Week 2 Section Solutions

1. Collection Mystery (CollectionMystery6 on CodeStepByStep)

Stacks:	Output:
`{1, 2, 3, 4, 5, 6}`	{6, 4, 2, 1, 3, 5}
`{42, 3, 12, 15, 9, 71, 88}`	{88, 12, 42, 3, 15, 9, 71}
`{65, 30, 10, 20, 45, 55, 6, 1}`	{6, 20, 10, 30, 65, 45, 55, 1}

2. Mirror

```
void mirror(Grid<int>& grid) {
    for (int r = 0; r < grid.numRows(); r++) {
        // start at r+1 rather than 0 to avoid double-swapping
        for (int c = r + 1; c < grid.numCols(); c++) {
            int temp = grid[r][c];
            grid[r][c] = grid[c][r];
            grid[c][r] = temp;
        }
    }
}
```

Bonus:

```
void mirror(Grid<int>& grid) {
    Grid<int> result(grid.numCols(), grid.numRows());
    for (int r = 0; r < grid.numRows(); r++) {
        for (int c = 0; c < grid.numCols(); c++) {
            result[r][c] = grid[c][r];
        }
    }

    grid = result;
}
```

Thanks to Marty Stepp and other CS106B and X instructors and TAs for contributing problems on this handout.

3. CrossSum

```
int crossSum(Grid<int>& grid, int row, int col) {
    int sum = 0;
    for (int c = 0; c < grid.numCols(); c++) {
        sum += grid[row][c];
    }
    for (int r = 0; r < grid.numRows(); r++) {
        sum += grid[r][col];
    }
    sum -= grid[row][col]; // subtract center because it was added twice
    return sum;
}
```

4. RemoveConsecutiveDuplicates

```
void removeConsecutiveDuplicates(Vector<int>& v) {
    for (int i = 0; i < v.size() - 1; i++) {
        if (v[i] == v[i + 1]) {
            v.remove(i + 1);
            i--;
        }
    }
}
```

5. Stutter

```
void stutter(Queue<int>& q) {
    int size = q.size();
    for (int i = 0; i < size; i++) {
        int n = q.dequeue();
        q.enqueue(n);
        q.enqueue(n);
    }
}
```

6. CheckBalance

```
int checkBalance(string code) {
    Stack<char> parens;
    for (int i = 0; i < (int) code.length(); i++) {
        char c = code[i];
        if (c == '(' || c == '{') {
            parens.push(c);
        } else if (c == ')' || c == '}') {
            if (parens.isEmpty()) {
                return i;
            }
            char top = parens.pop();
            if ((top == '(' && c != ')') || (top == '{' && c != '}')) {
                return i;
            }
        }
    }
    if (parens.isEmpty()) {
        return -1;              // balanced
    } else {
        return code.length();
    }
}
```