Homework 1

1. Consider the following iterative function:

```
int square(int n) {
    int result = 0;
    for (int i = 1; i <= n; i++)
        result += 2 * i - 1;
    return result;
}
```

Rewrite the function square using recursion and add preconditions and postconditions as comments. Then prove by induction that the recursive function you wrote is correct.

// Precondition: $n > 0$

```
int square(int n) {
    if (n <= 0) {
        return 0;
    }
    return square(n - 1) + (2 * n - 1);
}
```

// Postcondition: returns n^2

Inductive proof:

Base case

$n = 0$

$square(0) = 0$

Assumption

$square(k) = k^2$

$k = n - 1$

$square(k) = (n-1)^2$

Function space definition

$square(n-1) = square(n-1) * (2n-1)$

$square(n) = (n-1)^2 + (2n-1) = (n-1)(n-1) + (2n-1) = 2n^2 + 2$

2. Suppose the number of steps required in the worst case for two algorithms are as follows

   Algorithm 1: $f(n) = 3n^2 + 9$
   Algorithm 2: $g(n) = 51n + 17$

   Determine at what integer value of n, algorithm 2 becomes more efficient than algorithm 1.

$$f(n) = 3n^2 + 9$$
$$g(n) = 51n + 17$$

Algorithm 2 is more efficient when $g(n) < f(n)$

$$h(n) = f(n) - g(n)$$
So, $g(n)$ is more efficient when $h(n) > 0$

Set equal to each other

$$3n^2 + 9 = 51n + 17$$
$$3n^2 - 51n - 8 = 0$$

Solve by quadratic

$$\frac{-(-51) \pm \sqrt(51^2 - 4 * 3 * -8)}{6}$$
$$\frac{51 \pm \sqrt(2601 + 96)}{6}$$
$$\frac{51 \pm 51.93}{6} = 17.05, -0.15$$

At $n = 17$ they are both equal, therefore at $n = 18$ $g(n)$ is more efficient

---

3. Given the following function that sorts an array of values:

```java
void bubbleSort(double[] array) {
    for (int i = 0; i < array.length; i++)
        for (int j = array.length - 1; j > i; j--)
            if (array[j] < array[j - 1])
                swap(array, j, j - 1);
}
```

Let n be the length of the array. Using summation evaluation, determine the number of swaps that are performed in the worst case as a function of n.

Determine the inner count

inner loop range: [n-1, i+1]
iteration count:
count = (start - end) + 1
count =

$$(n-1)-(i+1)+1$$
$$(n-1-i-1)+1$$
$$(n-2-i)+1\$$$
$$n-1-i$$

$$f(i)=n-1-i$$

Total worst case swaps

$$T(n)=f(0)+f(1)+\ldots+f(n-1)$$
$$T(n)=(n-1)+(n-2)+\ldots+1+0$$

Summation Series (Arithmetic)

$$\sum_{i=0}^{n-1} i=\frac{n(n-1)}{2}$$

---

4. Given the following recursive function and its corresponding helper function that returns the sum of all the elements of an array that are located at even subscripts:

```
int sumEvenElements(int array[], int i) {
    if (i >= array.length)
        return 0;
    return array[i] + sumEvenElements(array, i + 2);
}

int sumEvenElements(int array[]) {
    return sumEvenElments(array, 0)
}
```

Assume n is the length of the array. Find the initial condition and recurrence equation that expresses the execution time for the worst case of the recursive function and then solve that recurrence.

indices = $\left\lceil \frac{n}{2} \right\rceil$

$T(0) = b_0$ (empty array immediate return)
$T(1) = b_1$ (array[x] ➡ one)