CS 106B Section 1 (Week 2)

Welcome to CS 106B section! Work with your classmates and section leader on these problems. Feel free to jump around to different topics based on what you want to discuss. There is no requirement on how many you need to finish. Additionally, some problems have a bonus on them if you want something extra to think about.

Recommended problems: #2, #4
Extra practice problems: 4.6, 4.8, 5.6, 5.9 (from Textbook)

---

1. Parameter Mystery #1. What is the console output of the following program?

```
1   string mystery(string a, string& b) {
2       b += a[0];
3       b[0] = 'X';
4       a = "Chris";
5       return a;
6   }
7
8   int main() {
9       string a = "Mehran";
10      string b = "Megan";
11      b = mystery(b, a);  // What if this line were:  mystery(b, a);      ?
12                          // Bonus: what if it were:  a = mystery(b, a);  ?
13      cout << a << endl;
14      cout << b << endl;
15      return 0;
16  }
```

---

2. Even Average. *(console input/output)*
Write a complete program that prompts the user for a sequence of non-zero integers, and then prints the average of all <u>even</u> numbers typed (ignoring odds). Stop prompting when the user types -1. Sample output:

```
Integer? 1
Integer? 3
Integer? 2
Integer? 6
Integer? 4
Integer? 10
Integer? 9
Integer? -1
Average: 5.5
```

Bonus: Also print the entered integers (odds and evens) that were greater than the average. Sample output:

```
Numbers greater than average: 6 10 9
```

CS 106B Section 1 (Week 2)

3. cumulative. *(Vector)*

Write a function named **cumulative** that accepts a reference to a Vector of integers as a parameter and modifies it so each element contains the sum of the original vector up through that index. For example, if a Vector variable v stores {1, 1, 2, 3, 5}, the call of cumulative(v); should modify it to store {1, 2, 4, 7, 12}. (There are many ways to do this problem. Try to do it in one pass through the original vector.)

```
void cumulative(Vector<int>& vec) { ...
```

---

4. crossSum. *(Grid)*

Write a function named crossSum that accepts three parameters: a reference to a Grid of integers, an integer row, and an integer column, and prints the sum of all numbers that appear in the row and/or column provided. For example, if a Grid variable named grid stores the following integers, the call of crossSum(grid, 1, 1) should return 2+5+8+4+6 or 25. If the row and/or column passed is out of bounds, return 0.

```
{{1, 2, 3},
{4, 5, 6},
{7, 8, 9}}

int crossSum(Grid<int>& grid, int row, int col) { ...
```

---

5. splitStack. *(Stack/Queue)*

Write a function named splitStack that accepts a reference to a Stack of integers as its parameter, and re-orders the stack so that all the non-negative numbers are at the top and all the negative numbers are at the bottom. For example, if a Stack variable named s stores {4, 0, -1, 5, -6, -3, 2, 7} from bottom to top, the call of splitStack(s); should change s to store an ordering such as {-3, -6, -1, 7, 2, 5, 0, 4}. You may declare only a single Queue as auxiliary storage, but do not declare any other data structures (e.g. arrays, grids, vectors, stacks).

```
void splitStack(Stack<int>& stack) { ...
```

---

6. Debugging removeDuplicates. *(Vector debugging)*

The following is a function that is supposed to get rid of any consecutive duplicates from a Vector, so for example, {1, 2, 2, 3, 4, 4, 4, 7} should become {1, 2, 3, 4, 7}. However, there are two bugs. What are they, and how can we fix the code?

```
1  void removeDuplicates(Vector<int>& vec) {
2      for (int i = 0; i < vec.size(); i++) {
3          if (vec[i] == vec[i + 1]) {
4              vec.remove(i + 1);
5          }
6      }
7  }
```