1. Consider the following code fragment.

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

   (a) Give a double summation for the exact number of times 'foobar' is printed.

   (b) Simplify your summation. Show your work.

2. Consider the following code fragment.

```
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'
```

   (a) Give a triple summation for the exact number of times 'foobar' is printed.

   (b) Simplify your summation. Show your work.

3. Consider a bubble-sort-like algorithm for sorting a list L with an even number of elements n: Group the elements into $n/2$ pairs indexed by (1,2), (3,4), ..., $(n-1, n)$. Sort each pair. Then run "bubble sort" treating each pair as a single element, but whenever there is a comparison in standard bubble sort, sort the two pairs into two new pairs, leaving the two smaller elements in the earlier pair and the two larger elements in the later pair. Since every pair is always sorted, this sorting – or merging – of two pairs can be done with at most three comparisons. Pseudo-code on back.

   (a) Assume $n = 4$. What is the best-case number of comparisons? Just state the number and show your input. Otherwise, no justification needed.

   (b) Assume $n = 4$. What is the worst-case number of comparisons? Just state the number and show your input. Otherwise, no justification needed.

   (c) Assume $n = 6$. What is the best-case number of comparisons? Just state the number and show your input. Otherwise, no justification needed.

   (d) Assume $n = 6$. What is the worst-case number of comparisons? Just state the number and show your input. Otherwise, no justification needed.

   (e) Make the simplifying assumption that every merge takes exactly three comparisons in the worst case. Write a formula with a double summation for the exact number of comparisons the algorithm uses in the worst case (for n even).

   (f) Simplify your summation. Show your work.

   (g) How does the number of comparisons compare with bubble sort?

```
i ← 1
while i < n do
    if L[i] > L[i+1] then L[i] ↔ L[i+1]
    i ← i + 2
end while
for i = n/2 downto 2 do
    j ← 1
    while (j+1)/2 < i do COMMENT: merge (L[j],L[j+1]) with (L[j+2],L[j+3])
        A ← L[j]; B ← L[j+1]; C ← L[j+2]; D ← L[j+3]
        if A<C then
            L[j] ← A
            if B<C then
                L[j+1] ← B; L[j+2] ← C; L[j+3] ← D
            else
                L[j+1] ← C
                if B<D then
                    L[j+2] ← B; L[j+3] ← D
                else
                    L[j+2] ← D; L[j+3] ← B
                end if
            end if
        else
            L[j] ← C
            if D<A then
                L[j+1] ← D; L[j+2] ← A; L[j+3] ← B
            else
                L[j+1] ← A
                if B<D then
                    L[j+2] ← B; L[j+3] ← D
                else
                    L[j+2] ← D; L[j+3] ← B
                end if
            end if
        end if
        j ← j + 2
    end while
end for
```