Assignment 9 VS CODE (Advance Programming)

Name – Paras Aggarwal

UID – 22BCS16963

Set Matrix Zeroes

Given an m x n matrix, if an element is 0, set its entire row and column to 0.

Solution:

```
#include <iostream>
#include <vector>
using namespace std;

class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {
        int m = matrix.size();
        int n = matrix[0].size();
        bool firstRow = false, firstCol = false;

        // Check if first column has any zeros
        for (int i = 0; i < m; i++) {
            if (matrix[i][0] == 0) firstCol = true;
        }
```

```
// Check if first row has any zeros
for (int j = 0; j < n; j++) {
    if (matrix[0][j] == 0) firstRow = true;
}

// Use first row and column as markers
for (int i = 1; i < m; i++) {
    for (int j = 1; j < n; j++) {
        if (matrix[i][j] == 0) {
            matrix[i][0] = 0;
            matrix[0][j] = 0;
        }
    }
}

// Set cells to zero based on markers
for (int i = 1; i < m; i++) {
    for (int j = 1; j < n; j++) {
        if (matrix[i][0] == 0 || matrix[0][j] == 0) {
            matrix[i][j] = 0;
        }
    }
}

// Zero out the first row if needed
if (firstRow) {
    for (int j = 0; j < n; j++) {
```

```
                matrix[0][j] = 0;
            }
        }

        // Zero out the first column if needed
        if (firstCol) {
            for (int i = 0; i < m; i++) {
                matrix[i][0] = 0;
            }
        }
    }
};

// For testing
void printMatrix(const vector<vector<int>>& matrix) {
    for (const auto& row : matrix) {
        for (int val : row) {
            cout << val << " ";
        }
        cout << endl;
    }
    cout << endl;
}

int main() {
    Solution sol;
    vector<vector<int>> matrix = {
```

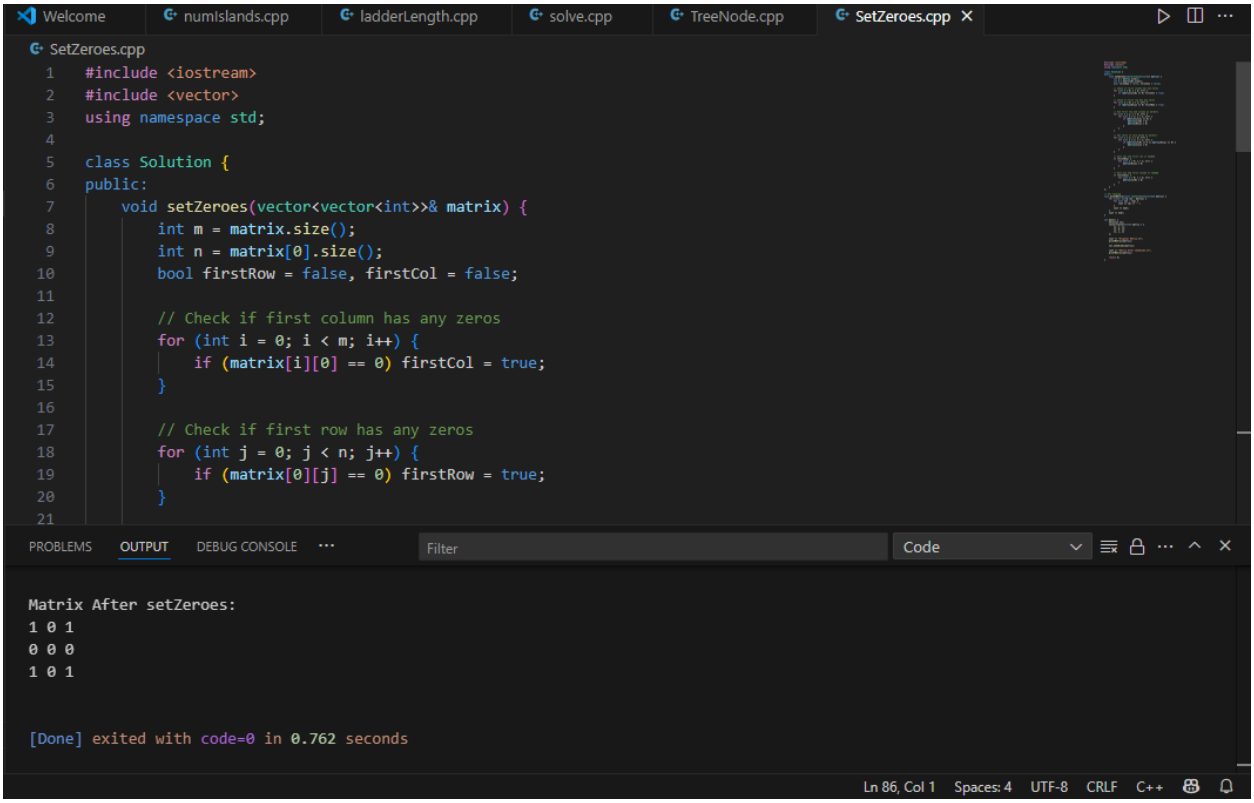
```
    {1, 1, 1},
    {1, 0, 1},
    {1, 1, 1}
};

cout << "Original Matrix:\n";
printMatrix(matrix);

sol.setZeroes(matrix);

cout << "Matrix After setZeroes:\n";
printMatrix(matrix);

return 0;
```



Longest Substring Without Repeating Characters:

Given a string s, find the length of the longest substring that does not contain any repeating characters.

Solution:

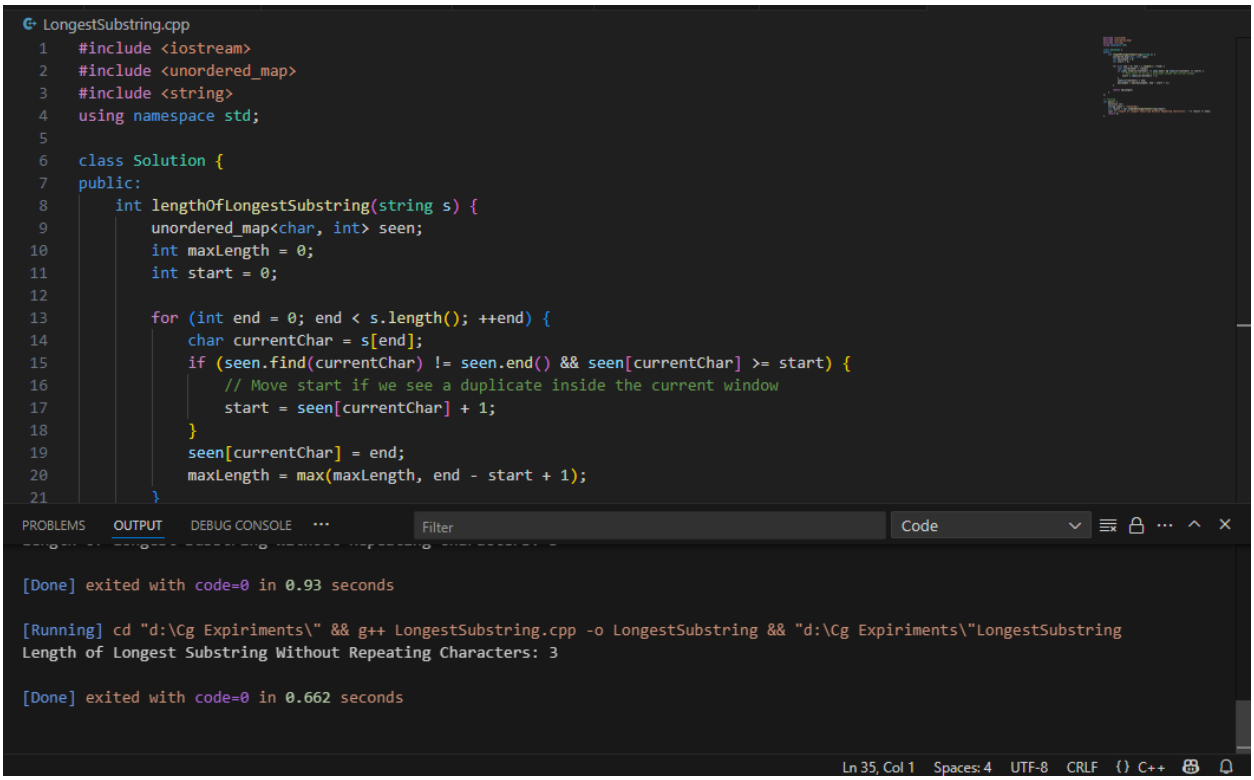
```
#include <iostream>
#include <unordered_map>
#include <string>
using namespace std;

class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        unordered_map<char, int> seen;
        int maxLength = 0;
        int start = 0;

        for (int end = 0; end < s.length(); ++end) {
            char currentChar = s[end];
            if (seen.find(currentChar) != seen.end() && seen[currentChar] >= start) {
                // Move start if we see a duplicate inside the current window
                start = seen[currentChar] + 1;
            }
            seen[currentChar] = end;
            maxLength = max(maxLength, end - start + 1);
        }
```

```
        return maxLength;
    }
};

// Testing
int main() {
    Solution sol;
    string input = "abcabcbb";
    int result = sol.lengthOfLongestSubstring(input);
    cout << "Length of Longest Substring Without Repeating Characters: " << result << endl;
    return 0;
}
```



Reverse Linked List II:

Given the head of a singly linked list and two integers left and right, reverse the nodes of the list from position left to right

Solution:

```
#include <iostream>
#include <vector>
using namespace std;

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

class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (!head || left == right) return head;

        ListNode* dummy = new ListNode(0);
        dummy->next = head;
        ListNode* prev = dummy;

        for (int i = 1; i < left; i++) {
            prev = prev->next;
        }
```

```
        ListNode* curr = prev->next;
        ListNode* next = nullptr;
        for (int i = 0; i < right - left; i++) {
            next = curr->next;
            curr->next = next->next;
            next->next = prev->next;
            prev->next = next;
        }

        return dummy->next;
    }
};

void printList(ListNode* head) {
    while (head) {
        cout << head->val;
        if (head->next) cout << " -> ";
        head = head->next;
    }
    cout << endl;
}

ListNode* createList(vector<int> vals) {
    if (vals.empty()) return nullptr;
    ListNode* head = new ListNode(vals[0]);
    ListNode* curr = head;
```

```
    for (int i = 1; i < vals.size(); i++) {
        curr->next = new ListNode(vals[i]);
        curr = curr->next;
    }
    return head;
}

int main() {
    Solution sol;
    vector<int> vals = {1, 2, 3, 4, 5};
    ListNode* head = createList(vals);
    cout << "Original List: ";
    printList(head);

    int left = 2, right = 4;
    ListNode* result = sol.reverseBetween(head, left, right);
    cout << "Modified List: ";
    printList(result);

    return 0;
}
```

ReverseBetween.cpp

```
57  int main() {
58      vector<int> vals = {1, 2, 3, 4, 5};
59      ListNode* head = createList(vals);
60      cout << "Original List: ";
61      printList(head);
62
63      int left = 2, right = 4;
64      ListNode* result = sol.reverseBetween(head, left, right);
65      cout << "Modified List: ";
66      printList(result);
67
68      return 0;
69  }
70
71
```

PROBLEMS OUTPUT DEBUG CONSOLE ... Filter Code

```
[Done] exited with code=1 in 0.281 seconds

[Running] cd "d:\Cg Expiriments\" && g++ ReverseBetween.cpp -o ReverseBetween && "d:\Cg Expiriments\"ReverseBetween
Original List: 1 -> 2 -> 3 -> 4 -> 5
Modified List: 1 -> 4 -> 3 -> 2 -> 5

[Done] exited with code=0 in 0.537 seconds
```