WORKSHEET 9

Student Name: Bhupesh Kumar **UID:22BCS11723**

Branch: CSE **Section/Group: NTPP 603/B**

Semester: 06 **Date of Performance: 03/04/2025**

Subject Name: AP Lab II **Subject Code: 22CSP-351**

1. Aim:

a. Number of Islands.

b. Word Ladder

c. Surrounded Regions

2. Source Code:

a.

```
  class Solution {
public:
 int numIslands(vector<vector<char>>& grid) {
   constexpr int kDirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
   const int m = grid.size();
   const int n = grid[0].size();
   int ans = 0;

   auto bfs = [&](int r, int c) {
     queue<pair<int, int>> q{{{r, c}}};
     grid[r][c] = '2';  // Mark '2' as visited.
     while (!q.empty()) {
       const auto [i, j] = q.front();
       q.pop();
       for (const auto& [dx, dy] : kDirs) {
         const int x = i + dx;
         const int y = j + dy;
         if (x < 0 || x == m || y < 0 || y == n)
           continue;
```

```
          if (grid[x][y] != '1')
            continue;
          q.emplace(x, y);
          grid[x][y] = '2';  // Mark '2' as visited.
        }
      }
    };

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        if (grid[i][j] == '1') {
          bfs(i, j);
          ++ans;
        }

    return ans;
  }
};
```

b.

```
class Solution {
 public:
  void solve(vector<vector<char>>& board) {
    if (board.empty())
      return;

    constexpr int kDirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
    const int m = board.size();
    const int n = board[0].size();

    queue<pair<int, int>> q;

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        if (i * j == 0 || i == m - 1 || j == n - 1)
          if (board[i][j] == 'O') {
            q.emplace(i, j);
            board[i][j] = '*';
          }

    // Mark the grids that stretch from the four sides with '*'.
    while (!q.empty()) {
```

```
        const auto [i, j] = q.front();
        q.pop();
        for (const auto& [dx, dy] : kDirs) {
          const int x = i + dx;
          const int y = j + dy;
          if (x < 0 || x == m || y < 0 || y == n)
            continue;
          if (board[x][y] != 'O')
            continue;
          q.emplace(x, y);
          board[x][y] = '*';
        }
      }

      for (vector<char>& row : board)
        for (char& c : row)
          if (c == '*')
            c = 'O';
          else if (c == 'O')
            c = 'X';
  }
};
```

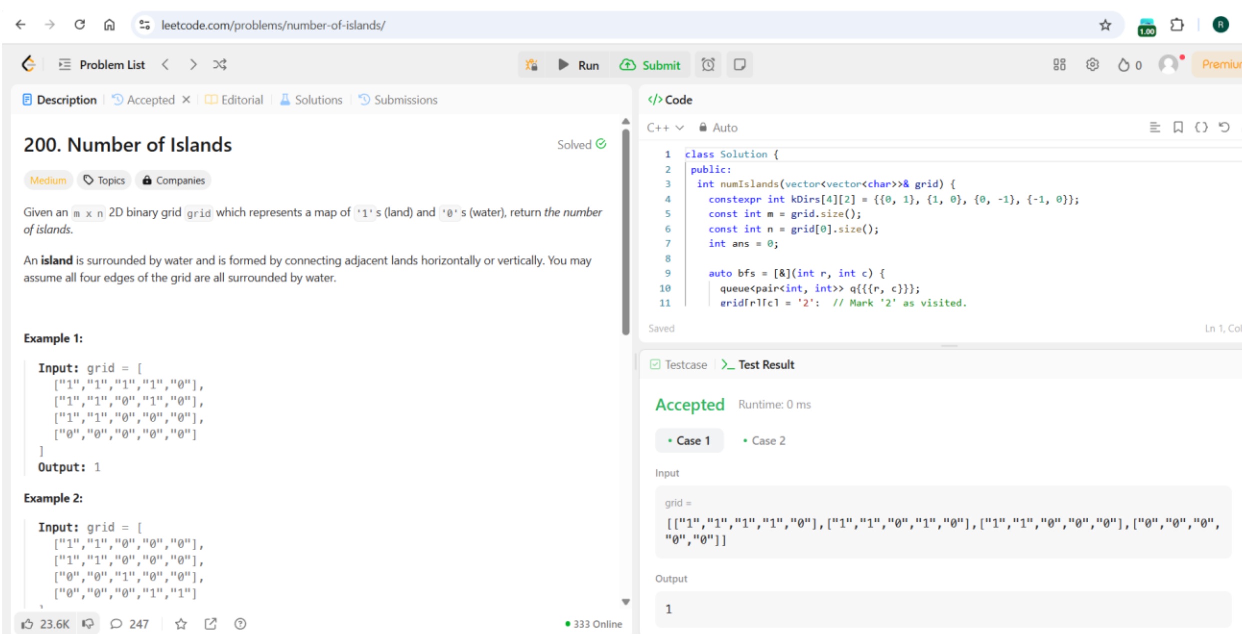
C.

```
class Solution {
 public:
  int ladderLength(string beginWord, string endWord, vector<string>& wordList) {
    unordered_set<string> wordSet(wordList.begin(), wordList.end());
    if (!wordSet.contains(endWord))
      return 0;

    queue<string> q{{beginWord}};

    for (int step = 1; !q.empty(); ++step)
      for (int sz = q.size(); sz > 0; --sz) {
        string word = q.front();
        q.pop();
        for (int i = 0; i < word.length(); ++i) {
          const char cache = word[i];
          for (char c = 'a'; c <= 'z'; ++c) {
            word[i] = c;
            if (word == endWord)
              return step + 1;
            if (wordSet.contains(word)) {
```

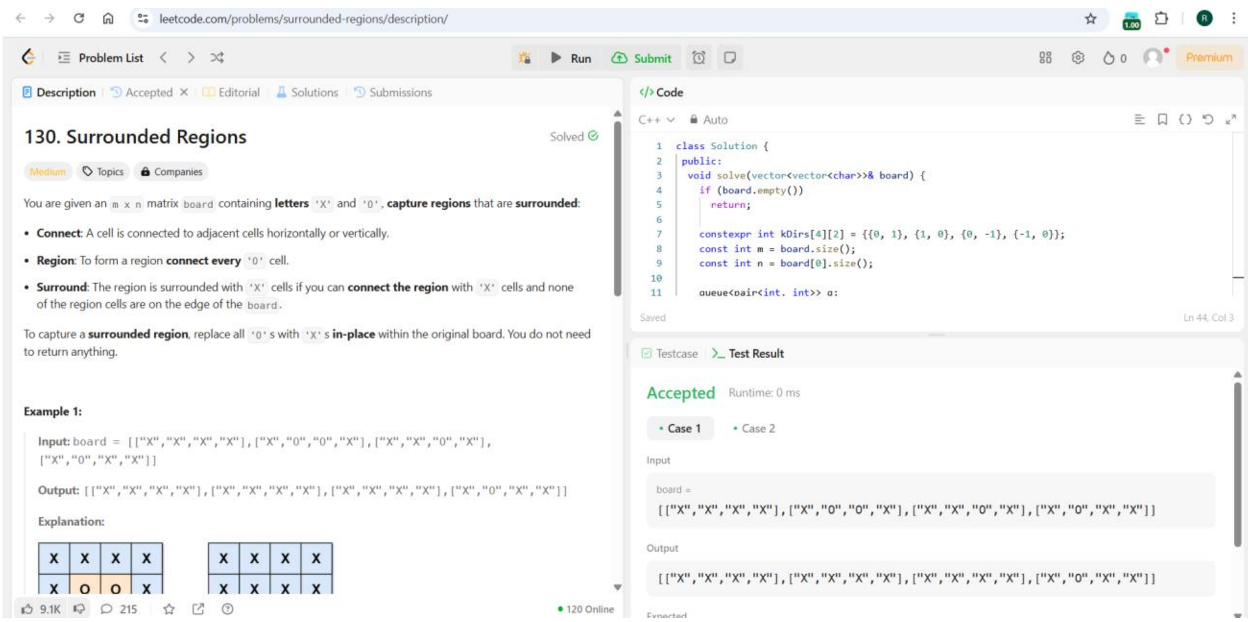
```
                q.push(word);
                wordSet.erase(word);
              }
            }
            word[i] = cache;
          }
        }

      return 0;
    }
};
```

3. Screenshot of Outputs:

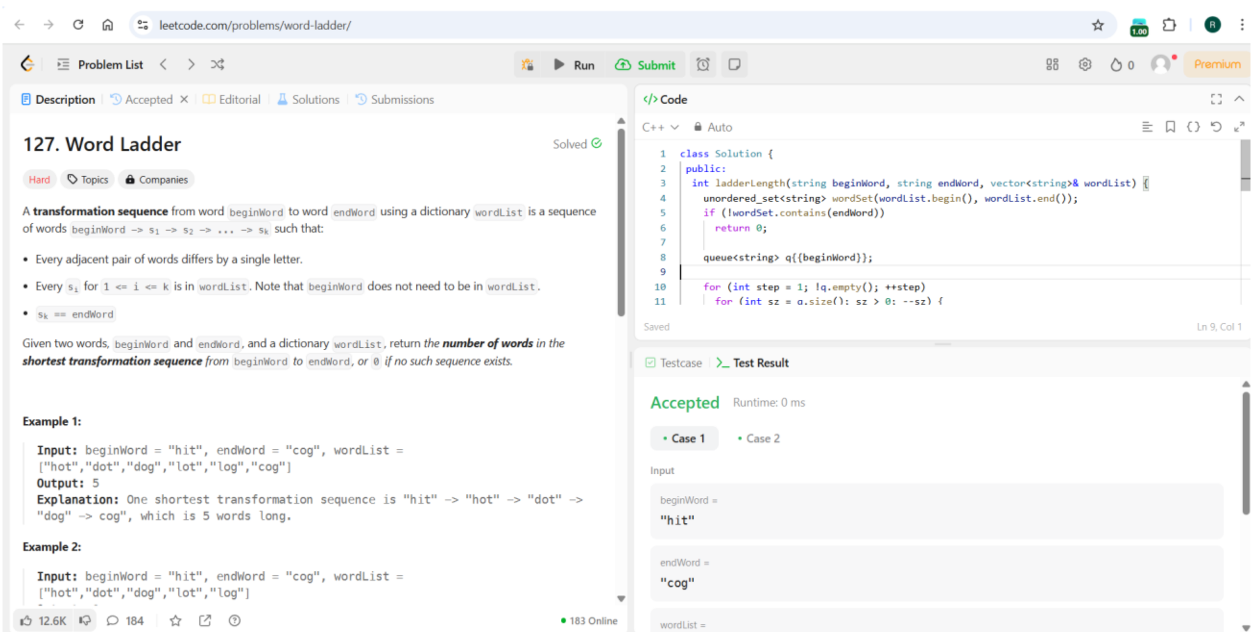
a.



b.



c.



4. Learning Outcomes

(i) Learned about graph data structures.