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.

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.

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

```
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.

4. Given the following recursive function and it 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.