Problem Set 2 **Name:** Hudson Arney

Submission instructions: Your written answers should be submitted to Canvas as a PDF.

1. For each expression below, give two valid examples of big-o notation for that expression. One example should be consistent with convention (e.g., a tight bound) and one should violate convention (e.g., an unnecessarily loose bound).

	Tight Bounds	Unnecessarily Loose Bound
a) $64n^2 + 3n - 2$	$O(n^2)$	$O(n^3)$
b) $6n + 2$	$O(n)$	$O(n^2)$
c) $n + \log n + 2$	$O(n)$	$O(n^2)$

2. Illustrate the steps of insertion sort on the following lists. You should show the key, which elements are shifted, and the list after the insertion.

a) [5, 4, 3, 2, 1]

Step 1: k=4, [4, 5, 3, 2, 1]

Step 2: k=3, [3, 4, 5, 2, 1]

Step 3: k=2, [2, 3, 4, 5, 1]

Step 4: k=1, [1, 2, 3, 4, 5]

b) [5, 2, 3, 4, 8, 1, 2]

Step 1: k=2, [2, 5, 3, 4, 8, 1, 2]

Step 2: k=3, [2, 3, 5, 4, 8, 1, 2]

Step 3: k=4, [2, 3, 4, 5, 8, 1, 2]

Step 4: k=8, [2, 3, 4, 5, 8, 1, 2]

Step 5: k=1, [1, 2, 3, 4, 5, 8, 2]

Step 6: k=2, [1, 2, 2, 3, 4, 5, 8]

3. For each function below, answer the following questions.

```
public static int findMinimum(int[] input) {
     int minimum = Integer.MAX_VALUE;
     for(int i = 0; i < input.length; i++) {
          if(input[i] < minimum) {
               minimum = input[i];
          }
     }
     return minimum;
}
```

a) Trace the code for following input lists: [1, 2, 3, 4, 5], [4, 2, 3, 8, 1], [5, 4, 3, 2, 1]. Write the values of **`minimum`** and **`i`** at each step.

b) Describe the general idea behind the function in words.

c) What loop invariant is maintained?

d) Write the equation for the number of instructions executed in terms of the array size using exact analysis.

e) Express your answer from (d) in big-o notation.

a) [1,2,3,4,5] 1: i=0, min=1 2: i=1, min=1 3: i=2, min=1 4: i=3, min=1 5: i=4, min=1

[4,2,3,8,1] 1: i=0, min=4 2: i=1, min=2 3: i=2, min=2 4: i=3, min=2 5: i=4, min=1

[5,4,3,2,1] 1: i=0, min=5 2: i=1, min=4 3: i=2, min=3 4: i=3, min=2 5: i=4, min=1

b) Look through all elements in the array to achieve the minimum

c) Minimum always holds the minimum value in the array from index 0 to i during the iteration

d) $C_1 + C_2 + n(C_3 + C_4 + C_5 + C_6) + C_7 = 4n+3$ ⇒ worst case

e) O(n), the number of instructions executed because it grows with the size of the input array.

4.

```
public static int slowMaximumIndex(int[] input) {
    int maximum = -1;
    for(int i = 0; i < input.length; i++) {
        boolean larger = true;
        for(int j = 0; j < input.length; j++) {
            if(input[i] < input[j]) {
                larger = false;
            }
        }
        if(larger) {
            maximum = i;
            break;
        }
    }
    return maximum;
}
```

a) Trace the code for following input lists: [1, 2, 3, 4, 5], [4, 2, 3, 8, 1], [5, 4, 3, 2, 1]. Write the values of **maximum**, **i, and larger** after each iteration of the outer loop.

b) Describe the general idea behind the function in words.

c) What loop invariant is maintained?

d) Write the equation for the number of instructions executed in terms of the array size using exact analysis.

e) What is the run time in the best case? What is the run time in the worst case?

f) Express your answers from (e) in big-o notation.

a) [1,2,3,4,5] 1: i=0, max=0, lar=false 2: i=1, max=0, lar=false 3: i=2, max=0, lar=false 4: i=3, max=0, lar=false 5: i=4, max=4, lar=true

[4,2,3,8,1] 1: i=0, max=0, lar=false 2: i=1, max=0, lar=false 3: i=2, max=0, lar=false 4: i=3, max=0, lar=false 5: i=4, max=4, lar=true

[5,4,3,2,1] 1: i=0, max=0, lar=false 2: i=1, max=0, lar=false 3: i=2, max=0, lar=false 4: i=3, max=0, lar=false 5: i=4, max=0, lar=false

b) Find the index of the max element by using nested for loops

c) "maximum" contains the index of the max element

d) Worst: $4n^2+7n+4$

e) Best: larger is true in the first iteration
Worst: larger is true in the last iteration

f) Worst: $O(n^2)$, Best: $O(n)$

5.

```
public static void partitionBySign(int[] input) {
    int i = 0;
    for(int j = 0; j < input.length; j++) {
        if(input[j] < 0) {
            int tmp = input[i];
            input[i] = input[j];
            input[j] = tmp;
            i++;
        }
        System.out.println(i + " " + j);
        System.out.println(Arrays.toString(input));
        System.out.println();
    }
}
```

Worst case is if input[i] < 0, $O(n^2)$

Best case is $O(n^2)$, if input[i] ≥ 0 ∀i ⇒ x = 0 ⇒ $T(n) \in O(n^2)$

a) Trace the code for following input lists: [-1, -2, 0, 4, 5], [4, -2, 0, 8, -1], [5, 4, 0, -2, -1]. Write the values of **i**, **j**, and **input** at each step.

b) Describe the general idea behind the function in words.

c) What loop invariant is maintained?

d) Write the equation for the number of instructions executed in terms of the array size using exact analysis.

e) Express your answer from (d) in big-o notation.

a) [-1,-2,0,4,5] 1) j=1, i=0, input=[-2,-1,0,4,5] → 3) j=2, i=1, input=[-2,-1,0,4,5]

1) j=0, i=0, input=[-1,-2,0,4,5]

4) j=3, i=1, input=[-2,-1,0,4,5] → 5) j=4, i=2, input=[-2,-1,0,4,5]

[4,-2,0,8,-1]

1) j=0, i=0, input=[4,-2,0,8,-1] → 2) j=1, i=0, input=[-2,4,0,8,-1] → 3) j=2, i=1, input=[-2,4,0,8,-1]

4) j=3, i=1, input=[-2,4,0,8,-1] → 5) j=4, i=2, input=[-2,4,-1,8,0]

[5,4,0,-2,-1]

1) i=0, j=0, input=[5,4,0,-2,-1] → 2) j=1, i=0, input=[5,4,0,-2,-1] → 3) j=2, i=0, input=[5,4,0,-2,-1]

4) i=0, j=3, input=[-2,4,0,5,-1] → 5) i=1, j=4, input=[-2,-1,0,5,4]

b) To partition the input array into a negative numbers part and non-negative numbers part

c) all elements from 0 to i-1 are negative or zero, all from i to j-1 are not

d) $10n + 2$

e) $O(n)$

6.

```
SELECTION-SORT(A)
n = A.length
for j=1 to n - 1
    smallest = j
    for i in j + 1 to n
        if A[i] < A[smallest]:
            smallest = i
    tmp = A[j]
    A[j] = A[smallest]
    A[smallest] = tmp
```

$1+1+n(1+n(1+1+1+1+1)+1+1+1+1)+1$

$3+6n+5n^2$

a) Trace the code for following input lists: [1, 2, 3, 4, 5], [4, 2, 3, 8, 1], [5, 4, 3, 2, 1]. Write the values of **j** and **A** after each iteration of the outer loop.

b) Describe the general idea behind the function in words.

c) What loop invariant is maintained?

d) Write the equation for the number of instructions executed in terms of the array size using exact analysis.

e) Express your answer from (d) in big-o notation.

f) Will selection sort execute fewer instructions on a sorted list than a reverse sorted list? How does this compare with insertion sort?

7. Identify and describe real-world problems that can be solved using the insertion sort algorithm.

Insertion sort can be used for sorting names in a database, if new names get added they can be automatically sorted.

6) [1,2,3,4,5] ↦ 1) j=1, A=[1,2,3,4,5] → 2) j=2, A=[1,2,3,4,5] → 3) j=3, A=[1,2,3,4,5]
→ 4) j=4, A=[1,2,3,4,5]

[4,2,3,8,1] → 1) j=1, A=[1,2,3,8,4] 2) j=2, A=[1,2,3,8,4] 3) j=3, A=[1,2,3,8,4]
4) j=4, A=[1,2,3,8,4]

[5,4,3,2,1] ↦ 1) j=1, A=[1,4,3,2,5] → 2) j=2, A=[1,2,3,4,5] → 3) j=3, A=[1,2,3,4,5]
→ 4) j=4, A=[1,2,3,4,5]

b) It finds the smallest element in the unsorted portion of the array and swaps it with the element at the begining of the unsorted portion.

c) A[0:j] is sorted, all elements in A[j+1:n] are greater than A[0:j]

d) $5n^2+6n+3$

e) $O(n^2)$

f) The same # of instructions on a sorted list, which is not as efficient as insertion sort