CS 106B Section Handout 2 (Week 3)

1. **Elements in Common:** Write a function `numInCommon` that takes two `Vector<int>` parameters by reference and returns the number of unique integers that occur in both lists. Use one or more `Sets` as storage to help you solve this problem.

   For example, if one list contains the values:

   ```
   [3, 7, 3, -1, 2, 3, 7, 2, 15, 15]
   ```

   and the other list contains the values:

   ```
   [-5, 15, 2, -1, 7, 15, 36]
   ```

   Your method should return 4 (because the elements -1, 2, 7, and 15 occur in both lists).

2. **Reverse:** Write a method reverse that accepts a Map from integers to strings as a parameter and returns a new Map of strings to integers that is the original's "reverse". The reverse of a map is defined here to be a new map that uses the values from the original as its keys and the keys from the original as its values. Since a map's values need not be unique but its keys must be, it is acceptable to have any of the original keys as the value in the result. In other words, if the original map has pairs `(k1, v)` and `(k2, v)`, the new map must contain either the pair `(v, k1)` or `(v, k2)`.

   For example, the following map:

   ```
   {42=Marty, 81=Reid, 17=Marty, 31=Sarah, 56=Aaron, 3=Amy}
   ```

   Your method could return the following new map (the order of the key/value pairs does not matter):

   ```
   {Amy=3, Marty=17, Reid=81, Aaron=29, Sarah=31}
   ```

3. **Reorder:** Write a method that takes a queue of integers as a parameter and that puts the integers into sorted (nondecreasing) order assuming that the queue is already stored by absolute value. For example, suppose that a variable called `q` stores the following sequence of values:

   ```
   front [1, 2, -2, 4, -5, 8, -8, 12, -15, 23] back
   ```

   Notice that the values appear in sorted order if you ignore the sign of the numbers. The call of `reorder(q)` should reorder the values so that the queue stores this sequence of values:

   ```
   front [-15, -8, -5, -2, 1, 2, 4, 8, 12, 23] back
   ```

   Notice that the values now appear in sorted order taking into account the sign of the numbers.

4. **numUnique:** Write a method `numUnique` that takes a Vector of `ints` as a parameter by reference and returns the number of unique integer values in the list. Use a Set as auxiliary storage to help you solve this problem.

   For example, if a list contains the values:

   ```
   [3, 7, 3, -1, 2, 3, 7, 2, 15, 15]
   ```

   your method should return 5. The empty list contains 0 unique values.

CS 106B Section Handout 2 (Week 3)

5. **Recursive Mystery 1**: Determine the result of calling this function with each set of parameters:

```
void mystery1(int x, int y) {
    if (y == 1) {
        cout << x;
    } else {
        cout << (x * y) << ", ";
        mystery1(x, y-1);
        cout << ", " << (x * y);
    }
}
```

Call	**Output**
mystery1(4,1)	
mystery1(4,2)	
mystery1(8,2)	
mystery1(4,3)	
mystery1(3,4)	

6. **Recursive Mystery 2**: Determine the result of calling this function with each of the following parameters:

```
int mystery2(int n) {
    if (n < 0) {
        return mystery2(-n);
    } else if(n < 10) {
        return n;
    } else {
        return n % 10 + mystery2(n/10);
    }
}
```

Call	**Return Value**
mystery2(8)	
mystery2(74)	
mystery2(-52)	
mystery2(3052)	
mystery2(82534)	

CS 106B Section Handout 2 (Week 3)

7. starString: Write a method `starString` that accepts an integer parameter `n` and returns a string of stars (asterisks) `2n` long. Throw an error if `n` is less than zero. For example:

Call	**Return Value**	**Reason**
`starString(0)`	`“*”`	`Because 2^0 = 1`
`starString(1)`	`“**”`	`Because 2^1 = 2`
`starString(2)`	`“****”`	`Because 2^2 = 4`
`starString(3)`	`“********”`	`Because 2^3 = 8`
`starString(4)`	`“****************”`	`Because 2^4 = 16`

8. stutter: Write a function stutter that accepts a Stack of integers as a parameter and replaces every value in the stack with two occurrences of that value. For example, suppose a stack has the following values:

```
bottom [3, 7, 1, 14, 9] top
```

This stack should be transformed by `stutter` to instead store:

```
bottom [3, 3, 7, 7, 1, 1, 14, 14, 9, 9] top
```

Notice that you must preserve the order of the stack. If the original stack had a 9 on top of the stack, the new stack should have two 9s on top.

9. GCD: The greatest common divisor (often abbreviated to GCD) of two nonnegative integers is the largest integer that divides evenly into both. In the third century BCE, the Greek mathematician Euclid discovered that the greatest common divisor of two numbers `x` and `y` can be computed as follows:

- If `x` is evenly divisible by `y`, then `y` is the greatest common divisor
- Otherwise, the greatest common divisor of `x` and `y` is equal to the greatest common divisor of the remainder of `x` divided by `y`

Use Euclid’s insight to write a recursive function `gcd` that computes the greatest common divisor of `x` and `y`.