Arrays: Leetcode

(Part 2)

Q1. Two Sum

```
vector<int> twoSum(vector<int>& nums, int target) {
    unordered_map<int, int> m;
    for (int i = 0; i < nums.size(); ++i) {
        if (m.count(target - nums[i]))
            return {m[target - nums[i]], i};
        m[nums[i]] = i;
    }
    return {};
}
```

Q2. Add two numbers

```
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode* dummy = new ListNode(0);
    ListNode* curr = dummy;
    int carry = 0;
    while (l1 || l2 || carry) {
        int sum = (l1 ? l1->val : 0) + (l2 ? l2->val : 0) +
carry;
        carry = sum / 10;
        curr->next = new ListNode(sum % 10);
        curr = curr->next;
        if (l1) l1 = l1->next;
        if (l2) l2 = l2->next;
    }
    return dummy->next;
}
```

Q5. Longest Palindromic Substring

```
string longestPalindrome(string s) {
    int start = 0, maxLen = 1, n = s.size();
    vector<vector<bool>> dp(n, vector<bool>(n));
    for (int i = 0; i < n; ++i) dp[i][i] = true;
    for (int len = 2; len <= n; ++len)
        for (int i = 0; i <= n - len; ++i) {
            int j = i + len - 1;
            if (s[i] == s[j] && (len == 2 || dp[i+1][j-1]))
{
                dp[i][j] = true;
                if (len > maxLen) start = i, maxLen = len;
            }
        }
    return s.substr(start, maxLen);
}
```

Q14. Longest Common Prefix

```
string longestCommonPrefix(vector<string>& strs) {
    if (strs.empty()) return "";
    string prefix = strs[0];
    for (int i = 1; i < strs.size(); ++i)
        while (strs[i].find(prefix) != 0)
            prefix = prefix.substr(0, prefix.length() - 1);
    return prefix;
}
```

Q20. Valid Parentheses

```
bool isValid(string s) {
    stack<char> stk;
    unordered_map<char, char> m = {{')','('}, {']','['},
{'}','{'}};
    for (char c : s) {
        if (m.count(c)) {
            if (stk.empty() || stk.top() != m[c]) return
false;
            stk.pop();
        } else stk.push(c);
    }
    return stk.empty();
}
```

Q36. Valid Sudoku

```
bool isValidSudoku(vector<vector<char>>& board) {
    unordered_set<string> seen;
    for (int i = 0; i < 9; ++i)
        for (int j = 0; j < 9; ++j) {
            char num = board[i][j];
            if (num == '.') continue;
            string row = to_string(num) + " in row " +
to_string(i);
            string col = to_string(num) + " in col " +
to_string(j);
            string box = to_string(num) + " in box " +
to_string(i/3) + "-" + to_string(j/3);
            if (!seen.insert(row).second ||
!seen.insert(col).second || !seen.insert(box).second)
                return false;
        }
    return true;
}
```

Q48. Rotate Image

```
void rotate(vector<vector<int>>& matrix) {
    int n = matrix.size();
    for (int i = 0; i < n; ++i)
        for (int j = i; j < n; ++j)
            swap(matrix[i][j], matrix[j][i]);
    for (int i = 0; i < n; ++i)
        reverse(matrix[i].begin(), matrix[i].end());
}
```

Q54. Spiral Matrix

```
vector<int> spiralOrder(vector<vector<int>>& matrix) {
    vector<int> res;
    if (matrix.empty()) return res;
    int top = 0, bottom = matrix.size() - 1;
    int left = 0, right = matrix[0].size() - 1;
    while (top <= bottom && left <= right) {
        for (int i = left; i <= right; ++i)
res.push_back(matrix[top][i]);
        ++top;
        for (int i = top; i <= bottom; ++i)
res.push_back(matrix[i][right]);
        --right;
        if (top <= bottom)
            for (int i = right; i >= left; --i)
res.push_back(matrix[bottom][i]);
        --bottom;
        if (left <= right)
            for (int i = bottom; i >= top; --i)
res.push_back(matrix[i][left]);
        ++left;
    }
    return res;
}
```

Q59. Spiral Matrix II

```
vector<vector<int>> generateMatrix(int n) {
    vector<vector<int>> res(n, vector<int>(n));
    int top = 0, bottom = n-1, left = 0, right = n-1, val =
1;
    while (top <= bottom && left <= right) {
        for (int i = left; i <= right; ++i) res[top][i] =
val++;
        ++top;
        for (int i = top; i <= bottom; ++i) res[i][right] =
val++;
        --right;
        for (int i = right; i >= left; --i) res[bottom][i] =
val++;
        --bottom;
        for (int i = bottom; i >= top; --i) res[i][left] =
val++;
        ++left;
    }
    return res;
}
```

Q73. Set Matrix Zeroes

```
void setZeroes(vector<vector<int>>& matrix) {
    int m = matrix.size(), n = matrix[0].size();
    bool firstRow = false, firstCol = false;
    for (int i = 0; i < m; ++i) if (matrix[i][0] == 0)
firstCol = true;
    for (int j = 0; j < n; ++j) if (matrix[0][j] == 0)
firstRow = true;
    for (int i = 1; i < m; ++i)
        for (int j = 1; j < n; ++j)
            if (matrix[i][j] == 0)
                matrix[i][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 (firstRow) fill(matrix[0].begin(), matrix[0].end(),
0);
    if (firstCol) for (int i = 0; i < m; ++i) matrix[i][0] =
0;
}
```