Problem Set 4, Part I

Problem 1: Rewriting a method

1-1)

```
public static boolean search(Object item, Object[] arr) {
     // Always check for null references first!
     if (arr == null) {
         throw new IllegalArgumentException();
     }

     for (int i = 0; i< arr.length; i++) {
         if (arr[i].equals(item)) {
             return true;
         }
     }

     return false;
}
```

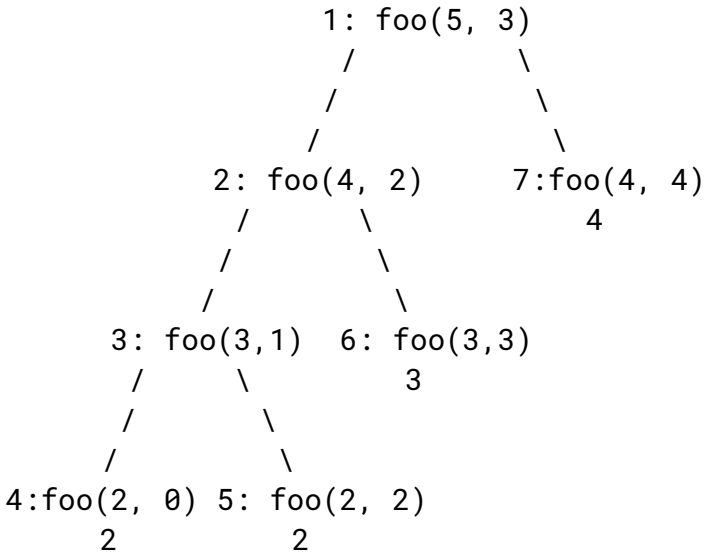
1-2)

```
public static boolean search(Object item, Object[] arr, int start) {
     // Always check for null references first!
         if (arr == null) {
             throw new IllegalArgumentException();
         }
         if(start == arr.length) {
             return false;
         }else {
             if(arr[start].equals(item)) {
                 return true;
              }else {
                 return search(item,arr,start +1);
             }
         }

}
```

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) n(n+1) / 2

3-2) O(n^2), since the function n(n+1)/2 = (n*2 + n)/2 = n*2/2 + n/2,the biggest power of the function is n*2/2 so the run time is O(n^2)

3-3)

```
static void generateSums(int n) {
      int sum = 0;
      for (int i = 1; i <= n; i++) {
            sum = sum + i;
            System.out.println(sum);
      }
}
```

3-4)

The time efficiency of the alternative implementation is O(n).
Since the new generateSums function only goes through the for loop as n times so the run time of new generateSums n time → O(n).