Ap-Fast Learners Assignment

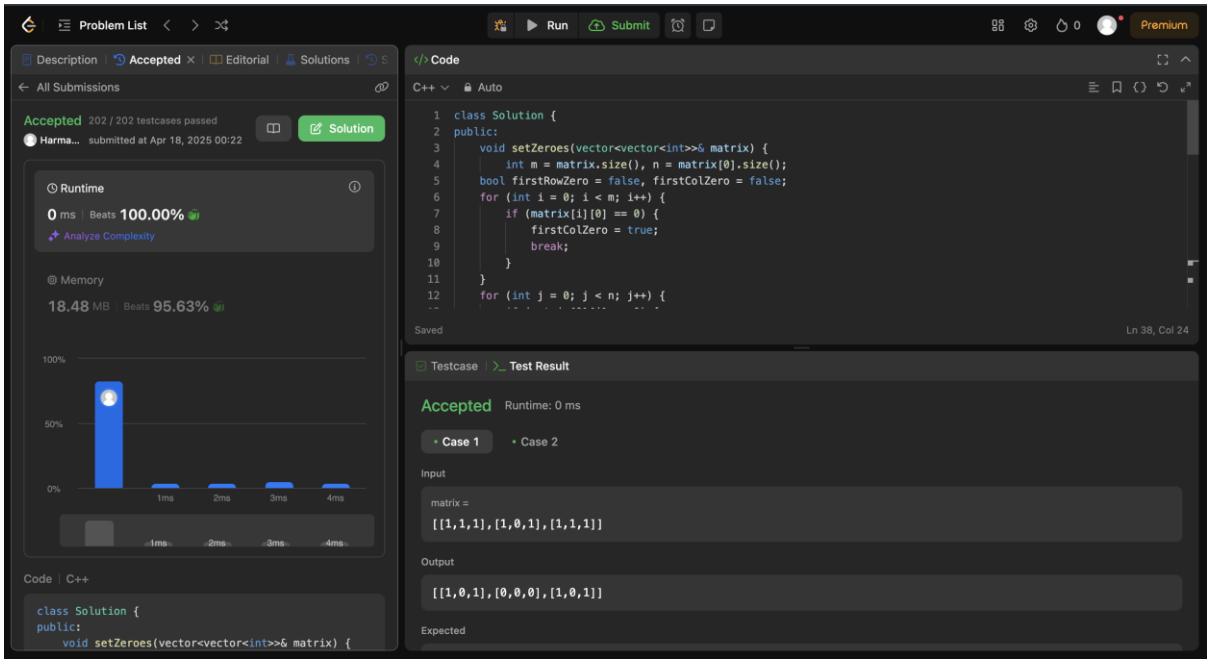
Name: Harmandeep Singh **Section:614-B**

UID: 22BCS14975

1.Set Matrix Zeroes

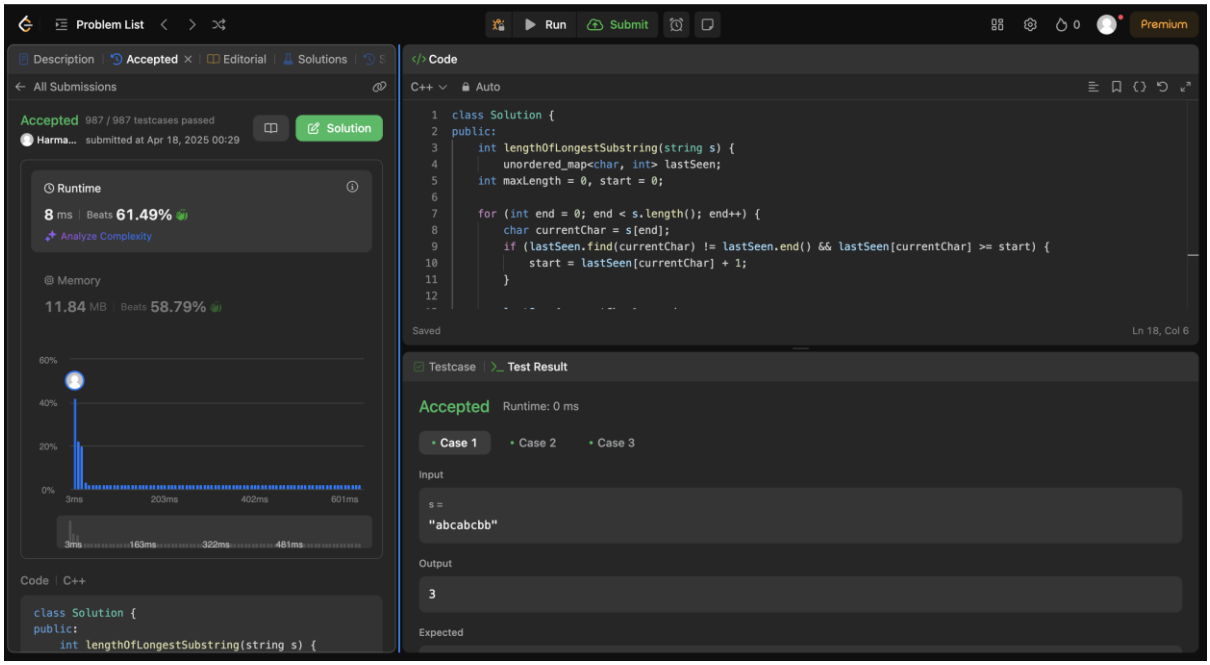
```
class Solution {
public:
void setZeroes(vector<vector<int>>& matrix) {
int m = matrix.size(), n = matrix[0].size();
bool firstRowZero = false, firstColZero = false;
for (int i = 0; i < m; i++) {
if (matrix[i][0] == 0) {
firstColZero = true;
break;
}
}
for (int j = 0; j < n; j++) {
if (matrix[0][j]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

```
}
}
}
};
```



2.Longest Substring without Repeating Characters

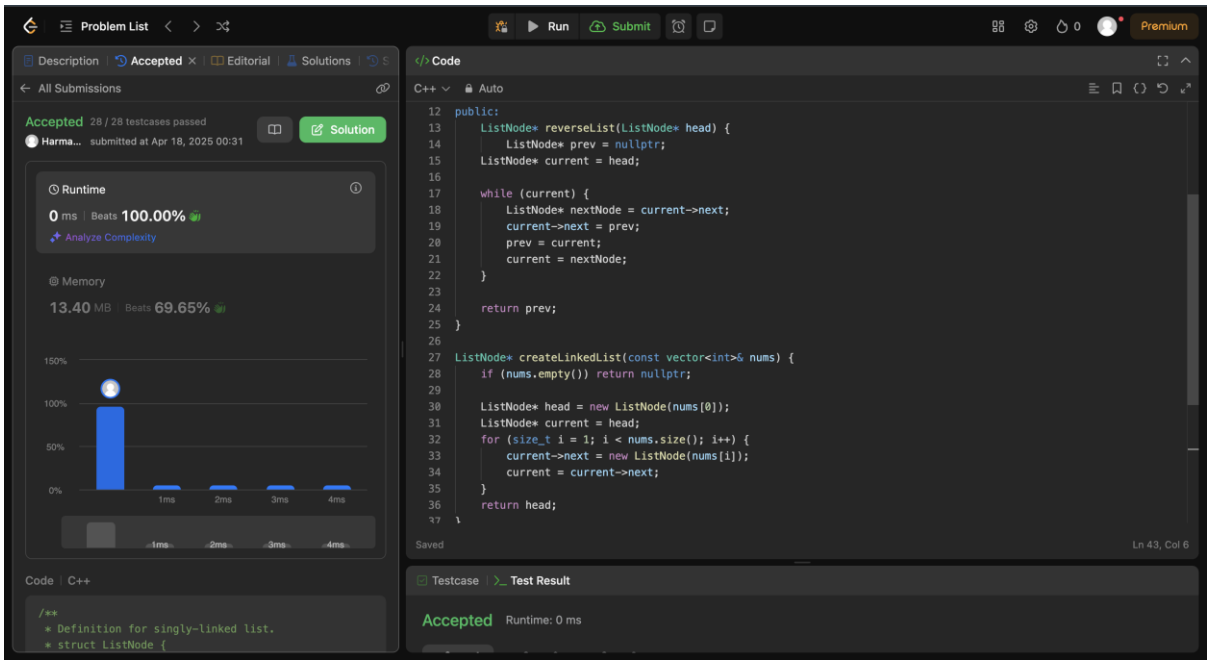
```
class Solution {
public:
int lengthOfLongestSubstring(string s) {
unordered_map<char, int> lastSeen;
int maxLength = 0, start = 0;

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;
}
};
```



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) {
```

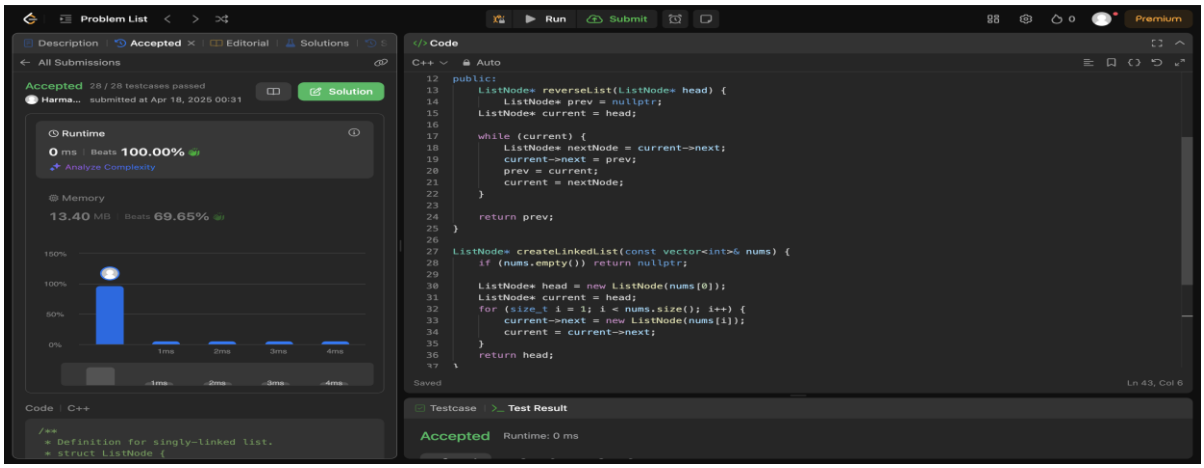
```
while (head) {
cout << head->val;
if (head->next) cout << " -> ";
head = head->next;
}
cout << endl;
}
};
```



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;
}
};
```



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;
}
};
```