Section #1

Tyler Conklin and Kate Rydberg (Based on handouts by previous CS106B instructors and TAs, especially those of Ashley Taylor and Shreya Shankar.)

Strings, Grids, Vectors

Note: Section handouts always contain more material than can be done in section. We do this so that you have extra practice problems. These extra problems are great for studying for exams, or as "warm-up" problems for the assignment that uses the corresponding knowledge.

1. Mirror (Grid)

Write a function `mirror` that accepts a reference to a Grid of integers as a parameter and flips the grid along its diagonal. You may assume the grid is square; in other words, that it has the same number of rows as columns. For example, the grid below at left would be altered to give it the new grid state at right:

$$\begin{bmatrix} 6 & 1 & 9 & 4 \\ -2 & 5 & 8 & 12 \\ 14 & 39 & -6 & 18 \\ 21 & 55 & 73 & -3 \end{bmatrix} \rightarrow \begin{bmatrix} 6 & -2 & 14 & 21 \\ 1 & 5 & 39 & 55 \\ 9 & 8 & -6 & 73 \\ 4 & 12 & 18 & -3 \end{bmatrix}$$

Bonus: How would you solve this problem if the grid were not square?

2. Grid Mystery! (Grid)

What is the grid state after the following code?

```
Grid<int> g(4, 3);
for (int r = 0; r < g.numRows(); r++) {     // {{1, 2, 3},
    for (int c = 0; c < g.numCols(); c++) { //  {1, 2, 3},
        g[r][c] = c + 1;                    //  {1, 2, 3},
    }                                       //  {1, 2, 3}}
}

for (int c = 0; c < g.numCols(); c++) {
    for (int r = 1; r < g.numRows(); r++) {
        g[r][c] += g[r - 1][c];
    }
}
```

3. CrossSum (Grid)

Write a function named `crossSum` that accepts three parameters - a reference to a Grid of integers, and two integers for a row and column - and returns the sum of all numbers in the row/column cross provided. For example, if a grid named g stores the following integers:

	0	1	2
0	{{1,	2,	3}
1	{4,	5,	6}
2	{7,	8,	9}}

Then the call of `crossSum(g, 1, 1)` should return $(4+5+6+2+8)$ or 25. You may assume that the row and column passed are within the bounds of the grid. Do not modify the grid that is passed in.

4. Remove Consecutive Duplicates (Vector)

Write a function named `removeConsecutiveDuplicates` that accepts as a parameter a reference to a Vector of integers, and modifies it by removing any consecutive duplicates. For example, if a vector named v stores {`1, 2, 2, 3, 2, 2, 3`}, the call of `removeConsecutiveDuplicates(v);` should modify it to store {`1, 2, 3, 2, 3`}.

5. Collection Mystery! (Vector)

Write the output produced by the following function when passed each of the following vectors:

```
void collectionMystery4(Vector<int>& v) {
   for (int i = 1; i < v.size(); i += 2) {
       if (v[i - 1] >= v[i]) {
           v.remove(i);
           v.insert(0, 0);
       }
   }
   cout << v << endl;
}
```

{**10, 20, 10, 5**}

{**8, 2, 9, 7, -1, 55**}

{**0, 16, 9, 1, 64, 25, 25, 14, 0**}

6. Switch Pairs (Vector)

Write a function named `switchPairs` that switches the order of values in a Vector of integers in a pairwise fashion. Your function should accept as a parameter a reference to a Vector of integers and

should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. For example, if the vector initially stores $\{1, 4, 8, -3, 2, 7\}$, then your function should switch the first pair, (1 and 4), the second pair (8 and -3) and the third pair (2 and 7) to yield $\{4, 1, -3, 8, 7, 2\}$.

If there are an odd number of values in the list, the final element is not moved. For example, if the original list had been $\{1, 2, 3, 4, 5\}$, then the result would be $\{2, 1, 4, 3, 5\}$.