CSM CS61B
Runtime Practice

Problem 1. Orders of Growth.

(a) Simplify using Big-O Notation: $N^2 + NlogN + 1000N + 6$

(b) Calculate the following limit: $\lim_{n\to\infty} \frac{n^2}{2^n}$

(c) What does the result of the above limit imply about the exponential and polynomial runtimes. *Hint: Is one always greater (grows faster) than the other?*

(d) True or False: If function f has $\mathcal{O}(n)$ and $\Omega(1)$, its tight bound is always found by taking the average of n and 1 giving: $\Theta(n)$

Problem 2. Iteration
What is the runtime of the following functions?

```
public static void loopingMore(int[] array) {
    int n = array.length;
    for (int i = 0; i < 3*n*n*n; i++) {
        System.out.println('I love you');
    }
}
```

```
public static void doubleLoopingHalf(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = n; j > 0; j = j / 2) {
            System.out.println('I love you');
        }
    }
}
```

```
public static void weirdLooping(int n) {
    for (int i = 0; i < n; i++) {
        int num = Math.pow(2, i + 1) - 1;
        for (int j = 0; j < num; j++) {
            System.out.println('I love you');
        }
    }
}
```

Problem 3. Recursion

Practice using the Work Per Layer Summation Formula

```
public static void mergeSort(int[] arr, int l, int r) {
    if (l < r) {
        // Compute Middle Index
        int m = (l + r) / 2

        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r); // Runs in linear time
    }
}
```

```
public static void thinTree(int n) {
    if (N <= 1) {return;}
    else {
        thinTree(2);
        thinTree(n / 2);
    }

}
```

```
public static void splitter(int n) {
    if (N == 1) {return;}
    else {
        int i = 0;
        while (i < n) {
            i++
            System.out.println(i);
        }
        return splitter(n / 3) + splitter(2n / 3);
    }

}
```

Problem 4. Mutual Recursion
The following problem has been adapted from Prof. Hug's Sp15 Midterm 2

State the runtime of each function start with f1.

```
public static void f1(int n) {
    for (int i = 0; i < 2*n; i++) {
        System.out.println('Welcome');
    }
}
```

```
public static void f2(int n) {
    if (n == 0) { return; }
    f2(n / 3);
    f1(n);
    f2(n / 3);
    f1(n);
    f2(n / 3);
}
```

```
public static void f3(int n) {
    if (n == 0) { return; }
    f3(n - 1);
    f1(16);
    f3(n - 1);
}
```

```
public static void f4(int n) {
    if (n == 0) { return; }
    f4(n - 1);
    f1(16);
    f1(n);
    f4(n - 1);
}
```

```
public static void f5(int n, int m) {
    if (m <= 0) {
        return;
    } else {
        for (int i = 0; i < n; i++) {
            f5(n, m-1);
        }
    }
}
```