FALL 2019 - CS 303 Algorithms and Data Structures
Exam-1

50 points. 75 minutes. Closed book/notes. Name: ___________________________

1. For each of the following program fragments, determine the time complexity using the Θ notation. Explain your analysis next to each program fragment **[3*5 = 15 points]**

	Program Fragment	Time Complexity (Big O Notation)	Explanation

1	for (int i=0; i<n; i++){
	for (int j=0; j<n; j=j+2){
	a = a + i + j;
	}
	}		

2	for (int i=0; i<n; i++){
	a = a + i;
	}
	for (int j=0; j<n; j=j*2){
	a = a + j;
	}		

3	for (int i=0; i<n*n; i++){
	for (int j=0; j<n; j=j+3){
	a = a + i + j;
	}
	}		

4	for (int i=0; i<n; i++){
	for (int j=n; j>0; j--){
	a = a + i*j;
	}
	}		

5	int a=0,i=n;
	while(i>0){
	a=a+i;
	i=i/2;
	}		

2. Show the operation of sorting the following sequence using **insertion sort**. [**6 points**]
   Arr = [15, 22, 4, 55, 66, 28, 11, 95, 51]

   The insertion sort algorithm is given below.

INSERTION-SORT(A, n)

for $j = 2$ **to** n
    $key = A[j]$
    // Insert $A[j]$ into the sorted sequence $A[1 \mathinner{.\,.} j-1]$.
    $i = j - 1$
    while $i > 0$ and $A[i] > key$
        $A[i+1] = A[i]$
        $i = i - 1$
    $A[i+1] = key$

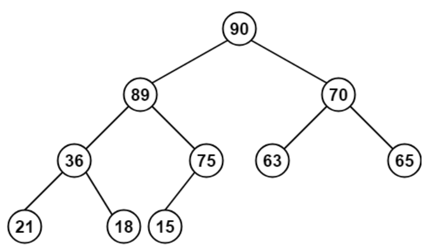
3. Given the following max-heap, show the operation of sorting using **heapsort**. [**8 points**]

   The **heapsort** algorithm is given below.

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

```
BUILD-MAX-HEAP(A)
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```



4. Show the operation of sorting the following sequence using quicksort. [**6 points**]
   Arr = [12 ,144, 3, 96, 111, 515, 175, 132, 25]

   *** **Show your pivot value for each step** *****

   The quicksort algorithm is given below.

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

5. Illustrate the operation of **bucket sort** on the following sequence; [**6 points**]

   Arr = [0.75, 0.21, 0.33, 0.36, 0.2, 0.66, 0.98, 0.17, 0.73]

   The algorithm for bucket sort is given below.

```
BUCKET-SORT(A, n)
  let B[0..n - 1] be a new array
  for i = 1 to n - 1
      make B[i] an empty list
  for i = 1 to n
      insert A[i] into list B[⌊n · A[i]⌋]
  for i = 0 to n - 1
      sort list B[i] with insertion sort
  concatenate lists B[0], B[1], ..., B[n - 1] together in order
  return the concatenated lists
```

6. What is the major difference between Quick Sort and Merge Sort? Explain it **[2 points]**

7. What are the properties that a binary tree has to satisfy to be a **heap**? Explain them **[5 points]**

8. What are the best/average/worst case time complexities for the Quick Sort? (Regular implementation) **[2 points]**