**Q1 . 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.**

**Source Code :**

#include <iostream>

struct ListNode {

int val;

ListNode\* next;

ListNode(int x) : val(x), next(nullptr) {}

};

ListNode\* reverseList(ListNode\* head) {

ListNode\* prev = nullptr;

ListNode\* curr = head;

ListNode\* next = nullptr;

while (curr != nullptr) {

next = curr->next; // Save next

curr->next = prev; // Reverse current node's pointer

prev = curr; // Move pointers one position ahead.

curr = next;

}

return prev;

}

void printList(ListNode\* head) {

ListNode\* curr = head;

while (curr != nullptr) {

std::cout << curr->val << " ";

curr = curr->next;

}

std::cout << std::endl;

}

int main() {

ListNode\* head = new ListNode(1);

head->next = new ListNode(2);

head->next->next = new ListNode(3);

head->next->next->next = new ListNode(4);

head->next->next->next->next = new ListNode(5);

std::cout << "Original list: ";

printList(head);

ListNode\* newHead = reverseList(head);

std::cout << "Reversed list: ";

printList(newHead);

while (newHead != nullptr) {

ListNode\* temp = newHead;

newHead = newHead->next;

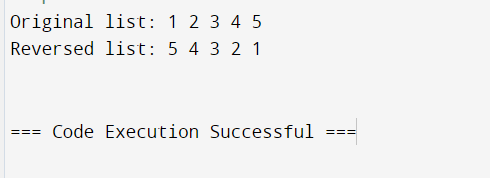
delete temp;

}

return 0;

}

**Output :**



**Q2 . 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.**

**Source Code :**

#include <iostream>

#include <unordered\_set>

#include <string>

using namespace std;

int lengthOfLongestSubstring(string s) {

int n = s.length();

int maxLen = 0;

int left = 0;

unordered\_set<char> seen;

for (int right = 0; right < n; right++) {

char c = s[right];

while (seen.count(c)) {

seen.erase(s[left]);

left++;

}

seen.insert(c);

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

}

return maxLen;

}

int main() {

string input;

cout << "Enter a string: ";

cin >> input;

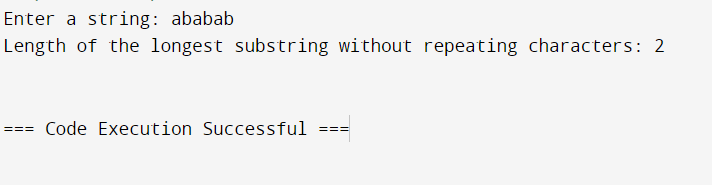
int maxLength = lengthOfLongestSubstring(input);

cout << "Length of the longest substring without repeating characters: " << maxLength << endl;

return 0;

}

**Output :**

****

**Q3 . 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.**

**Source Code**

#include <iostream>

#include <climits>

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

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

};

class Solution {

public:

int maxPathSum(TreeNode\* root) {

int maxSum = INT\_MIN;

maxGain(root, maxSum);

return maxSum;

}

private:

int maxGain(TreeNode\* node, int& maxSum) {

if (node == nullptr) {

return 0;

}

int leftGain = std::max(maxGain(node->left, maxSum), 0);

int rightGain = std::max(maxGain(node->right, maxSum), 0);

int currentPathSum = node->val + leftGain + rightGain;

maxSum = std::max(maxSum, currentPathSum);

return node->val + std::max(leftGain, rightGain);

}

};

TreeNode\* newNode(int data) {

TreeNode\* node = new TreeNode(data);

return node;

}

int main() {

TreeNode\* root = newNode(-10);

root->left = newNode(9);

root->right = newNode(20);

root->right->left = newNode(15);

root->right->right = newNode(7);

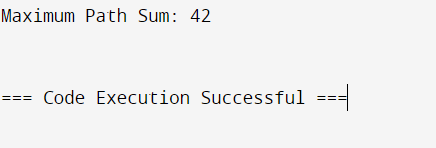
Solution sol;

std::cout << "Maximum Path Sum: " << sol.maxPathSum(root) << std::endl;

return 0;

}

**Output :**



**Q4 . 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.**

**Source Code :**

#include <iostream>

#include <string>

#include <queue>

#include <sstream>

using namespace std;

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(NULL), right(NULL) {}

};

class Codec {

public:

string serialize(TreeNode\* root) {

return serializeHelper(root);

}

TreeNode\* deserialize(string data) {

queue<string> nodes;

string node;

stringstream ss(data);

while (getline(ss, node, ',')) {

nodes.push(node);

}

return deserializeHelper(nodes);

}

private:

string serializeHelper(TreeNode\* root) {

if (root == NULL) {

return "#,";

}

return to\_string(root->val) + "," + serializeHelper(root->left) + serializeHelper(root->right);

}

TreeNode\* deserializeHelper(queue<string>& nodes) {

string node = nodes.front();

nodes.pop();

if (node == "#") {

return NULL;

}

TreeNode\* root = new TreeNode(stoi(node));

root->left = deserializeHelper(nodes);

root->right = deserializeHelper(nodes);

return root;

}

};

int main() {

Codec ser, deser;

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

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

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

string serializedTree = ser.serialize(root);

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

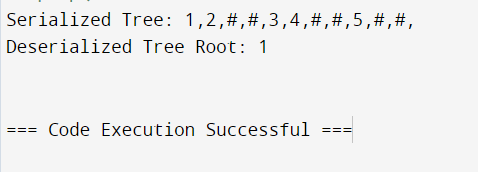
TreeNode\* deserializedTree = deser.deserialize(serializedTree);

cout << "Deserialized Tree Root: " << deserializedTree->val << endl;

return 0;

}

**Output :**



**Q5 . 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.**

**Source Code :**

#include <iostream>

#include <vector>

#include <algorithm>

void rotateArray(std::vector<int>& nums, int k) {

int n = nums.size();

k = k % n;

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

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

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

}

int main() {

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

int k = 3;

rotateArray(nums, k);

for (int num : nums) {

std::cout << num << " ";

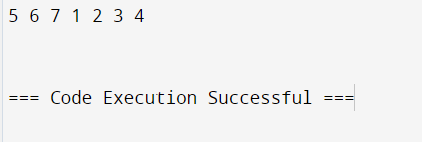
}

std::cout << std::endl;

return 0;

}

**Output :**

****

**Q6 . Write a function to find the factorial of a given number.The function should return the factorial of the number.**

**Source Code :**

#include <iostream>

unsigned long long factorial(int n) {

if (n < 0) {

return -1;

}

unsigned long long result = 1;

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

result \*= i;

}

return result;

}

int main() {

int number;

std::cout << "Enter a number: ";

std::cin >> number;

unsigned long long fact = factorial(number);

if (fact == -1) {

std::cout << "Factorial is not defined for negative numbers." << std::endl;

} else {

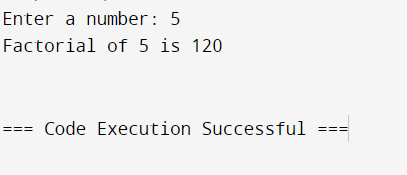
std::cout << "Factorial of " << number << " is " << fact << std::endl;

}

return 0;

}

**Output :**

****

**Q7 . 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.**

**Source Code :**

#include <iostream>

int sumOfDigits(int n) {

int sum = 0;

while (n != 0) {

sum += n % 10;

n /= 10;

}

return sum;

}

int main() {

int number;

std::cout << "Enter a number: ";

std::cin >> number;

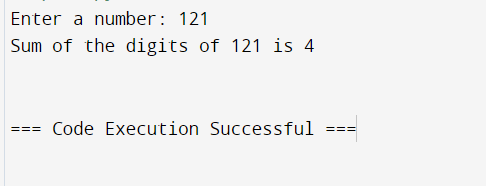
int sum = sumOfDigits(number);

std::cout << "Sum of the digits of " << number << " is " << sum << std::endl;

return 0;

}

**Output :**

****

**Q8 . Write a function to find the greatest common divisor (GCD) of two numbers. The function should return the GCD of a and b.**

**Source Code :**

#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 num1, num2;

cout << "Enter two integers: ";

cin >> num1 >> num2;

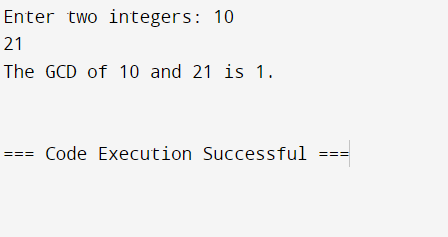
int result = gcd(num1, num2);

cout << "The GCD of " << num1 << " and " << num2 << " is " << result << "." << endl;

return 0;

}

**Output :**



**Q9 . 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.**

**Source Code :**

#include <iostream>

using namespace std;

int maxDifference(int arr[], int n) {

int maxDiff = arr[1] - arr[0];

int minElement = arr[0];

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

if (arr[i] - minElement > maxDiff) {

maxDiff = arr[i] - minElement;

}

if (arr[i] < minElement) {

minElement = arr[i];

}

}

return maxDiff;

}

int main() {

int n;

cout << "Enter the size of the array: ";

cin >> n;

int arr[n];

cout << "Enter the elements of the array: ";

for (int i = 0; i < n; i++) {

cin >> arr[i];

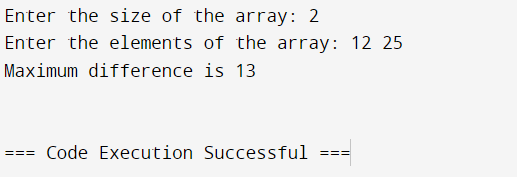
}

cout << "Maximum difference is " << maxDifference(arr, n) << endl;

return 0;

}

**Output :**

****

**Q10 .** **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.**

**Source Code :**

#include <iostream>

#include <cctype> // for isalpha function

using namespace std;

bool isAlphabetic(string str) {

for (char c : str) {

if (!isalpha(c)) {

return false;

}

}

return true;

}

int main() {

string input;

cout << "Enter a string: ";

getline(cin, input);

if (isAlphabetic(input)) {

cout << "The string contains only alphabetic characters." << endl;

} else {

cout << "The string contains non-alphabetic characters." << endl;

}

return 0;

}

**Output:**

