CS 5720 Design and Analysis of Algorithms
Homework #4

Submission requirements:

- Submit your work in PDF format to the appropriate assignment on Canvas.
- **5% extra credit** if your writeup is *typed*.

Assignment:

1. Determine whether each of the following recurrence relations can be solved using the Master Theorem. For those that can be solved with the Master Theorem, use the Master Theorem to determine the function's order of growth in terms of $\Theta(\cdot)$. For those than cannot be solved with the Master Theorem, explain why. For each, you can assume that $T(1) = 1$.

   (a) $T(n) = 5T(n/3) + n$
   (b) $T(n) = 2.7T(n/5) + n^2$
   (c) $T(n) = 2T(n-1) + n$
   (d) $T(n) = 1.1T(0.2n) + 1$
   (e) $T(n) = 2T(n/2) + n \log_2 n$
   (f) $T(n) = 2T(n/2) + \sqrt{n}$
   (g) $T(n) = 4T(n/2) + \sqrt{n^4 - n + 10}$
   (h) $T(n) = 7T(n/3) + \sum_{i=1}^{n} i$
   (i) $T(n) = 4T(n/2) + n^n$
   (j) $T(n) = 8T(n/3) + n^3$

2. For each of the following divide-and-conquer algorithms, use the Master Theorem to determine the algorithm's worst-case order of growth in terms of $\Theta(\cdot)$. What is a worst-case input to the algorithm? What is the best-case complexity of each algorithm, and what is a best-case input? Note: the notation $A[i \,..\, j]$ means "the subarray of A with first index i and last index j."

(a)

Algorithm 1

```
1: function REC1(A[0..n − 1])
2:     if n == 1 then return A[0]
3:     else
4:         if REC1(A[0 .. ⌊n/3⌋])< 0 then return A[0]
5:         else return REC1(A[⌊n/3⌋ .. 2⌊n/3⌋])
6:         end if
7:     end if
8: end function
```

(b)

Algorithm 2

```
1: function REC2(A[0..n − 1])
2:     if n == 1 then return A[0]
3:     else return REC2(A[0 .. ⌊n/2⌋ − 1]) + REC2(A[⌊n/2⌋ .. n − 1])
4:     end if
5: end function
```