1. Write a function to reverse a singly linked list.The function should take the head of the list and return the new head of the reversed list.

#include <iostream>

using namespace std;

struct Node {

int val;

Node \*next;

Node(int y) : val(y), next(nullptr) {}

};

Node\* reverseList(Node\* head) {

Node \*prev = nullptr;

Node \*curr = head;

Node \*next = nullptr;

while (curr != nullptr) {

next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

return prev;

}

void printList(Node \*head) {

Node \*temp = head;

while (temp != nullptr) {

cout << temp->val << " ";

temp = temp->next;

}

cout << endl;

}

Node\* createLinkedList(int arr[], int n) {

if (n == 0)

return nullptr;

Node \*head = new Node(arr[0]);

Node \*curr = head;

for (int i = 1; i < n; ++i) {

curr->next = new Node(arr[i]);

curr = curr->next;

}

return head;

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

int n = sizeof(arr) / sizeof(arr[0]);

Node \*head = createLinkedList(arr, n);

cout << "Original list: ";

printList(head);

Node \*reversedHead = reverseList(head);

cout << "Reversed list: ";

printList(reversedHead);

Node \*temp;

while (reversedHead != nullptr) {

temp = reversedHead;

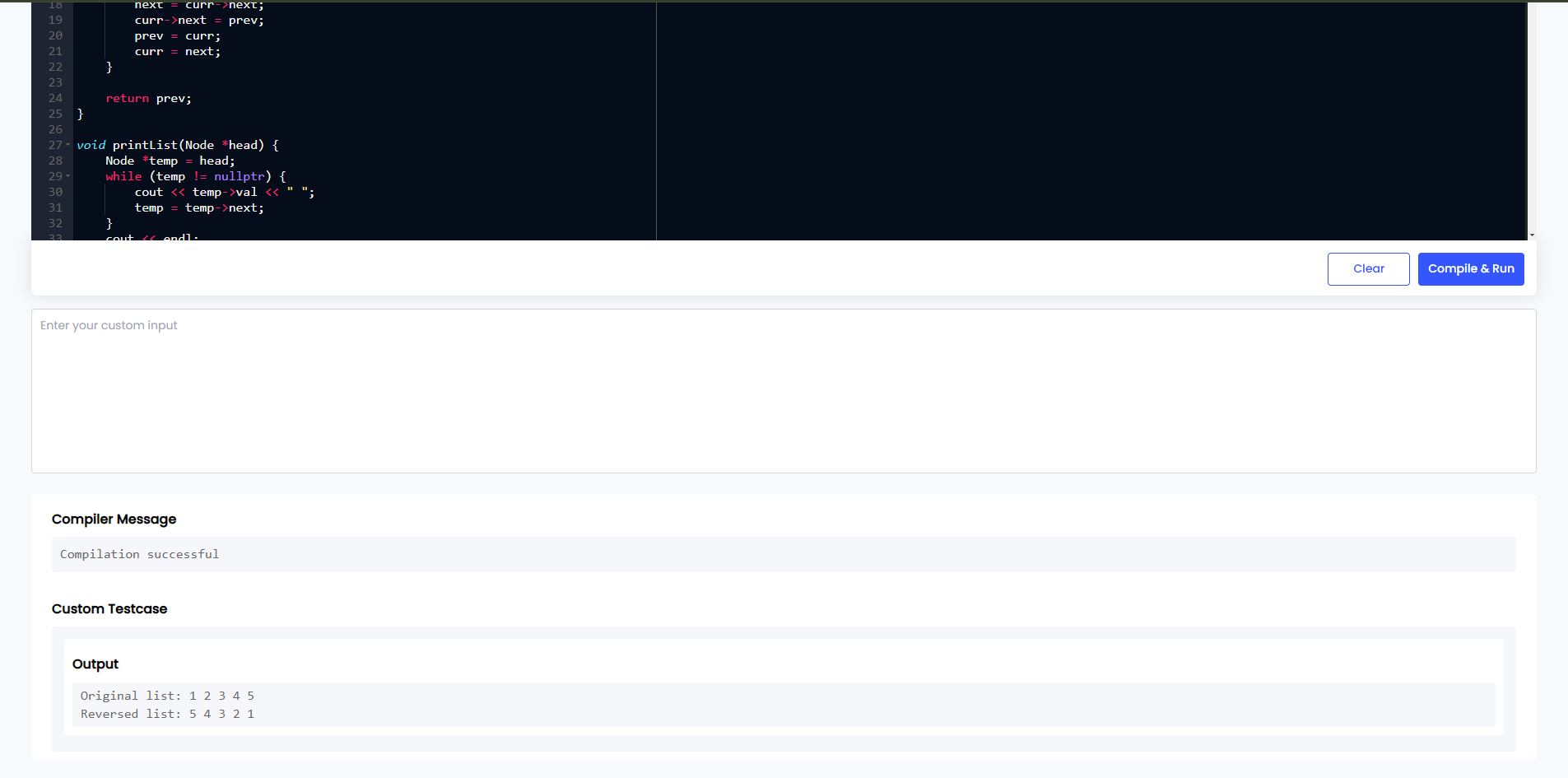
reversedHead = reversedHead->next;

delete temp;

}

return 0;

}



1. Given a string, find the length of the longest substring without repeating characters.The function should return an integer representing the length of the longest substring without repeating characters.

#include <iostream>

#include <unordered\_map>

#include <algorithm>

using namespace std;

int lols(string s) {

unordered\_map<char, int> charMap;

int maxLength = 0;

int left = 0;

for (int right = 0; right < s.length(); ++right) {

char currentChar = s[right];

if (charMap.find(currentChar) != charMap.end() && charMap[currentChar] >= left) {

left = charMap[currentChar] + 1;

}

charMap[currentChar] = right;

maxLength = max(maxLength, right - left + 1);

}

return maxLength;

}

int main() {

string s1 = "bacaxcb";

string s2 = "bbb";

string s3 = "pikachu";

string s4 = "";

cout <<s1 << "\" is: " << lols(s1) << endl;

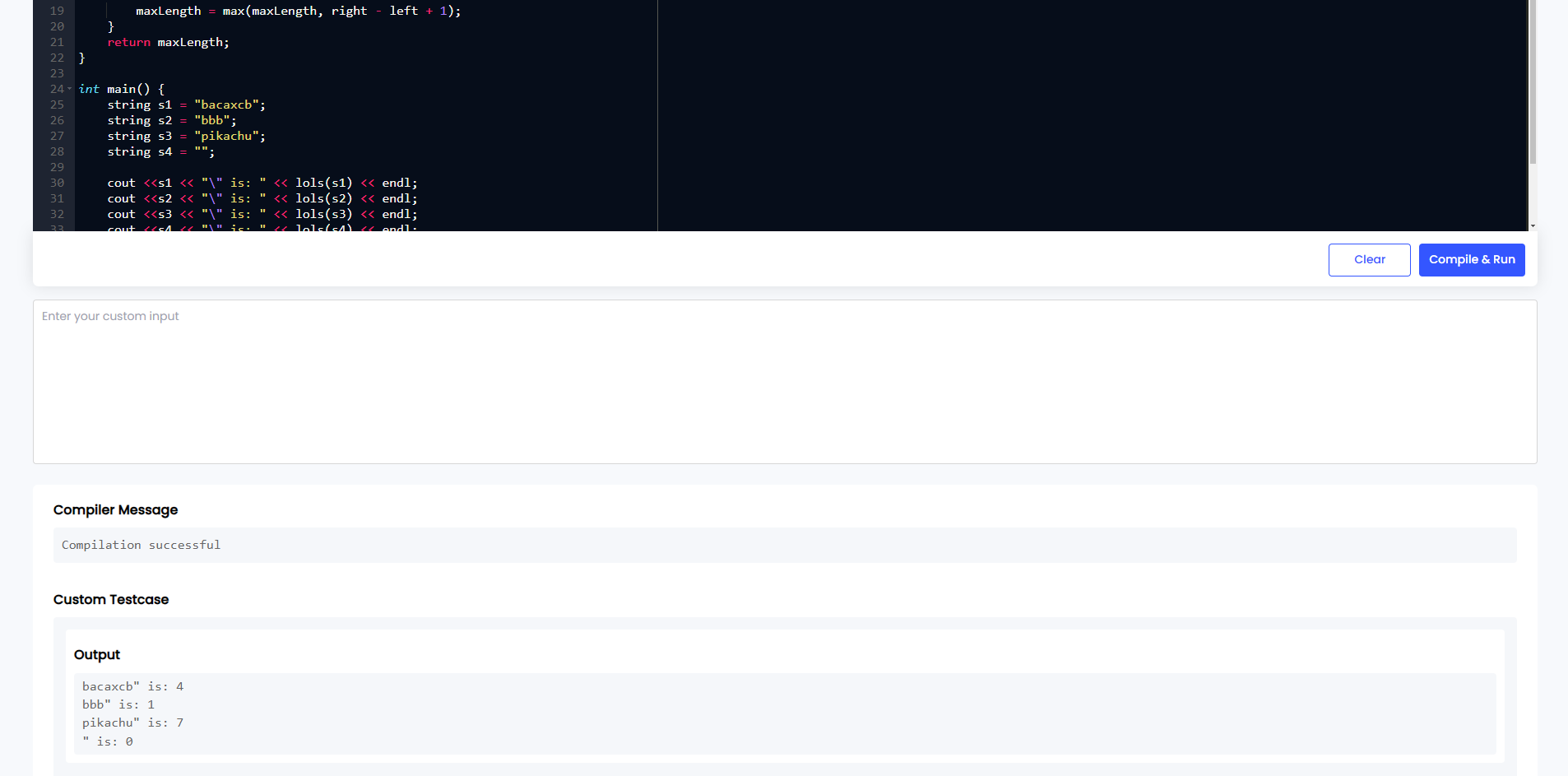
cout <<s2 << "\" is: " << lols(s2) << endl;

cout <<s3 << "\" is: " << lols(s3) << endl;

cout <<s4 << "\" is: " << lols(s4) << endl;

return 0;

}



1. Given a non-empty binary tree, find the maximum path sum. A path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain at least one node and does not need to go through the root.The function should return an integer representing the maximum path sum.

#include <iostream>

#include <algorithm>

#include <climits>

using namespace std;

struct Node {

int val;

Node \*left;

Node \*right;

Node(int x) : val(x), left(nullptr), right(nullptr) {}

};

int maxSum = INT\_MIN;

int maxPath(Node\* node) {

if (node == nullptr)

return 0;

int leftMax = max(0, maxPath(node->left));

int rightMax = max(0, maxPath(node->right));

int nodeMax = node->val + leftMax + rightMax;

maxSum = max(maxSum, nodeMax);

return node->val + max(leftMax, rightMax);

}

int maxPathSum(Node\* root) {

maxSum = INT\_MIN;

maxPath(root);

return maxSum;

}

Node\* BinaryTree(int arr[], int n, int i) {

Node\* node = nullptr;

if (i < n) {

node = new Node(arr[i]);

node->left = BinaryTree(arr, n, 2 \* i + 1);

node->right = BinaryTree(arr, n, 2 \* i + 2);

}

return node;

}

int main() {

int arr[] = {2, 6, 8};

int n = sizeof(arr) / sizeof(arr[0]);

Node\* root = BinaryTree(arr, n, 0);

if (root == nullptr) {

cout << "Maximum path sum: 0" << endl;

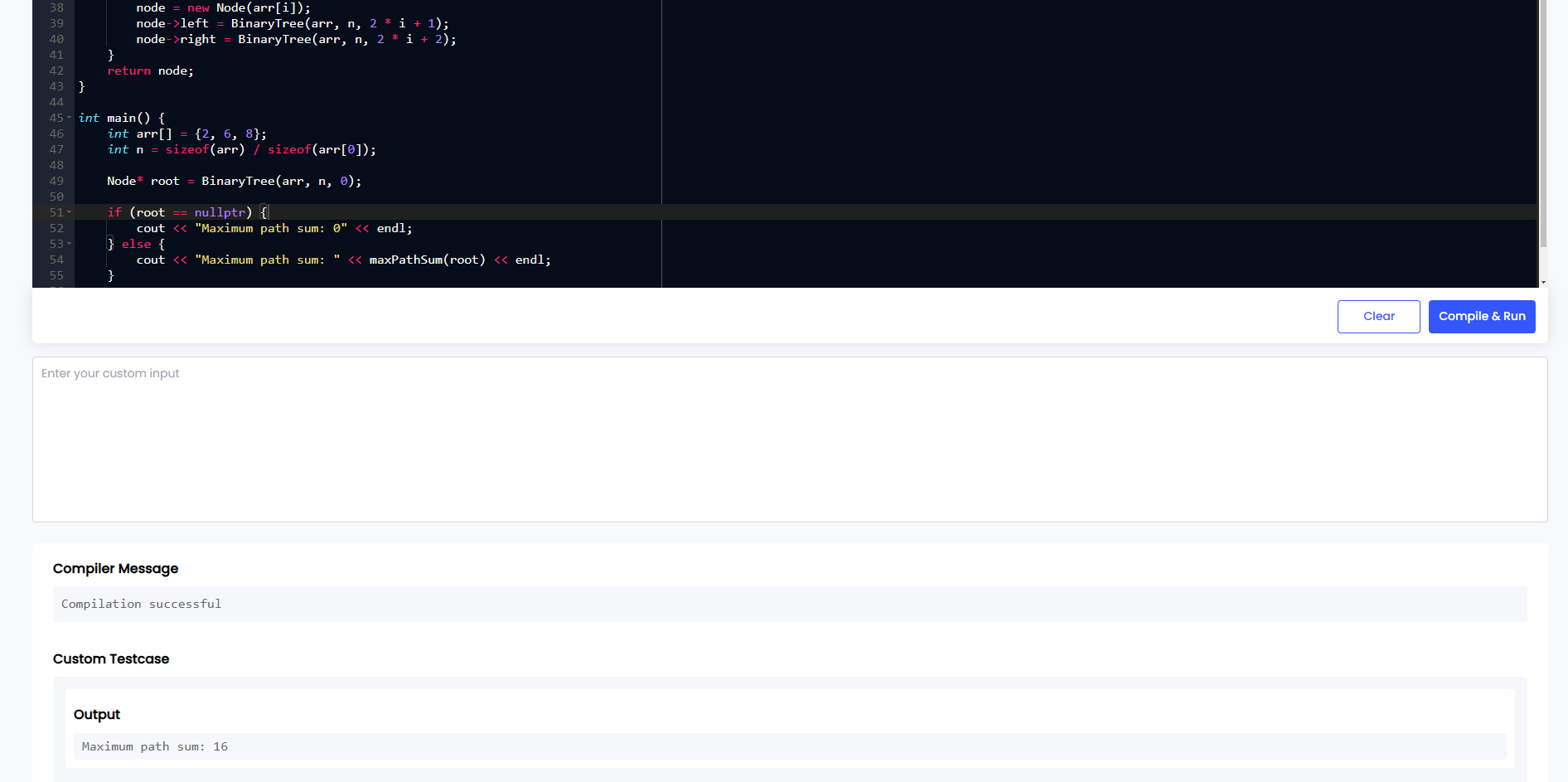
} else {

cout << "Maximum path sum: " << maxPathSum(root) << endl;

}

return 0;

}



4.Design an algorithm to serialize and deserialize a binary tree. Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment. Implement the serialize and deserialize methods.

#include <iostream>

#include <sstream>

#include <string>

#include <queue>

#include <vector>

using namespace std;

struct Node {

int val;

Node \*left;

Node \*right;

Node(int x) : val(x), left(nullptr), right(nullptr) {}

};

class Codec {

public:

string serialize(Node\* root) {

if (!root) return "";

stringstream ss;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* node = q.front();

q.pop();

if (node) {

ss << node->val << " ";

q.push(node->left);

q.push(node->right);

} else {

ss << "# ";

}

}

return ss.str();

}

Node\* deserialize(const string& data) {

if (data.empty()) return nullptr;

stringstream ss(data);

string val;

ss >> val;

Node\* root = new Node(stoi(val));

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* node = q.front();

q.pop();

if (ss >> val) {

if (val != "#") {

node->left = new Node(stoi(val));

q.push(node->left);

}

}

if (ss >> val) {

if (val != "#") {

node->right = new Node(stoi(val));

q.push(node->right);

}

}

}

return root;

}

};

void printTree(Node\* root) {

if (!root) return;

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

Node\* node = q.front();

q.pop();

if (node) {

cout << node->val << " ";

q.push(node->left);

q.push(node->right);

} else {

cout << "# ";

}

}

cout << endl;

}

int main() {

Node\* root = new Node(1);

root->left = new Node(2);

root->right = new Node(3);

root->right->left = new Node(4);

root->right->right = new Node(5);

Codec codec;

string serializedTree = codec.serialize(root);

cout << "Serialized tree: " << serializedTree << endl;

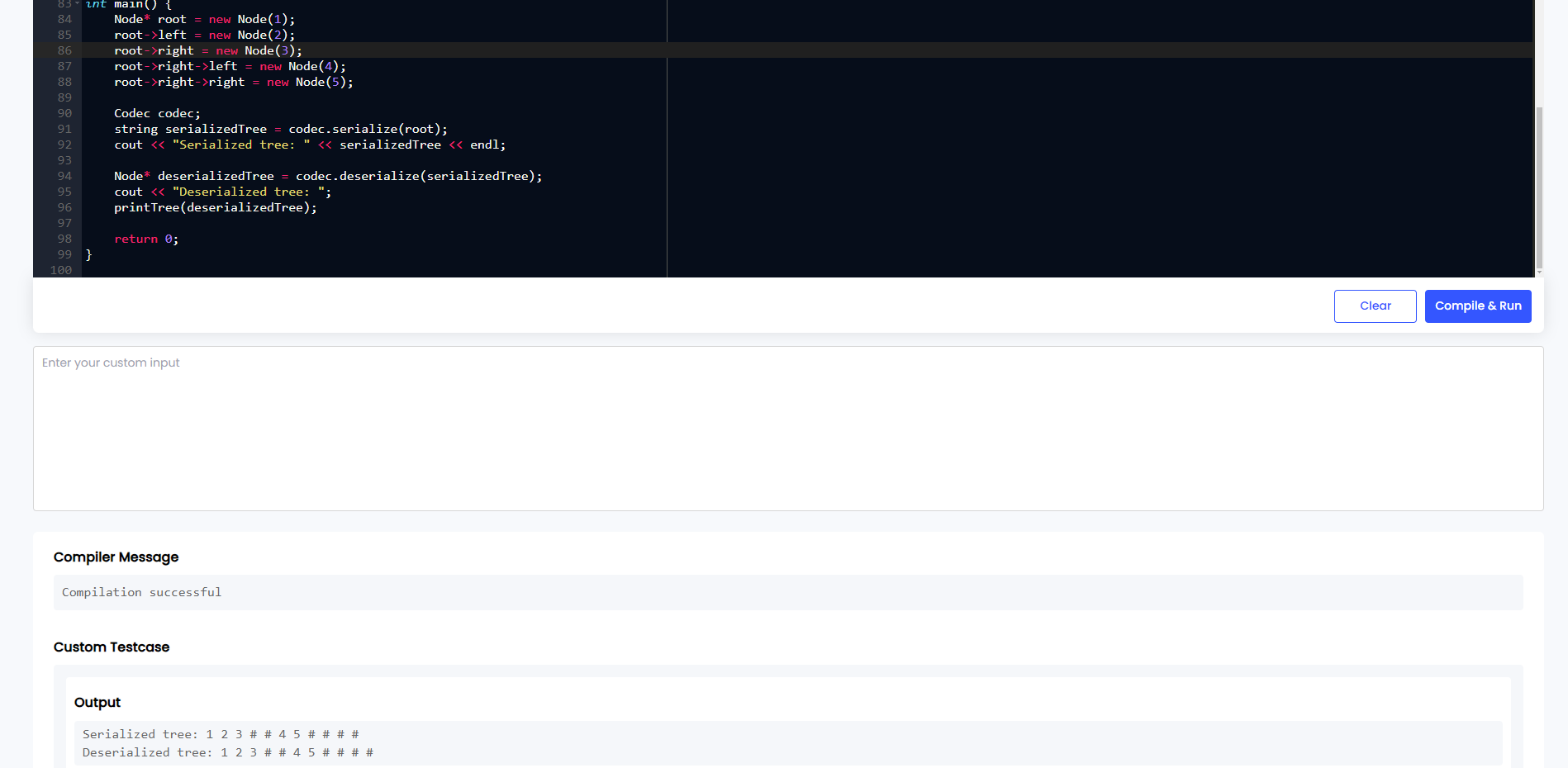
Node\* deserializedTree = codec.deserialize(serializedTree);

cout << "Deserialized tree: ";

printTree(deserializedTree);

return 0;

}



5.Write a function to rotate an array to the right by k steps.The function should modify the array in place to achieve the rotation.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void rotate(vector<int>& nums, int k) {

int n = nums.size();

k = k % n;

reverse(nums.begin(), nums.end());

reverse(nums.begin(), nums.begin() + k);

reverse(nums.begin() + k, nums.end());

}

void Array(const vector<int>& nums) {

for (int num : nums) {

cout << num << " ";

}

cout << endl;

}

int main() {

vector<int> nums = {1, 2, 3, 4, 5, 6, 7};

int k = 6;

cout << "Original array: ";

Array(nums);

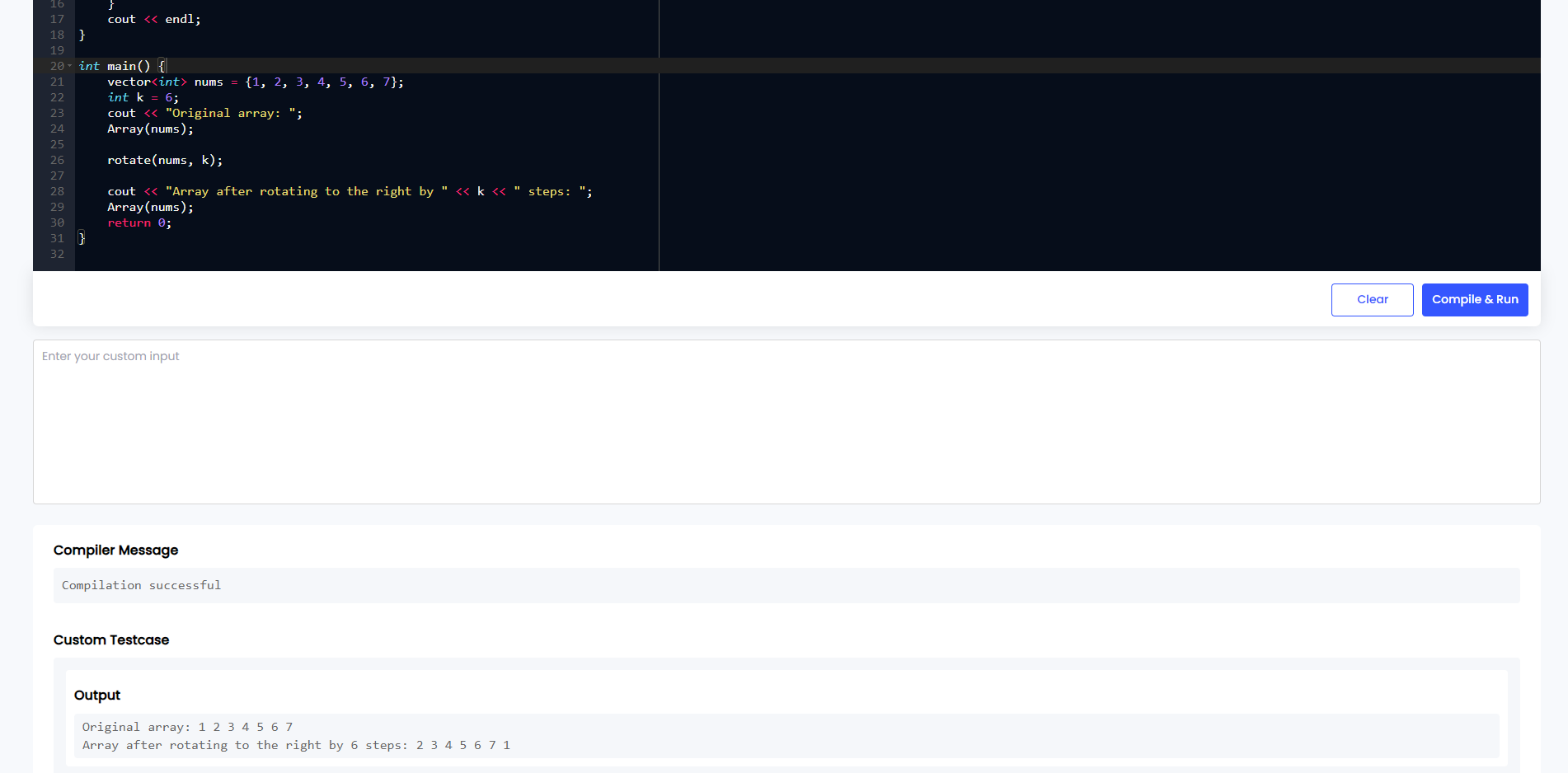
rotate(nums, k);

cout << "Array after rotating to the right by " << k << " steps: ";

Array(nums);

return 0;

}



6.Write a function to find the factorial of a given number.The function should return the factorial of the number.

#include <iostream>

using namespace std;

long long factorial(int n) {

if (n < 0) {

throw invalid\_argument("Factorial is not defined");

}

long long result = 1;

for (int i = 1; i <= n; ++i) {

result \*= i;

}

return result;

}

int main() {

int num = 5;

try {

long long result = factorial(num);

cout << "Factorial of " << num << " is " << result << endl;

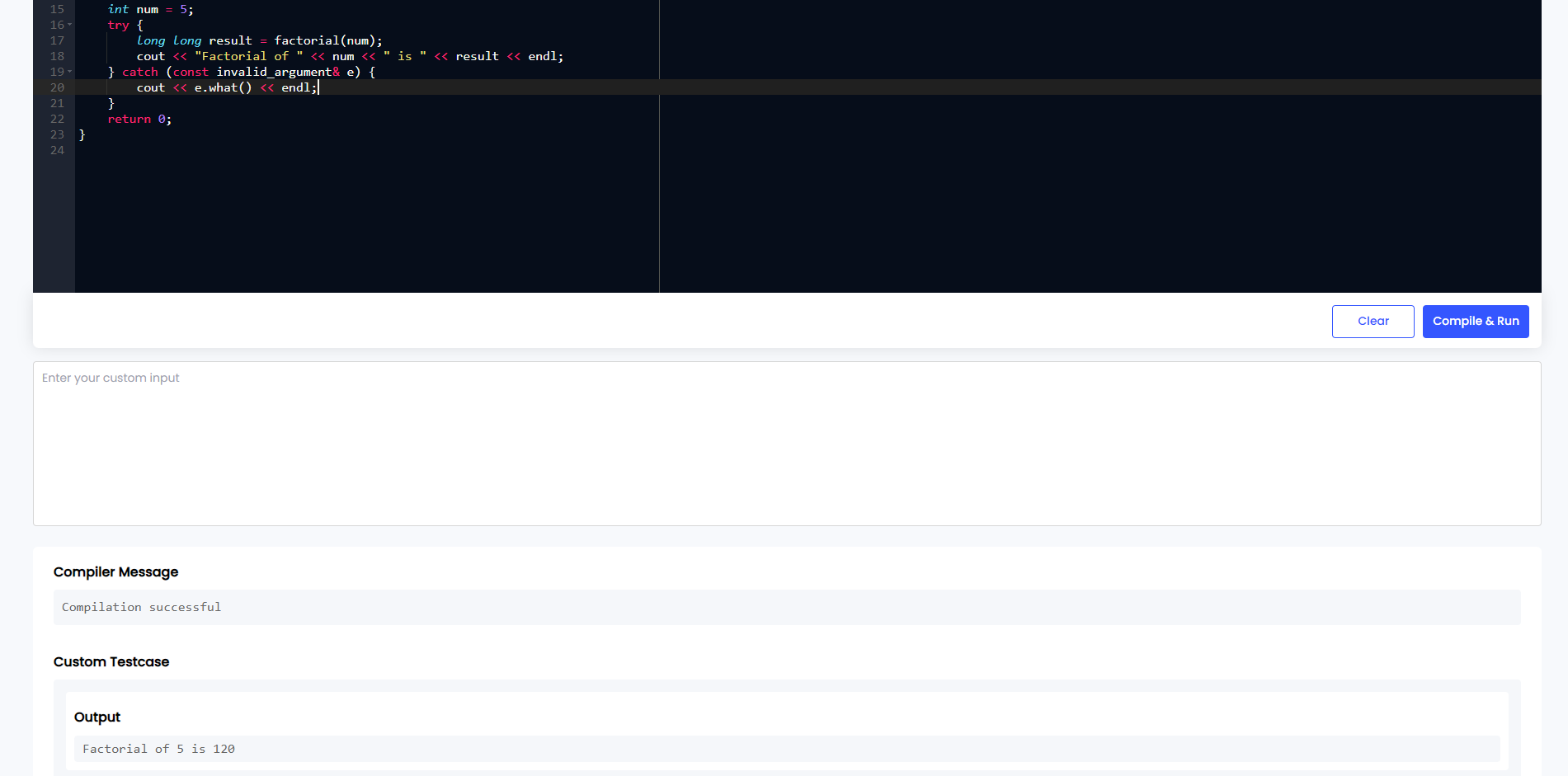
} catch (const invalid\_argument& e) {

cout << e.what() << endl;

}

return 0;

}



7.Write a function to compute the sum of the digits of a given number.The function should return the sum of the digits of the number.

#include <iostream>

using namespace std;

int sum(int num) {

if (num < 0) {

num = -num;

}

int sum = 0;

while (num > 0) {

sum += num % 10;

num /= 10;

}

return sum;

}

int main() {

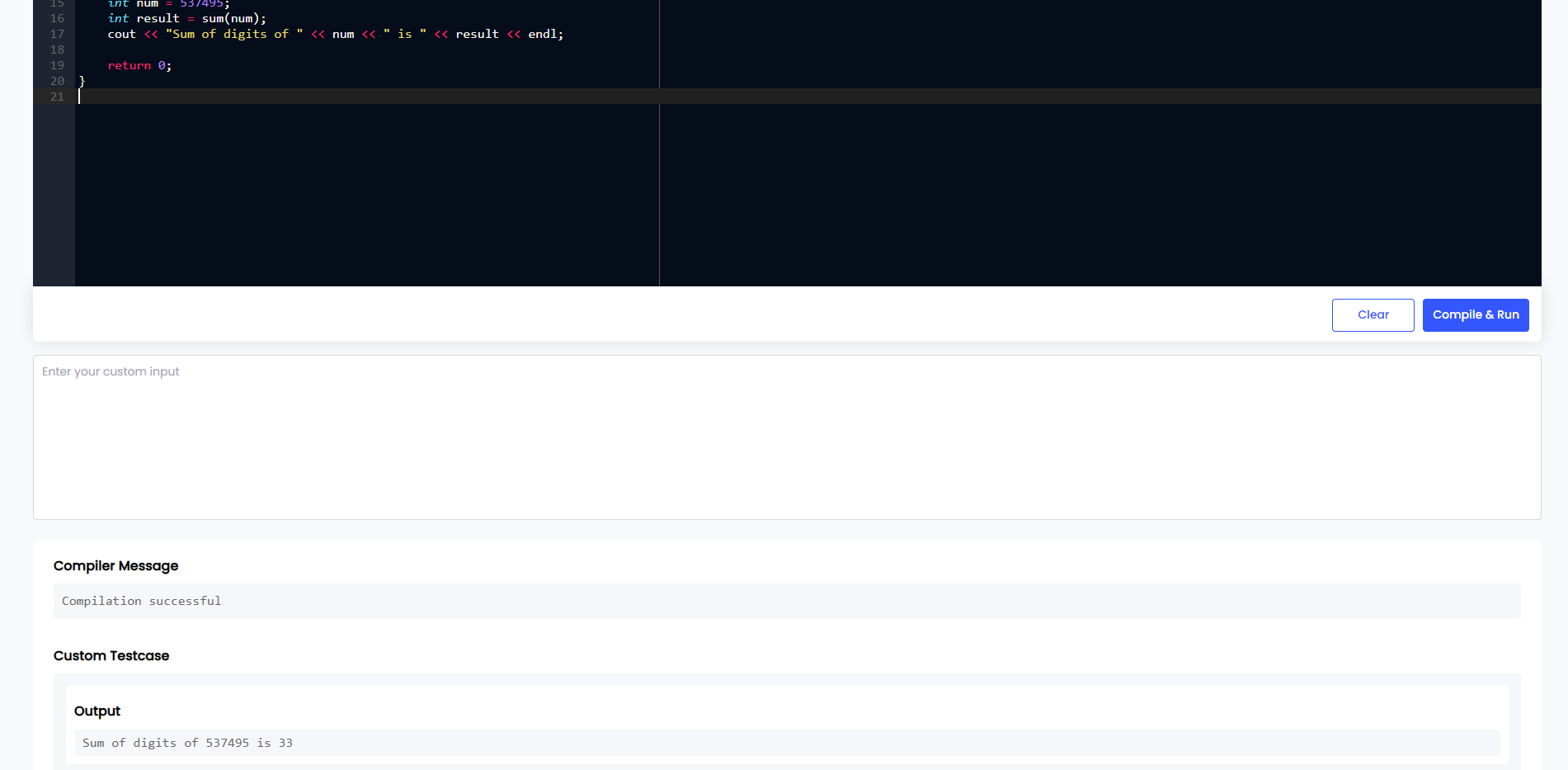
int num = 537495;

int result = sum(num);

cout << "Sum of digits of " << num << " is " << result << endl;

return 0;

}



8.Write a function to find the greatest common divisor (GCD) of two numbers. The function should return the GCD of a and b.

#include <iostream>

using namespace std;

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int main() {

int a = 89;

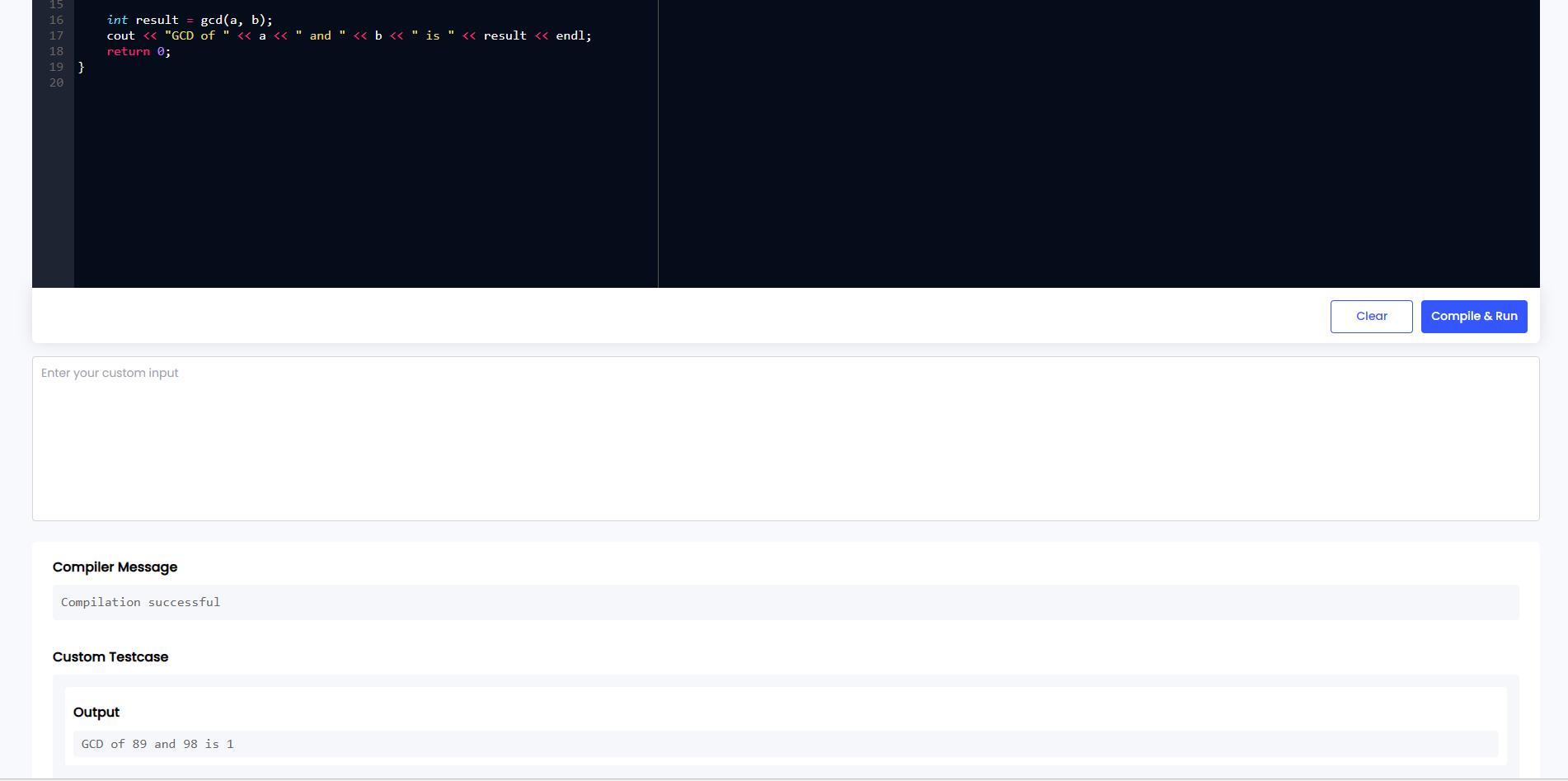
int b = 98;

int result = gcd(a, b);

cout << "GCD of " << a << " and " << b << " is " << result << endl;

return 0;

}



9.Write a function to find the maximum difference between any two elements in an array.The function should return the maximum difference between any two elements in the array.

#include <iostream>

#include <vector>

#include <algorithm>

#include <limits>

using namespace std;

int max(const vector<int>& nums) {

int minElement = numeric\_limits<int>::max();

int maxDiff = numeric\_limits<int>::min();

for (int num : nums) {

minElement = min(minElement, num);

maxDiff = max(maxDiff, num - minElement);

}

return maxDiff;

}

int main() {

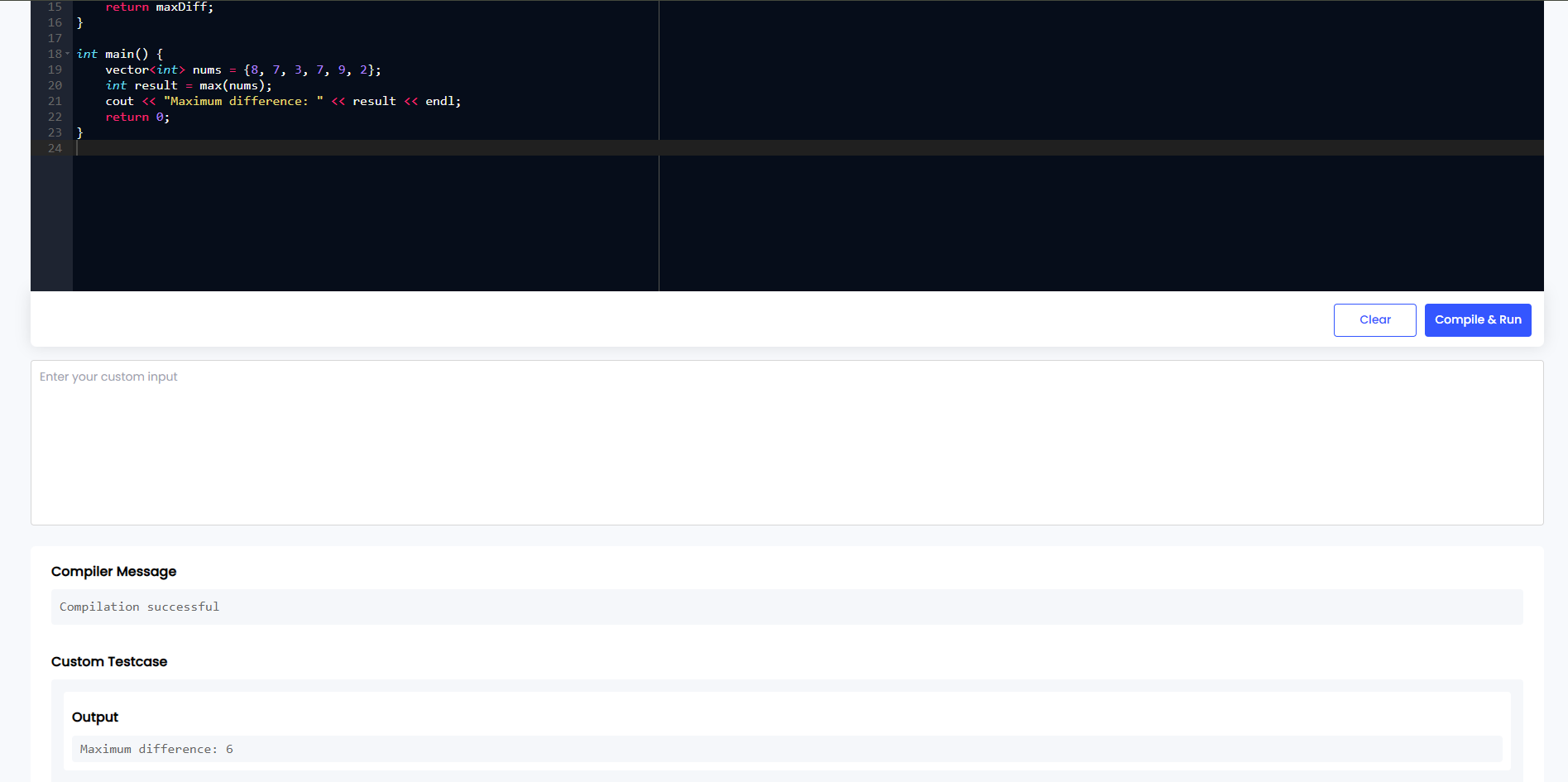
vector<int> nums = {8, 7, 3, 7, 9, 2};

int result = max(nums);

cout << "Maximum difference: " << result << endl;

return 0;

}



10.Write a function to check if a given string contains only alphabetic characters.The function should return true if the string contains only alphabetic characters, and false otherwise.

#include <iostream>

#include <string>

using namespace std;

bool coa(const string& str) {

for (char ch : str) {

if (!isalpha(ch)) {

return false;

}

}

return true;

}

int main() {

string str1 = "GlobalTrends";

string str2 = "Global123";

cout << str1 << " contains only alphabetic characters: "

<< (coa(str1) ? "true" : "false") << endl;

cout << str2 << " contains only alphabetic characters: "

<< (coa(str2) ? "true" : "false") << endl;

return 0;

}

