Problem 1. Consider the following code fragment.

```plaintext
for i = 1 to n do
    for j = i to 2*i do
        output 'foobar'
```

Let $T(n)$ denote the number of times 'foobar' is printed as a function of $n$. Express $T(n)$ as a summation (actually two nested summations), and give a closed-form solution for $T(n)$ (i.e. no summations) by simplifying your summation. Show your work.

Problem 2. Consider the following code fragment.

```plaintext
for i = 1 to n do
    for j = i to 2*i do
        for k = 2*n-j+1 to 2*n do
            output 'foobar'
```

Let $T(n)$ denote the number of times 'foobar' is printed as a function of $n$. Express $T(n)$ as a summation (actually three nested summations), and give a closed-form solution for $T(n)$ (i.e. no summations) by simplifying your summation. Show your work.

Problem 3. Consider the following program for sorting:

```plaintext
i ← 1
while i < n do
    if A[i] > A[i+1] then
        A[i] ↔ A[i+1]
        i ← 1
    else
        i ← i + 1
end if
end while
```

(a) What is the worst case input for comparisons. (No justification needed.)

(b) Write a summation (actually nested summations) for how many comparisons the algorithm uses in the worst case. Hint 1: The algorithm has a final pass to confirm that the list is sorted. Leave this pass out of your summation (but do not drop the term). Hint 2: Try some small examples (n=1,2,3,4 etc.).

(c) Simplify your summation.