CS106B
Spring 2016 Cynthia Lee

Section 1 (Week 2) - SOLUTION

1. Parameter Mystery #1.

```
Output:          If changed to cs(b, a);       Changed to a = cs(b, a);
XehranM          XehranM                       Cynthia
Cynthia          Marty                         Marty
```

2. Even Average.

```
// Bonus code in comments (numbers must be saved into a Vector)
int main() {
    int num = getInteger("Integer?");
    int sum = 0;
    int count = 0;
    // Vector<int> numbers;
    while (num != -1) {
        if (num % 2 == 0) {
            sum += num;
            count++;
        }
        // numbers += num;
        num = getInteger("Integer?");
    }
    double average = (double) sum / count;
    cout << "Average: " << average << endl;

    // cout << "Numbers greater than average:" << endl;
    // for (int i = 0; i < numbers.size(); i++) {
    //     if (numbers[i] > average) {
    //         cout << numbers[i] << " ";
    //     }
    //     cout << endl;
    // }

    return 0;
}
```

3. cumulative.

```
// Modifies each element of v at index i to store the sum of elements 0..i.
void cumulative(Vector<int>& v) {
    for (int i = 1; i < vec.size(); i++) {
        vec[i] += vec[i - 1];
    }
}
```

4. crossSum.

```
// Returns the sum of all elements in the given row and/or column of the
grid.
int crossSum(Grid<int>& grid, int row, int col) {
    if (!grid.inBounds(row, col)) return 0; // fixed April 6 2016
    int sum = 0;
    for (int i = 0; i < grid.numCols(); i++) {
        sum += grid[row][i];
    }
    for (int i = 0; i < grid.numRows(); i++) {
        sum += grid[i][col];
    }
    return sum – grid[row][col];
}
```

5. splitStack.

```
// Reorders the stack so negatives are on bottom and non-negatives are on
top.
void splitStack(Stack<int>& stack) {
    Queue<int> queue;             // Move everything into a queue
    while (!stack.isEmpty()) {
        queue.enqueue(stack.pop());
    }

    int size = queue.size();      // Put negative numbers back in stack;
    for (int i = 0; i < size; i++) {  // re-insert non-negative into queue
        int num = queue.dequeue();
        if (num >= 0) {
            queue.enqueue(num);
        } else {
            stack.push(num);
        }
    }

    while (!queue.isEmpty()) {    // Put remaining non-negs back in stack
        stack.push(queue.dequeue());
    }
}
```

6. Debugging removeDuplicates.

```
The for loop iterates once too far, and we need to decrement the loop index
whenever we remove an element.

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