CSE 317 Design and Analysis of Algorithms	*Assigned: Wednesday, 24 Jan*
Problem Set 1	
Spring 2024	*Due: 11:59 pm Monday, 5 Feb*

To facilitate grading and timely feedback
please note that all submissions are through Gradescope.

Solve each problem on a new page and put your name on each page.
You MUST typeset your solutions using LaTeXand correctly link
solutions on Gradescope.

1. Choose the correct option. You must provide reasoning. Incorrect reasoning for correct choice will get zero credit.
   (a) Suppose algorithm A runs in $\Theta(n^2)$ time for all inputs of size n and algorithm B runs in $\Theta(2^n)$ for all inputs of size n. A is faster than B:
      i. Always
      ii. Never
      iii. On a finite number of inputs
      iv. On all but at most a finite number of inputs
      v. None of the above is necessarily true
   (b) The recursion $T(n) = 1$ for any $n \leq 10$ and $T(n) = n^3 + n^{2.93} + 8T(n/2)$ solves to:
      i. $\Theta(n^3)$
      ii. $\Theta(n^3 \log n)$
      iii. $\Theta(n^3 (\log n)^{2.93})$
      iv. $\Theta(n^{2.93})$
      v. $\Theta(n^{2.93+3})$

2. Rank the following functions by order of growth, that is, find an arrangement $g_1, g_2, \ldots, g_{25}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{24} = \Omega(g_{25})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

$(3/2)^n$	$(\sqrt{2})^{\log n}$	$\log^* n$	n^2	$(\log n)!$
n^3	$\log^2 n$	$\log(n!)$	2^{2^n}	$n^{1/\log n}$
$\log\log n$	$n2^n$	$n^{\log\log n}$	$\ln n$	2^n
$2^{\log n}$	$(\log n)^{\log n}$	$4^{\log n}$	$(n+1)!$	$\sqrt{\log n}$
$n!$	$2^{\sqrt{2\log n}}$	n	$n \log n$	1

3. A carnival game consists of three dice in a cage. A player can bet a dollar on any of the numbers 1 through 6. The cage is shaken, and the payoff is as follows. If the player's number does not appear on any of the dice, he loses his dollar. Otherwise, if his number appears on exactly k of the three dice, for $k = 1, 2, 3$, he keeps his dollar and wins k more dollars. What is his expected gain from playing the carnival game once?

4. Although merge sort runs in $\Theta(n \log n)$ worst-case and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small n. So, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using standard merging mechanism, where k is a value to be determined.

   (a) Show that n/k sublists, each of length k can be sorted by insertion sort in $\Theta(nk)$ worst-case time.

   (b) Show that the sublists can be merged in $\Theta(n \log(n/k))$ worst-case time.

   (c) Given that the modified algorithm runs in $\Theta(nk + n\log(n/k))$ worst-case time, what is the largest asymptotic value of k as a function of n for which the modified algorithm has the same asymptotic running running time as standard merge sort?

   (d) How should k be chosen in practice?

5. Consider the following problem.

   Input: n^2 distinct numbers in some arbitrary order.

   Output: an $n \times n$ matrix containing the input numbers, and having all rows and all columns sorted in increasing order.

   Example: $n = 3$, so $n^2 = 9$. Say the 9 numbers are the digits $1, 2, \ldots, 9$. Possible outputs include:

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 1 & 4 & 5 \\ 2 & 6 & 7 \\ 3 & 8 & 9 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 1 & 3 & 4 \\ 2 & 5 & 8 \\ 6 & 7 & 9 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \text{or} \quad \ldots$$

   It is clear that we can solve this problem in time $2n^2 \log n$ by just sorting the input (remember that $\log(n^2) = 2\log n$) and then outputting the first n elements as the first row, the next n elements as the second row, and so on. Your job in this problem is to prove a matching $\Omega(n^2 \log n)$ *lower bound* in the comparison-based model of computation, i.e., a lower bound of $cn^2 \log n$ for some constant $c > 0$ that is independent of n. For simplicity, you can assume n is a power of 2.

   Hint: You may use the facts that $m! > \left(\frac{m}{e}\right)^m$ and that and comparison based sorting algorithm must make at least $\log(n!) = \Omega(n \log n)$ comparisons.