ES242 **Name:** _____
30/9/2018 **Roll Number** _____
Time Limit: 120 Minutes

Important Instructions

1. **Please write your answers in the space provided in this paper only.**
2. **You can first do the rough work on the rough sheets provided to you and when you are confident about your answer, then you can write the answer in the main answer script.**
3. **Please write pseudocode for the algorithmic questions asked in the paper. Also prove the correctness and running time of your algorithm.**
4. **There is no need to give proof of algorithms done in the class**

Grade Table (for instructor use only)

Question	Points	Score
1	3	
2	9	
3	6	
4	10	
Total:	28	

---

1. (3 points) **Sorting**

   You are given an array A of n integers (you do not know the range of these n numbers). Your job is to arrange these numbers such that all the negative integers come before all the positive integers. For example, if the input is $A = [1, 4, -2, 5, -9, 7, -8]$, then one of the valid answer is $[-2, -9, -8, 4, 5, 7]$. Given an $O(n)$ time algorithm for this problem that uses only $O(1)$ extra space.

.

2. **Stable Sorting**

A sorting algorithm is called stable if if two objects with equal keys appear in the same order in sorted output as they appear in the input array. For example, assume that the following names are to be sorted with respect to their age:

$A = [(Ashish, 20), (Bijoy, 30), (Charu, 15), (Dev, 30), (Sayan, 5)]$, then in the stable sorting algorithm $Bijoy$ should always come before Dev. Which of the following algorithm is stable (only write yes or no as the answer, no need to write justification)

(a) (1 point) Insertion Sort

(b) (1 point) Heap Sort

(c) (1 point) Merge Sort

(d) (1 point) Quick Sort

(e) (5 points) Given a sorting algorithm which is not stable show how can you change it to make it stable without increasing its asymptotic worst case running time.

.

3. **Odd Even Merge**

Assume that we have to merge two sorted array A_1 and A_2 of size n, say for example $A_1 = [2, 6, 10, 11]$ and $A_2 = [4, 7, 9, 15]$. To this end, we use the following non-trivial (and probably nonsensical at first reading) algorithm. We first merge all the odd elements of A_1 and A_2 to get $A_{odd} = [2, 4, 9, 10]$. Then we merge all the even elements to get $A_{even} = [6, 7, 11, 15]$. We now create a new array A such that all the odd elements of A come from A_{odd} and even elements come from A_{even}. Formally if $A_{odd} = [a_1, a_2, \ldots, a_n]$ and $A_{even} = [b_1, b_2, \ldots, b_n]$, then $A = [a_1, b_1, a_2, b_2, \ldots, a_n, b_n]$. In our example, $A = [2, 6, 4, 7, 9, 11, 10, 15]$. Note that A is not a sorted.

(a) (1 point) Prove that the smallest number in A_{odd} is the smallest number in $A_1 \cup A_2$.

(b) (5 points) Show how will you convert A into a sorted array using at most $n/2$ comparisons and swaps. Prove that your algorithm is correct.

.

4. **Basic Data-Structures: Linked List, Stacks, Queues**

   Assume that n buildings are built in a line from left to right. At the right of the n^{th} building, there is a river. From the top of some buildings you can see the river. For example, from the n^{th} building, you can see the river. From the top of i^{th} building, you can see the river if there is no building j $(j > i)$ such that the height of the j^{th} building is greater than the height of i^{th} building. For every building from which you cannot see the river, you have to find the nearest building to its right that has a greater height.

   Formally, you are given a unsorted array of n distinct heights. For each index i, you have to find the the smallest index greater than i such that $A[j] > A[i]$. If no such j exists for i, then your algorithm should print that"River can be seen from building i".

   (a) (2 points) Give an $O(n^2)$ time algorithm for this problem.

   (b) (8 points) Give an $O(n)$ time algorithm for this problem.

.

.