Problem Set 4, Part I

Problem 1: Rewriting a method

1-1)

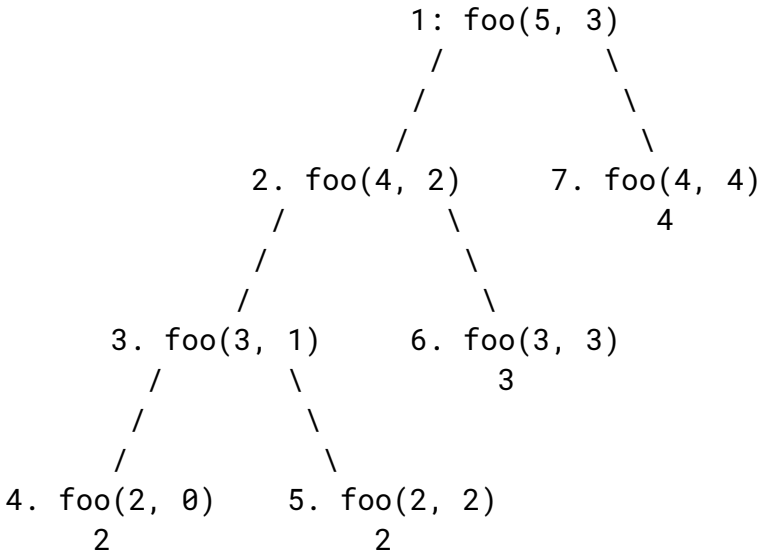
```
public static boolean search(Object item, Object[] arr) {
    if (arr == null) {
        throw new IllegalArgumentException();
    }
    for (int i = 0; i < arr.length; i++) {
        if (arr[i].equals(item)) {
            return true;
        }
    }
    return false;
}
```

1-2)

```
public static boolean search(Object item, Object[] arr, int start) {
    if (arr == null) {
        throw new IllegalArgumentException();
    }
    if (start < arr.length) {
        if (arr[start] == item) {
            return true;
        } else {
             return search(item,arr,start + 1);
        }
    }
    return false;
}
```

Problem 2: A method that makes multiple recursive calls

2-1)



2-2)

```
Call 4 (foo(2,0)) returns 2
Call 5 (foo(2,2)) returns 2
Call 3 (foo(3,1)) returns 4
Call 6 (foo(3,3)) returns 3
Call 2 (foo(4,2)) returns 7
Call 7 (foo(4,4)) returns 4
Call 1 (foo(5,3)) returns 11
```

Problem 3: Sum generator

3-1)

The exact formula for the number of times that the line that increases the sum is executed is (n(n+1))/2.

3-2)

The time efficiency of the method generateSums is quadratic. In big-O notation, this can be described as $O(n^2)$. To describe the time complexity of this method, we can notice that the result of the method prints an arithmetic sequence. Furthermore, for n number of sums, every sum goes through n sum calls. Additionally, we focus on the largest term and disregard its coefficient when evaluating big-O notation.

3-3)

```
public static void sumGen(int n) {
    int sum = 0;
    int timer = 1;
    while (timer <= n) {
        sum += timer;
        timer++;
        System.out.println(sum);
    }
}
```

3-4)

The time efficiency of the alternative implemented method above called sumGen would be linear, or in big-O notation, O(n). Breaking down the code, we can see that the sum line is executed a total of n times as there is only one while loop and the sum line is executed once per iteration. This is known as a linear relation.