CS106B Midterm, Fall 2025

CS106B Midterm, Fall 2025 Answer Key

Problem 1, Basic C++ and Containers (5 parts): 22pts

Part A:

```
i= 0, j= 0.
i= 1, j= 1. i= 1, j= 0.
i= 2, j= 2. i= 2, j= 1. i= 2, j= 0.
```

Part B:

```
3
13
```

Part C:

```
Stanford
University
Junior
Leland
```

Part D:

```
m, p
a, r
r, o
g, g
o, r
r, a
p, m
```

Part E:

```
7
8
9
10
```

Problem 2, Maximum cross-sum with Vectors and Grids: 12pts

```
int maxCrossSum(const Grid<int>& g) {
    int rows = g.numRows();
    int cols = g.numCols();

    Vector<int> rowCounts(rows);
    Vector<int> colCounts(cols);

    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            rowCounts[r] += g[r][c];
            colCounts[c] += g[r][c];
        }
    }

    // find the min
    int min = INT_MIN;

    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            int crossSum = rowCounts[r] + colCounts[c] - g[r][c];
            if (crossSum > min) {
                min = crossSum;
            }
        }
    }
    return min;
}
```

Problem 3, Printing First Non-Repeating Characters, 12 pts

```
string firstNonRepeatingStream(const string s) {
    Map<char,int> freq;
    Queue<char> q;
    string result;

    for (char c : s) {
        freq[c]++;    // update count
        q.enqueue(c); // keep arrival order

        // Discard any queue front that is no longer unique
        while (!q.isEmpty() && freq[q.peek()] > 1) {
            q.dequeue();
        }

        if (q.isEmpty()) {
            result += '_';
        } else {
            result += q.peek();
        }
    }
    return result;
}
```

Problem 4, Big O: 12pts

A. $O(N^2)$

B. $O(NlogN)$

C. $O(a^N)$

D. $O(logN)$

E. $O(N)$

F. $O(N^2)$

Problem 5, Recursion, Collapsing Runs: 6pts

```
string collapseRuns(string s) {
    if (s.length() <= 1) {
        return s;
    }
    if (s[0] == s[1]) {
        return collapseRuns(s.substr(1));
    } else {
        return s[0] + collapseRuns(s.substr(1));
    }
}
```

Problem 6, Backtracking, Recursive Maze Solver: 12pts

```
void solveMazeDFSRecursiveHelper(Grid<bool>& maze,
            Vector<GridLocation> &path,
            Set<GridLocation> &visited) {
    GridLocation pathEnd = path[path.size() - 1];
    GridLocation mazeExit = {maze.numRows() - 1, maze.numCols() - 1};

    // base case
    if (pathEnd == mazeExit) {
        return;
    }

    for (GridLocation possibleNext : generateValidMoves(maze, pathEnd)) {
        // chose the next path
        if (!visited.contains(possibleNext)) {
            path.add(possibleNext);
            visited.add(possibleNext);
            solveMazeDFSRecursiveHelper(maze, path, visited);
            if (path[path.size() - 1] == mazeExit) {
                break;
            }
            // undo
            path.remove(path.size() - 1);
        }
    }
}
```