ECE368 Fall 2016 Homework 8

IMPORTANT:

- Do NOT leave your name or Purdue ID on this homework.
- Write your homework security number at the TOP of EACH page.

Read and sign the ***Academic Honesty Statement*** that follows:

"In signing this statement, I hereby certify that the work on this exercise is my own and that I have not copied the work of any other student while completing it. I understand that, if I fail to honor this agreement, I will receive a score of zero for this exercise and will be subject to further disciplinary action."

`Homework security number:`

Please acknowledge any people who have helped you with this homework.

Question	Credits
1	
2	
3	

1. (20 points) Professor Jones has proposed a new sorting algorithm (pseudocode is given below):

```
Jones_Sort (A, i, j)
  if (A[i] > A[j])
    A[i] ↔ A[j]                 //swap element i with element j
  if (i+1) ≥ j
    return
  k = ⌊(j-i+1)/3⌋
  Jones_Sort (A, i, j-k)        //first two-thirds
  Jones_Sort (A, i+k, j)        //last two-thirds
  Jones_Sort (A, i, j-k)        //first two-thirds again
```

1(a). (5 points) If `A[]={6, 3, 2, 5}`, show the array after each swap when `Jones_Sort(A, 0, 3)` is called.

1(b). (10 points) Determine the worst-case time complexity of `Jones_Sort()` using `O()` notation in terms of the size of the array, `n`.

1(c). (5 points) How does this compare with the worst-case of bubble sort and heap-sort? Is this an efficient algorithm in terms of time?

2. (20 points) As we saw in class, the Mergesort algorithm relies on the `Merge()` procedure that merges two sorted arrays into a single sorted array. A commonly encountered problem is one of merging ‘‘`k`’’ sorted arrays into a single sorted array, where ‘‘`n`’’ is the total number of elements in all the input arrays together. Design the most efficient algorithm that you can to do this? (*Hint: Think about using a min-heap.*)

3. (30 points)

3(a). (15 points) Illustrate the operation of Bucket Sort on the following array `A = {0.79, 0.13, 0.16, 0.64, 0.39, 0.20, 0.89, 0.53, 0.71, 0.42}`. You are free to pick the number of buckets you want as well as the function that puts data into the buckets (e.g., you can use the function given in the slides).

3(b). (15 points) Explain why the worst-case running time of Bucket Sort is $O(n^2)$. What simple change can you make to the algorithm to make its worst case running time $O(nlogn)$?

4. (30 points + 10 points extra credit) You are given a *sorted* array `A[]` that consists of `n` integer elements. You are also given an integer `z`. Design an algorithm to determine whether or not there are two elements in `A[]` whose sum is exactly `z`. What is the time complexity of your algorithm in terms of `O()` notation? **Extra credit:** You will receive **10 extra points** if your algorithm can do this in `O(n)` time.