" << endl;

  if (checkMaxHeap(g))
    cout << "Max-Heap\n" << endl;
}
```

5- 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.

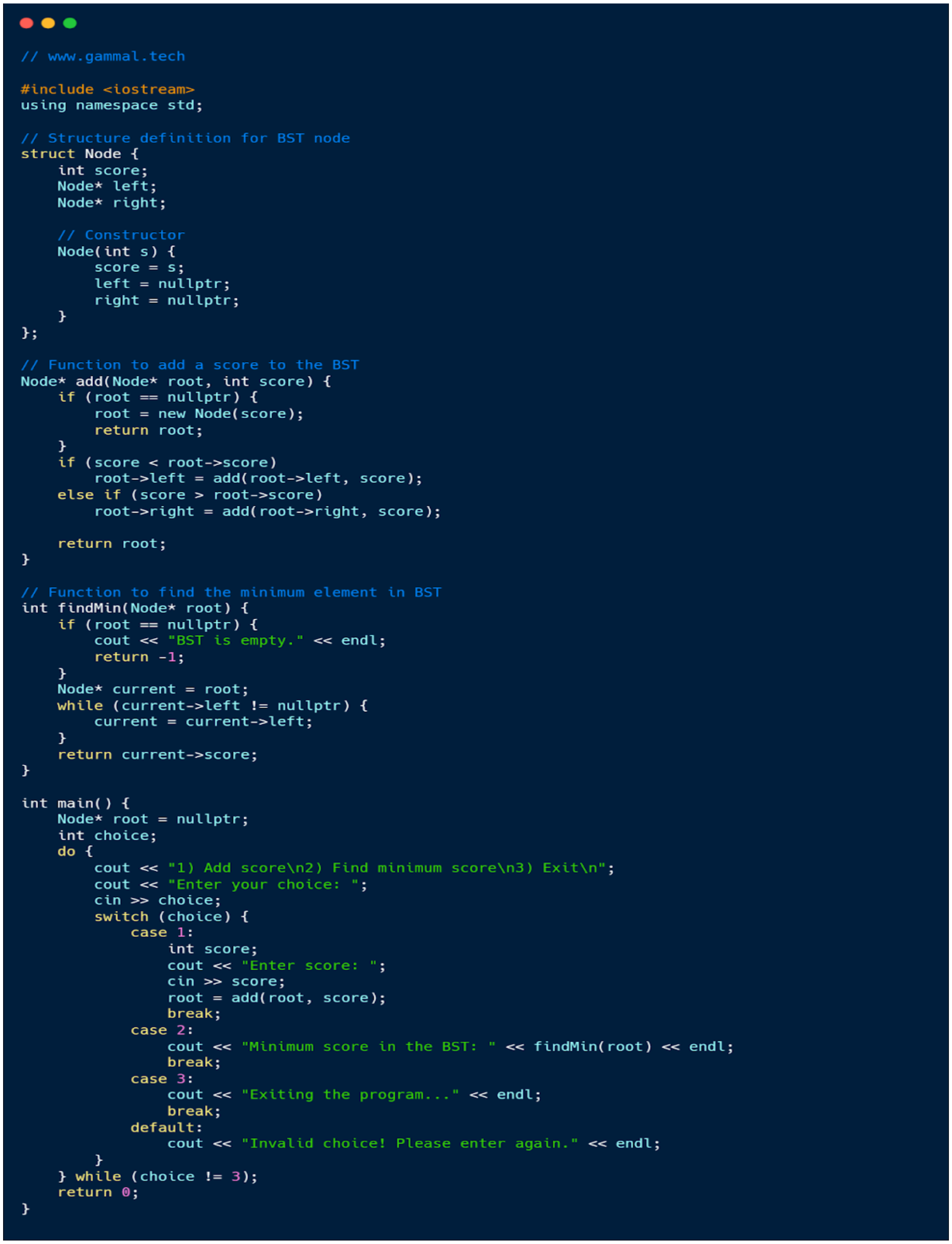
اكتب برنامج C++ للعثور على الحد الأدنى من العناصر في شجرة البحث الثنائية (BST). يجب أن يطالب البرنامج المستخدم بإدخال الدرجات لبناء BST. بعد إنشاء BST، يجب العثور على الحد الأدنى من الدرجات الموجودة في الشجرة وعرضها.

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

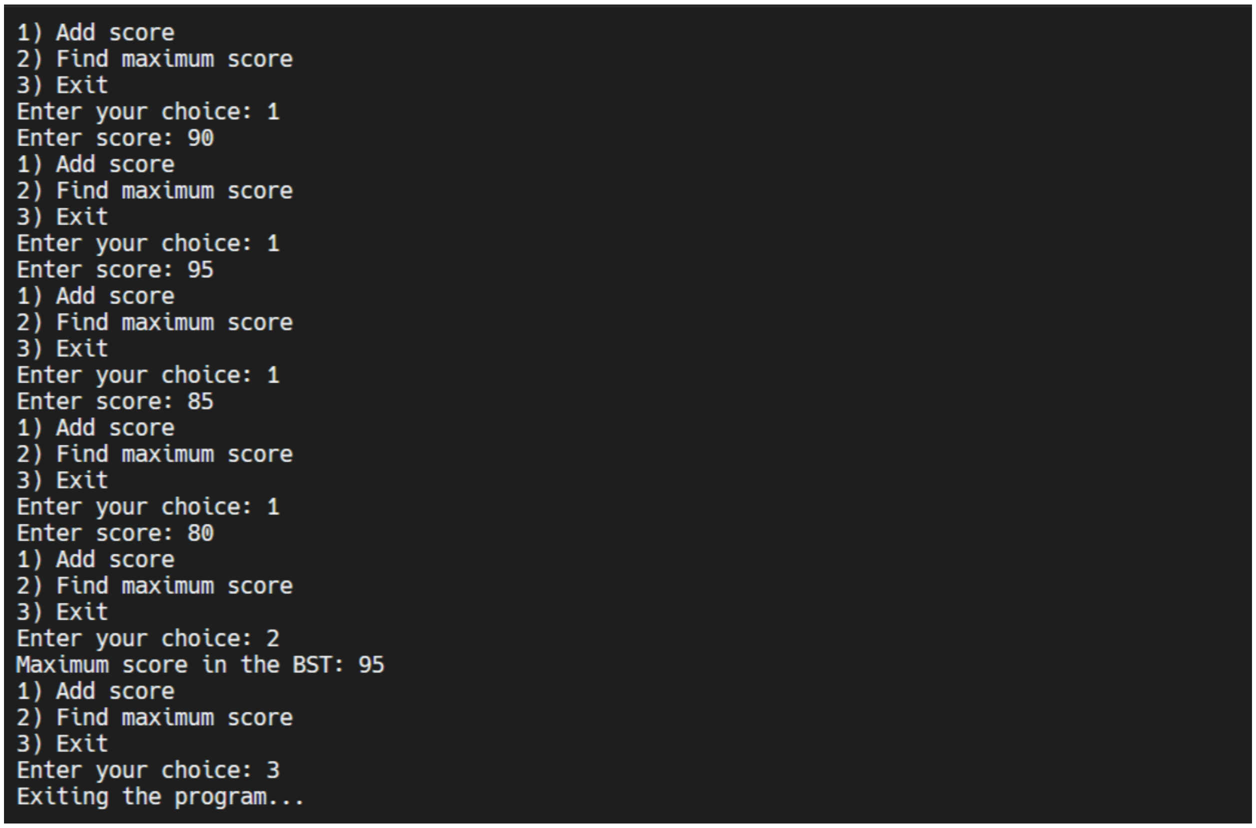


```
// www.gammal.tech

#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;
}
```

6- 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.

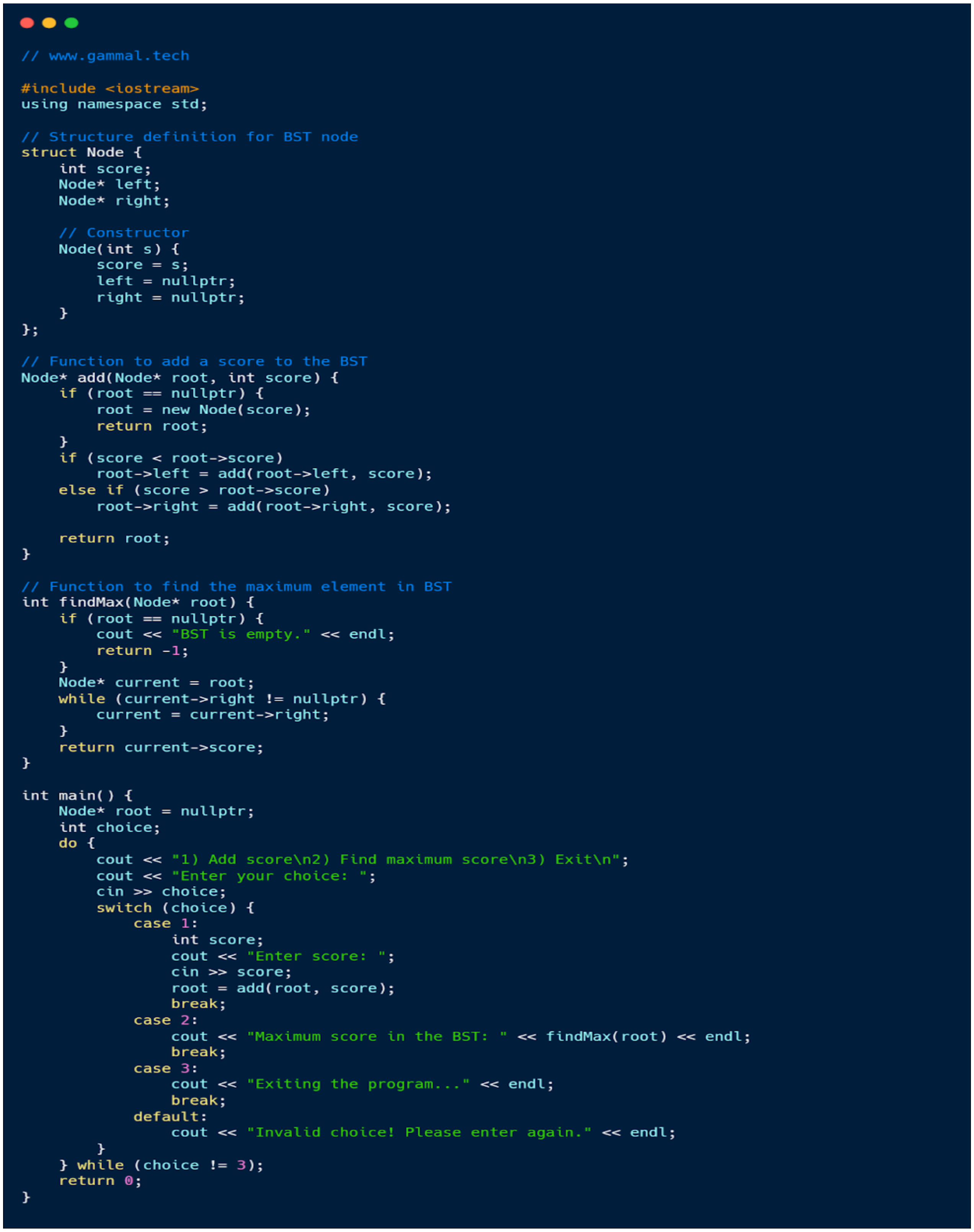
اكتب برنامج C++ للعثور على الحد الأقصى للعنصر في شجرة البحث الثنائية (BST). يجب أن يطالب البرنامج المستخدم بإدخال الدرجات لبناء BST. بعد إنشاء BST، يجب العثور على الحد الأقصى للدرجات الموجودة في الشجرة وعرضها.

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

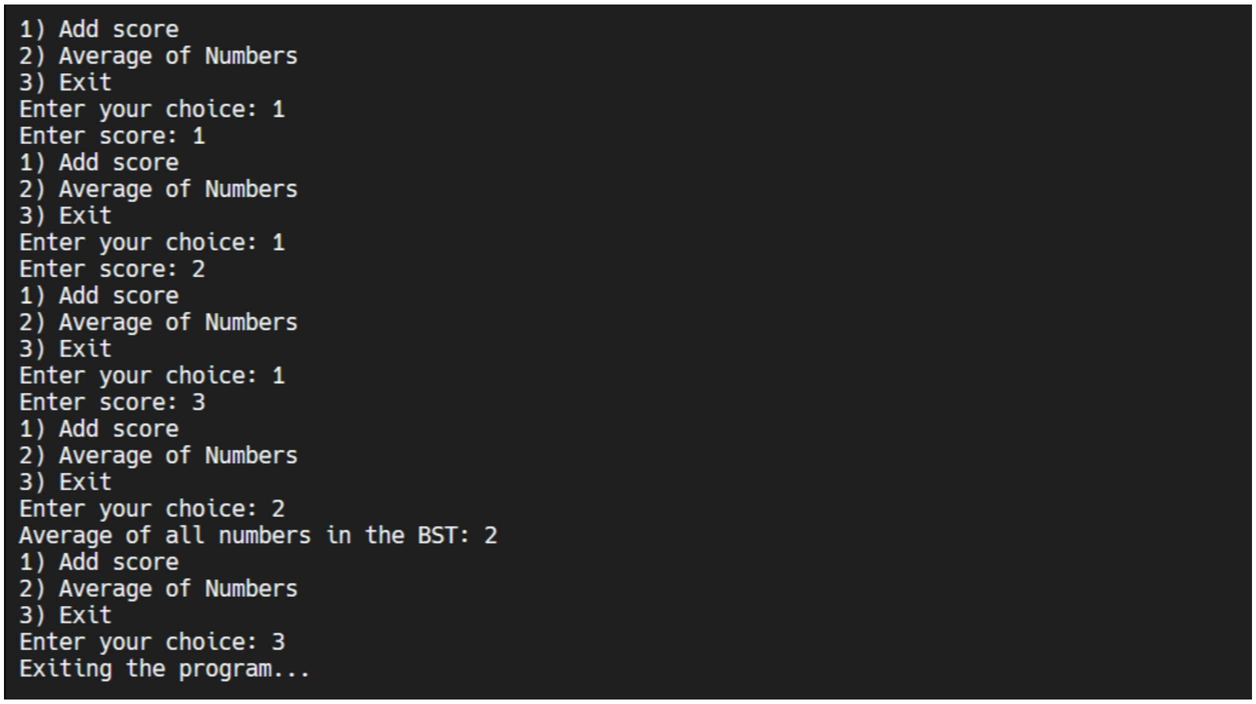


```cpp
// www.gammal.tech

#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;
}
```

7- 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.

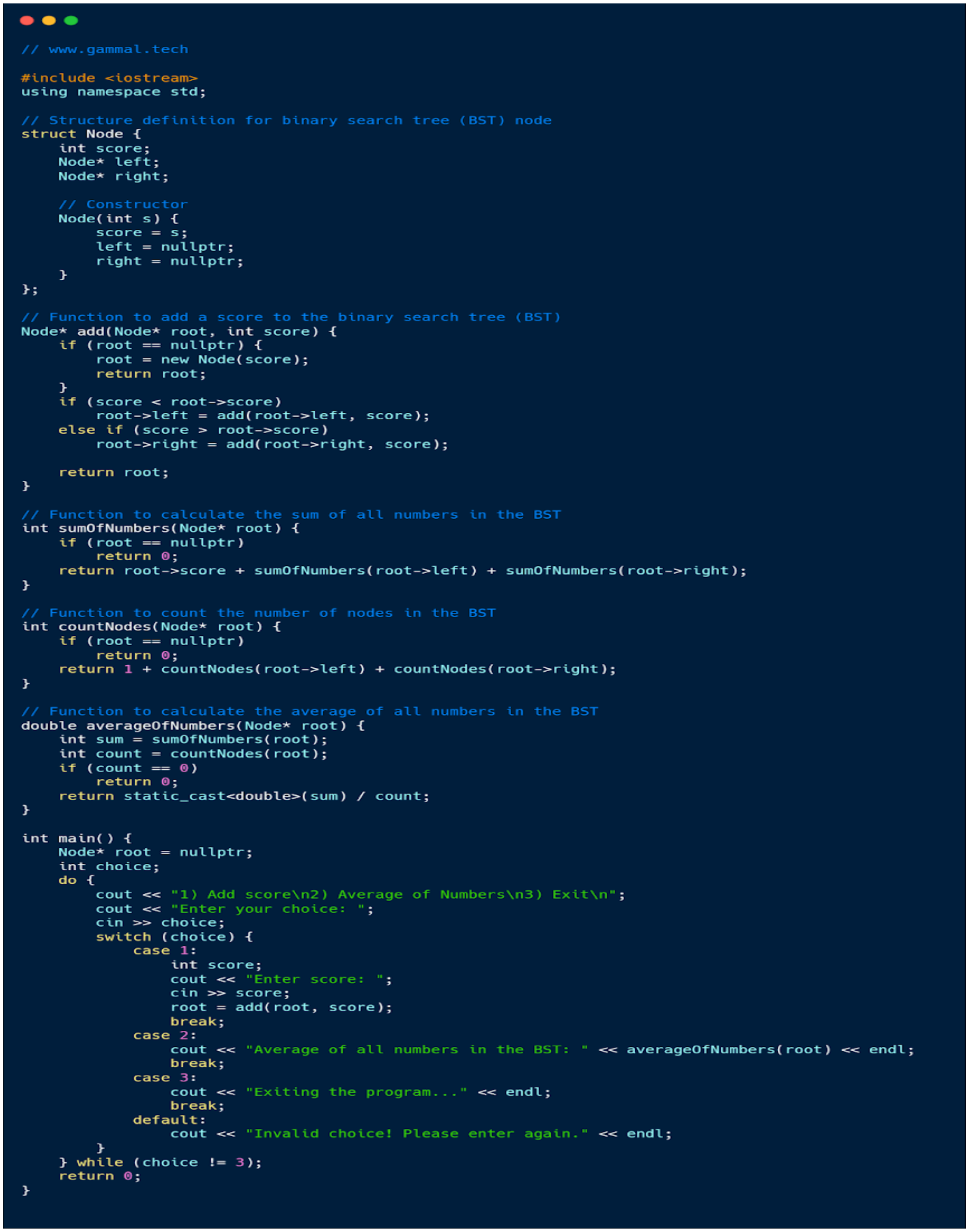
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، ثم ابحث عن متوسط جميع الأرقام في شجرة البحث الثنائية.

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

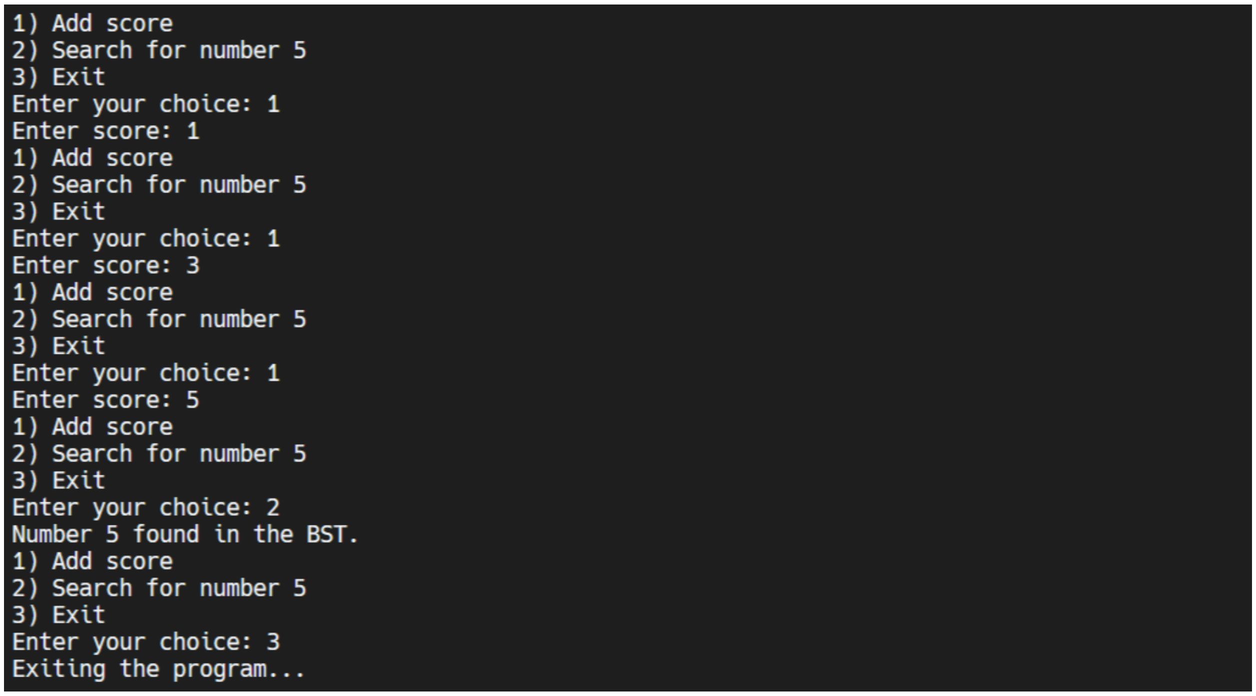


```
// www.gammal.tech

#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;
}
```

8- 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.

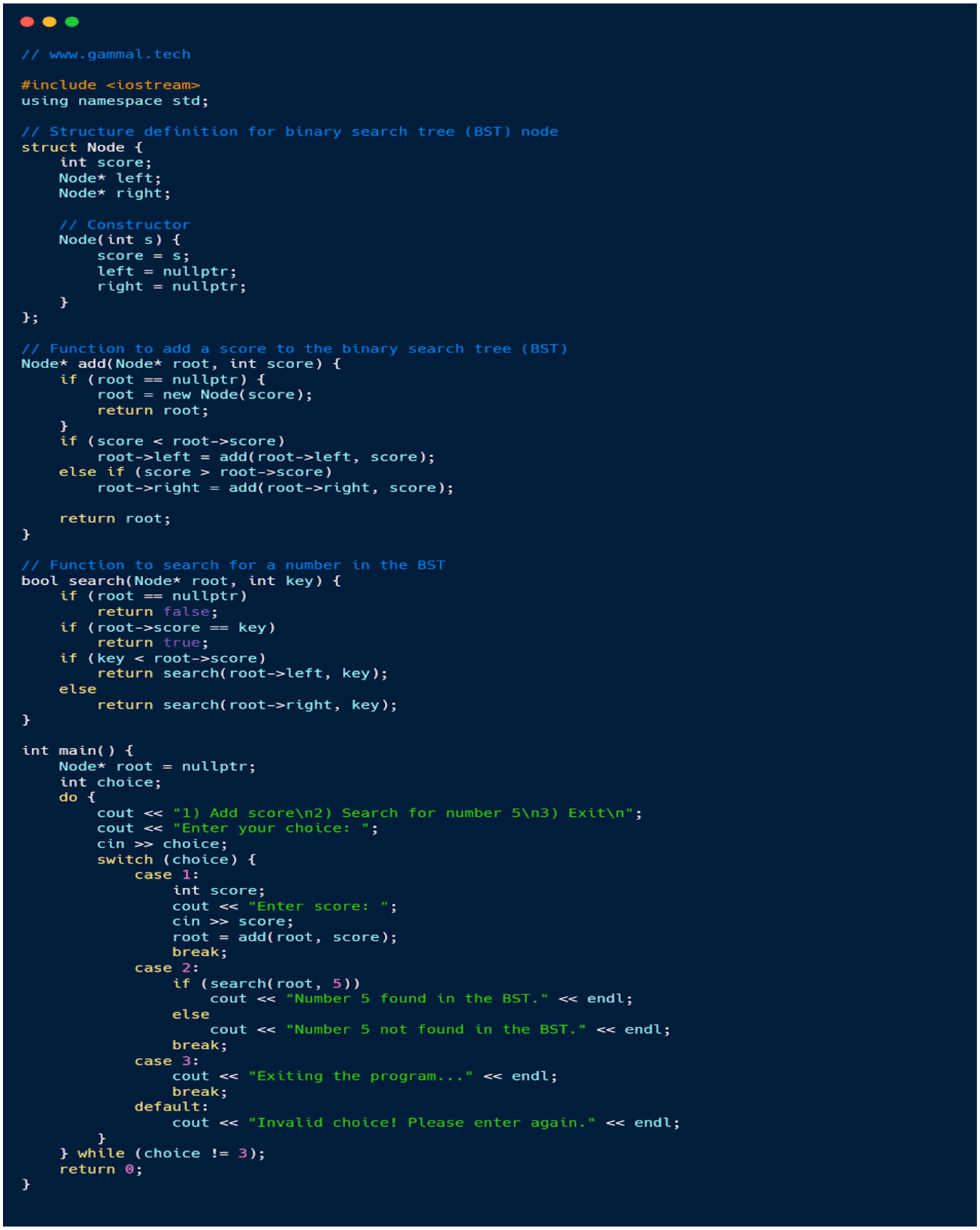
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) بناءً على درجات إدخال المستخدم ثم تحقق مما إذا كان الرقم 5 موجودًا في شجرة البحث الثنائية أم لا.

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

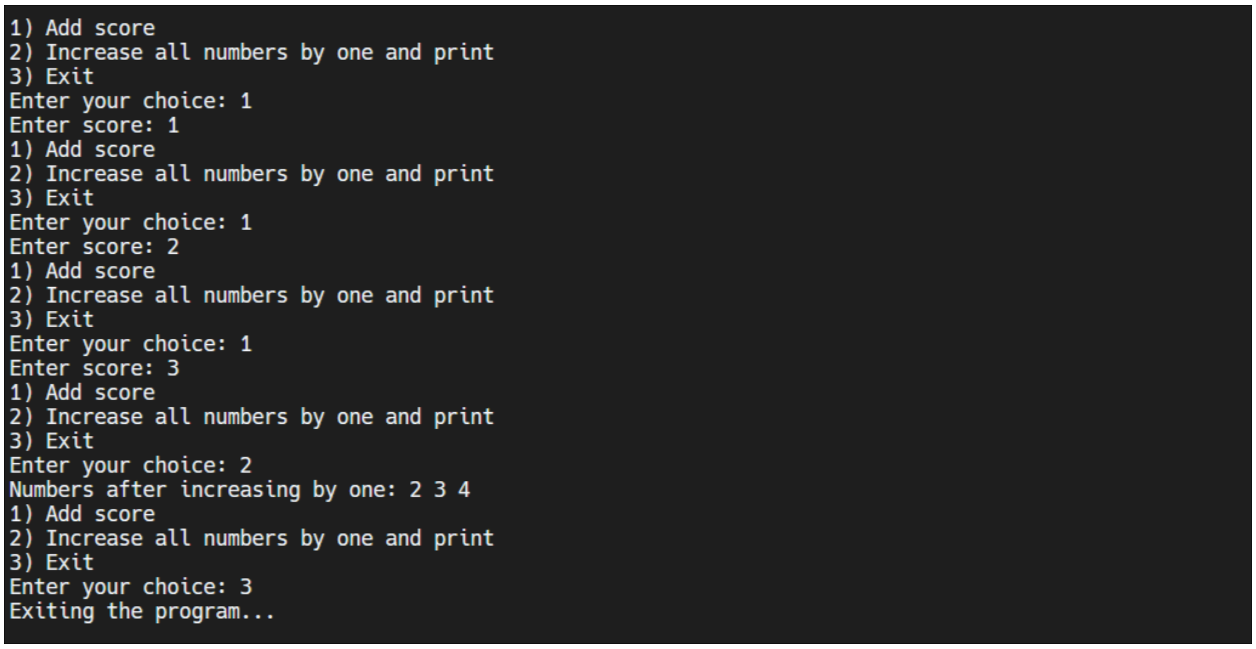


```cpp
// www.gammal.tech

#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;
}
```

9- 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.

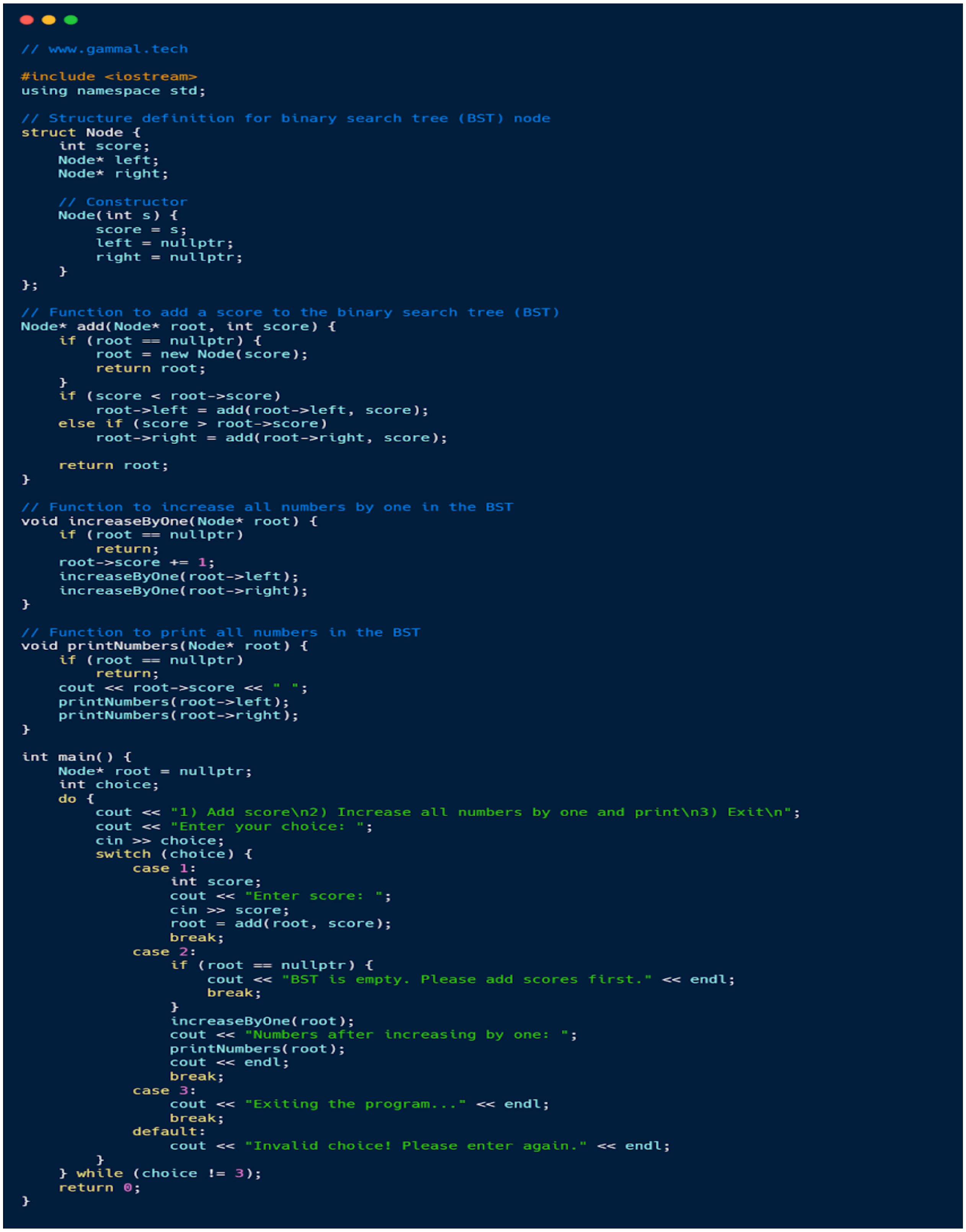
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، وقم بزيادة جميع الأرقام بمقدار واحد، ثم قم بطباعة جميع الأرقام الموجودة في BST.

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

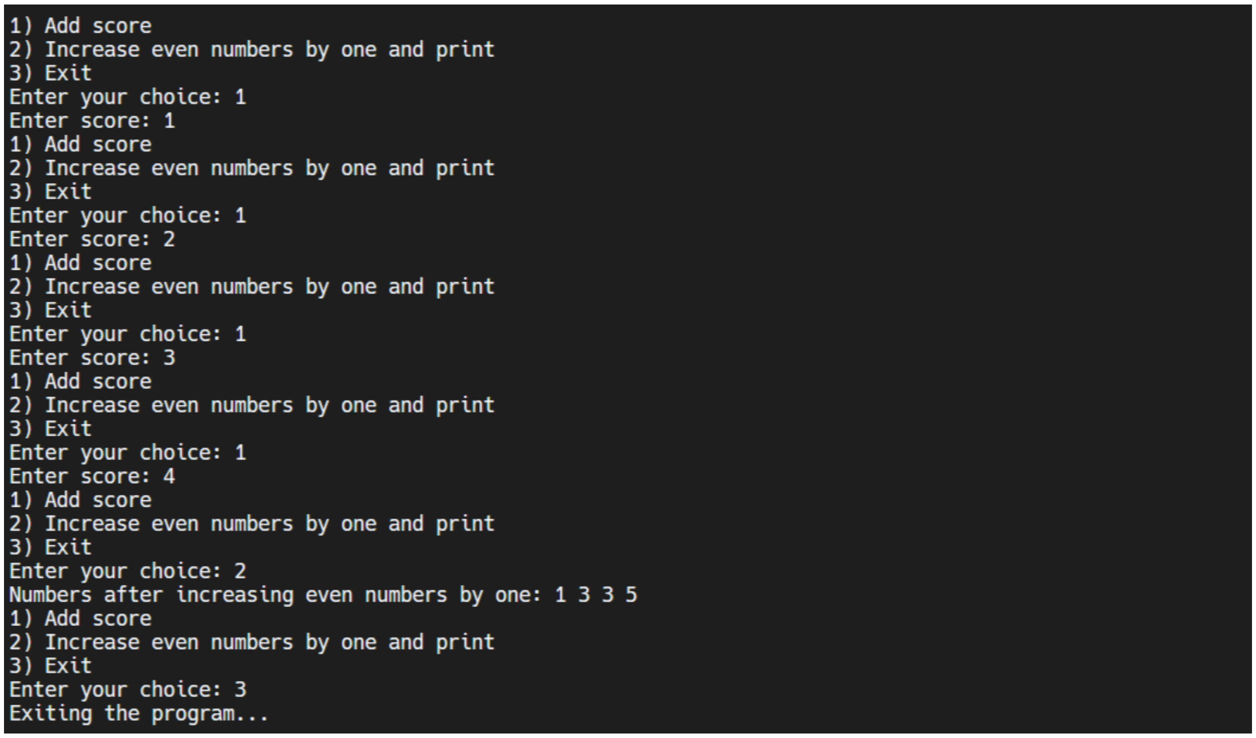


```cpp
// www.gammal.tech

#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;
}
```

10- 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.

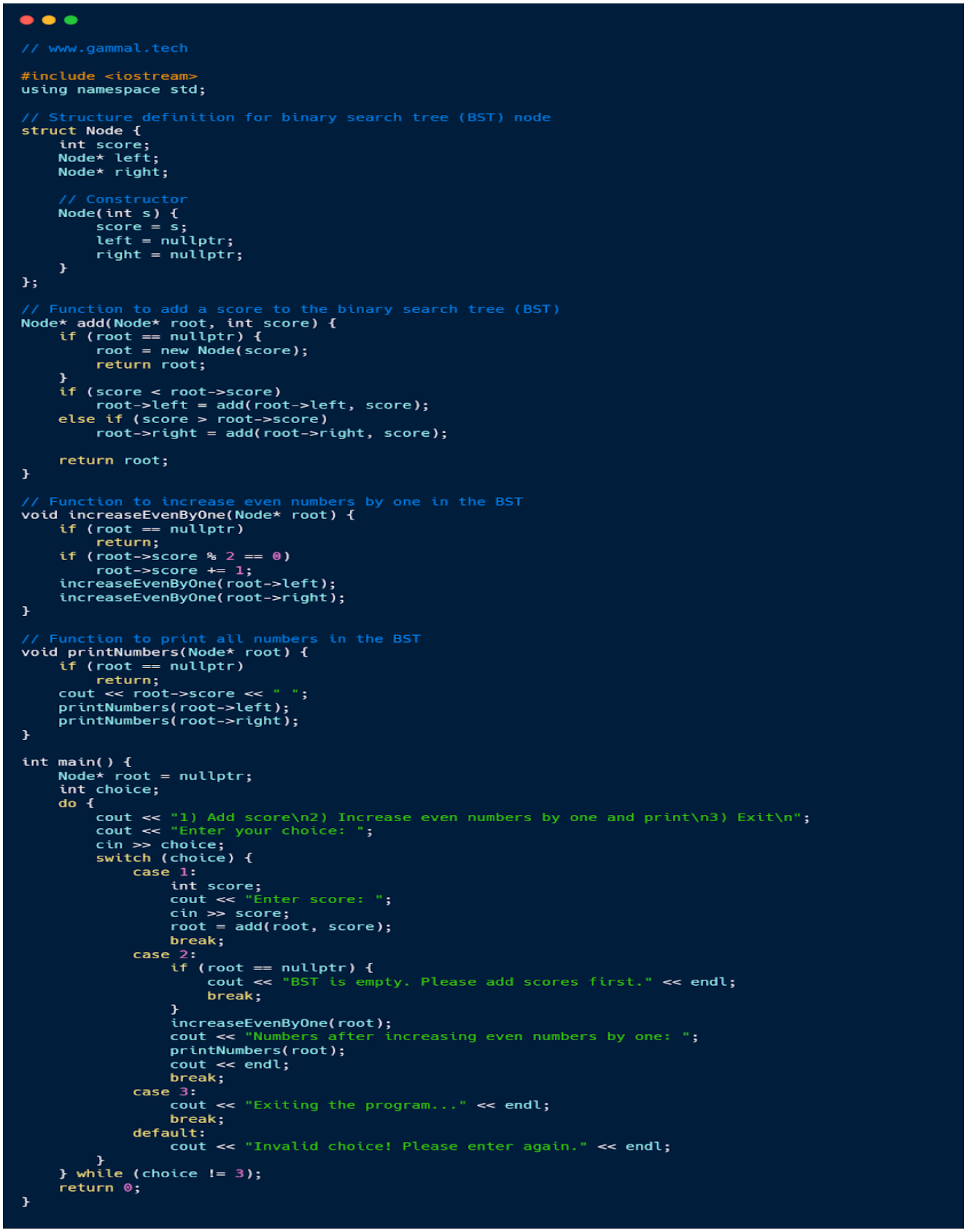
اكتب برنامج C++ لإنشاء شجرة بحث ثنائية (BST) استنادًا إلى درجات إدخال المستخدم، وقم بزيادة الأرقام الزوجية بمقدار واحد فقط، ثم اطبع جميع الأرقام الموجودة في شجرة البحث الثنائية.

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;
}
```