HW1
CMSC351
Jee Kang

Problem 1

(a) A best case scenario where each pair is out of order would look something like: $[20, 10, 30, 40, 60, 50]$ for $n = 6$.

Bubble sort would then only need to swap the first element with the second element in each pair, meaning that for n, the number of exchanges done by bubble sort would be equal to the number of pairs in the array, which equals $\frac{n}{2}$.

The number of exchanges is the best case is therefore: $\frac{n}{2}$

(b) A worst case scenario for $n = 6$ where each pair is in order would look something like: $[50, 60, 30, 40, 10, 20]$.

When bubble sort iterates through this array, each element in the ordered pair is swapped the same amount of times. For $n = 6$, the second element is the largest element so it is exchanged 4 times, to be placed at the n position. The second time the outer for loop is executed, the first element, 50 is exchanged once again 4 times to be placed at the $n - 1$ position. This means that the sum of the exchanges can be calculated for $n/2$, then the resulting values can be multiplied by 2 to calculate the total number of exchanges.

$$2 \cdot \sum_{i=2}^{\frac{n}{2}} \sum_{j=1}^{2i-2} 1$$

The upper bound for the second summation is multiplied by $2i - 2$ since we only need to calculate once per pair, we choose the even is. When i is even in the inner most loop, the largest value is located at position 2, meaning the exchanges start at position 2 not 1. This means we have to take the orignal worst case number of exchanges for the inner most loop, $i - 1$ and subtract it by 1 again, to give $i - 2$. Since we divided n by 2 in the outer most loop, we make i to $2i$ to restore it to it's original value.

$$2 \cdot \sum_{i=2}^{\frac{n}{2}} \sum_{j=1}^{2i-2} 1 = 2 \cdot \sum_{i=2}^{\frac{n}{2}} 2i - 2 = 4 \cdot \sum_{i=2}^{\frac{n}{2}} i - 1 = 4(\sum_{i=1}^{\frac{n}{2}} i - 1 - 1 - 1) = 4(\sum_{i=1}^{\frac{n}{2}} i - 1) = 4 \cdot \sum_{i=1}^{\frac{n}{2}} i - 4 \cdot \sum_{i=1}^{\frac{n}{2}} 1$$

$$= 4 \cdot \frac{(\frac{n}{2})(\frac{n}{2} + 1)}{2} - 4 \cdot \frac{n}{2} = n(\frac{n}{2} + 1) - 2n$$

(c) The average case, given that the pairs are randomly ordered and randomly positioned in the array, is the same as the average case for a standard bubble sort:

$$= \frac{(n)(n+1)}{4}$$

This works because regardless of the fact that the numbers are organized in ordered pairs, there is still a $\frac{1}{2}$ chance of an element in any given position being greater than the element to its immediate right, and being swapped. If the chosen element is to the left of it's given pair, there are two cases: it is larger than its corresponding pair to its right (**swap**), or it is smaller than its corresponding pair to its

right (**no swap**). If the chosen element is to the left of another, separate pair, there is also two cases: the pair to the right of the element is greater than the pair to which the chosen element belongs (**no swap**), or the pair to the right of element is less than the pair to which the chosen element belongs (**swap**).

Problem 2

(a) For a best case scenario where there are arbitrary groups of size k, the sample array would look something like: $[30, 20, 10, 60, 50, 40]$ for $k = 3$ and $n = 2k$

Before with the pairs, we said that the bubble sort would only need to swap once per pair, meaning that there would only be $\frac{n}{2}$ swaps. Now we need to find how many swaps there would be for each group of k elements, then multiply the result by $\frac{n}{k}$, which equals the number of groups of size k in the array. Taking the first part of the sample array, $[30, 20, 10]$, we see that this is just a worst case scenario for a bubble sort. Since we know in the worst case, a reverse ordered array, bubble sort takes $\frac{(n)(n+1)}{2}$ exchanges. Since in a group size k, $n = k$, the formula becomes $\frac{(k)(k+1)}{2}$. Multiply this number of exchanges by the number of groups in the set, and you get:

$$\frac{n}{k} \cdot \frac{(k)(k+1)}{2} = \frac{(n)(k+1)}{2}$$

(b) Looking at the logic from 1a., we said that for groups consisting of ordered pairs, the worst case can be calculated by:

$$2 \cdot \sum_{i=2}^{\frac{n}{2}} \sum_{j=1}^{2i-2} 1$$

This was calculated by using the logic that every element in the pair under goes the same number of exchanges in the bubble sort. For groups size k, the same logic holds: every element in a group sized k undergoes the same number of swaps as the other elements in the same group. Therefore the number of exchanges only has to be calculated once per group, then multiplied by the number of elements in the group.This means we can use the same summation we formulated in 1a, except we count $\frac{n}{k}$ outer loops. This gives the summation:

$$k \cdot \sum_{i=2}^{\frac{n}{k}} \sum_{j=1}^{ki-k} 1 = k \cdot \sum_{i=2}^{\frac{n}{k}} ki - k = k^2 \cdot \sum_{i=2}^{\frac{n}{k}} (i-1) = k^2 \cdot \sum_{i=1}^{\frac{n}{k}} (i-1) - 1 - 1 = k^2 \left(\sum_{i=1}^{\frac{n}{k}} i - \sum_{i=1}^{\frac{n}{k}} 1\right) = k^2 \left(\frac{(\frac{n}{k})(\frac{n}{k}+1)}{2} - \frac{n}{k}\right)$$

$$= \frac{(k)(n)(\frac{n}{k}+1)}{2} - (k)(n)$$

Problem 3

```
def bubbleSort(arr[] array, int n) {
    if n == 1
        return array
    else
        for j 1 to n - 1
            if array[j] > array[j+1] do
                swap
            end
        end
        bubbleSort(array, n - 1)
    end
}
```