

Ap-Fast Learners Assignment

Name: Harmandeep Singh

Section:614-B

UID: 22BCS14975

1.Set Matrix Zeroes

```
class Solution {
public:
void setZeroes(vector<vector<int>>& matrix) {
int m = matrix.size(), n = matrix[0].size();
bool firstRowZero = false, firstColZero = false;
for (int i = 0; i < m; i++) {
if (matrix[i][0] == 0) {
firstColZero = true;
break;
}
}
for (int j = 0; j < n; j++) {
if (matrix[0][j] == 0) {
firstRowZero = true;
break;
}
}
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;
}
}
}
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;
}
}
}
if (firstColZero) {
for (int i = 0; i < m; i++) {
matrix[i][0] = 0;
}
}
if (firstRowZero) {
for (int j = 0; j < n; j++) {
matrix[0][j] = 0;
}
```

```

}
}
}
};

```

Accepted 202 / 202 testcases passed
 Harma... submitted at Apr 18, 2025 00:22

Runtime: 0 ms | Beats 100.00%
 Memory: 18.48 MB | Beats 95.63%

Code | C++

```

class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {

```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

matrix =
 [[1,1,1],[1,0,1],[1,1,1]]

Output

[[1,0,1],[0,0,0],[1,0,1]]

Expected

2.Longest Substring without Repeating Characters

```

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

        for (int end = 0; end < s.length(); end++) {
            char currentChar = s[end];
            if (lastSeen.find(currentChar) != lastSeen.end() && lastSeen[currentChar] >= start) {
                start = lastSeen[currentChar] + 1;
            }

            lastSeen[currentChar] = end;
            maxLength = max(maxLength, end - start + 1);
        }

        return maxLength;
    }
};

```

The screenshot displays the LeetCode submission interface for the 'Longest Substring Without Repeating Characters' problem. The left sidebar shows the problem description, a 'Solution' button, and performance metrics: 8 ms runtime (beats 61.49%) and 11.84 MB memory (beats 58.79%). A bar chart shows the distribution of runtimes. The main area contains a C++ code editor with the following solution:

```

1 class Solution {
2 public:
3     int lengthOfLongestSubstring(string s) {
4         unordered_map<char, int> lastSeen;
5         int maxLength = 0, start = 0;
6
7         for (int end = 0; end < s.length(); end++) {
8             char currentChar = s[end];
9             if (lastSeen.find(currentChar) != lastSeen.end() && lastSeen[currentChar] >= start) {
10                 start = lastSeen[currentChar] + 1;
11             }
12             lastSeen[currentChar] = end;
13             maxLength = max(maxLength, end - start + 1);
14         }
15         return maxLength;
16     }
17 }

```

Below the code editor, the 'Testcase' section shows 'Accepted' status with a runtime of 0 ms. The test case details are as follows:

Case	Input	Output	Expected
Case 1	s = "abcbcb"	3	3

3.Reverse Linked List

```

class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* current = head;

        while (current) {
            ListNode* nextNode = current->next;
            current->next = prev;
            prev = current;
            current = nextNode;
        }

        return prev;
    }

    ListNode* createLinkedList(const vector<int>& nums) {
        if (nums.empty()) return nullptr;

        ListNode* head = new ListNode(nums[0]);
        ListNode* current = head;
        for (size_t i = 1; i < nums.size(); i++) {
            current->next = new ListNode(nums[i]);
            current = current->next;
        }
        return head;
    }

    void printLinkedList(ListNode* head) {

```

```

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

```

The screenshot shows a C++ IDE with a submission interface on the left and a code editor on the right. The left sidebar indicates that the solution is 'Accepted' with 28/28 testcases passed. It shows a runtime of 0 ms and memory usage of 13.40 MB. The code editor contains the following C++ code:

```

12 public:
13     ListNode* reverseList(ListNode* head) {
14         ListNode* prev = nullptr;
15         ListNode* current = head;
16
17         while (current) {
18             ListNode* nextNode = current->next;
19             current->next = prev;
20             prev = current;
21             current = nextNode;
22         }
23
24         return prev;
25     }
26
27     ListNode* createLinkedList(const vector<int>& nums) {
28         if (nums.empty()) return nullptr;
29
30         ListNode* head = new ListNode(nums[0]);
31         ListNode* current = head;
32         for (size_t i = 1; i < nums.size(); i++) {
33             current->next = new ListNode(nums[i]);
34             current = current->next;
35         }
36         return head;
37     }

```

The bottom panel shows the test result as 'Accepted' with a runtime of 0 ms.

4. Linked List cycle

```

class Solution {
public:
bool hasCycle(ListNode *head) {
if (!head || !head->next) return false;

ListNode* slow = head;
ListNode* fast = head;

while (fast && fast->next) {
slow = slow->next;
fast = fast->next->next;

if (slow == fast) {
return true;
}
}
}
}

```

```

return false;
}

ListNode* createCycleList(const vector<int>& nums, int pos) {
if (nums.empty()) return nullptr;

ListNode* head = new ListNode(nums[0]);
ListNode* current = head;
ListNode* cycleNode = nullptr;

for (size_t i = 1; i < nums.size(); i++) {
current->next = new ListNode(nums[i]);
current = current->next;
if (static_cast<int>(i) == pos) {
cycleNode = current;
}
}

if (pos >= 0) {
current->next = cycleNode;
}

return head;
}
};

```

The screenshot shows a C++ code editor with a solution for reversing a linked list. The left panel displays the following statistics:

- Runtime:** 0 ms, Beats: 100.00%
- Memory:** 13.40 MB, Beats: 69.65%

The right panel shows the C++ code:

```

// Code
12 public:
13     ListNode* reverseList(ListNode* head) {
14         ListNode* prev = nullptr;
15         ListNode* current = head;
16
17         while (current) {
18             ListNode* nextNode = current->next;
19             current->next = prev;
20             prev = current;
21             current = nextNode;
22         }
23
24         return prev;
25     }
26
27     ListNode* createLinkedList(const vector<int>& nums) {
28         if (nums.empty()) return nullptr;
29
30         ListNode* head = new ListNode(nums[0]);
31         ListNode* current = head;
32         for (size_t i = 1; i < nums.size(); i++) {
33             current->next = new ListNode(nums[i]);
34             current = current->next;
35         }
36         return head;
37     }
38
39     ~ListNode() {
40         delete next;
41     }
42 };

```

The bottom panel shows the test result: **Accepted** Runtime: 0 ms.

5. Search a 2D Matrix II

```

class Solution {
public:
bool searchMatrix(vector<vector<int>>& matrix, int target) {
if (matrix.empty() || matrix[0].empty()) return false;

int m = matrix.size();

```

```

int n = matrix[0].size();

int row = 0, col = n - 1;

while (row < m && col >= 0) {
    if (matrix[row][col] == target)
        return true;
    else if (matrix[row][col] > target)
        col--;
    else
        row++;
}

return false;
}
};

```

Accepted 29 / 29 testcases passed
 Harma... submitted at Apr 18, 2025 00:34

Runtime
 12 ms | Beats 39.50%

Memory
 11.96 MB | Beats 23.14%

Code | C++

```

/**
 * Definition for singly-linked list.
 * struct ListNode {

```

```

class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next) return false;

        ListNode* slow = head;
        ListNode* fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast) {
                return true;
            }
        }

        return false;
    }

    ListNode* createCycleList(const vector<int>& nums, int pos) {
        if (nums.empty()) return nullptr;

        ListNode* head = new ListNode(nums[0]);
        ListNode* current = head;
        ListNode* cycleNode = nullptr;

```

Testcase | Test Result

Accepted Runtime: 0 ms