1. **Multiple Choice:** Each question below has one of the following answers.

(a) $\Theta(1)$  
(b) $\Theta(\log n)$  
(c) $\Theta(n)$  
(d) $\Theta(n \log n)$  
(e) $\Theta(n^2)$

For each question, write the letter that corresponds to your answer. You do not need to justify your answers. Each correct answer earns you 1 point, but each incorrect answer costs you $\frac{1}{2}$ point. You cannot score below zero.

(a) What is $\sum_{i=1}^{n} \frac{n}{i}$? 
(b) What is $\sum_{i=1}^{n^2} 2^{-i}$? 
(c) How many digits do you need to write $\lceil \log(n!) \rceil$ in decimal? 
(d) What is the solution of the recurrence $T(n) = 16T(n/5) + n^2$? 
(e) What is the solution of the recurrence $T(n) = 7T\left(\lceil \frac{n+51}{7} \rceil - \sqrt{n}\right) + 19n - 2^{\log^* (n^2)} + 6$? 
(f) What is the best-case running time of randomized quicksort? 
(g) What is the worst-case running time of randomized quicksort? 
(h) The expected time for inserting one item into a treap is $O(\log n)$. What is the worst-case time for a sequence of $n$ insertions into an initially empty treap? 
(i) The amortized time for inserting one item into an $n$-node splay tree is $O(\log n)$. What is the worst-case time for a sequence of $n$ insertions into an initially empty splay tree? 
(j) How much time does it take to compute the first 50 digits of $\pi$?

2. Here is an alternative algorithm for generating a random permutation of $n$ numbers $1, \ldots, n$ (in the following $M \geq n$):

```plaintext
FINDPOSITION(B[1, \ldots, M], num):
  do
    pos ← RAND(1, M)
    while B[pos] > 0
    B[pos] ← num
```

```plaintext
RANDOMPERMUT(M, n):
  for i = 1 to M do
    B[i] ← 0
  for i = 1 to n do
    FINDPOSITION(B[1, \ldots, M], i)
  for i = 1 to M do
    if (B[i] ≠ 0)
      print( B[i] )
```

In the following, assume that RAND(1, M) runs in $O(1)$ time.

(a) (3pt) What is the expected running time of FINDPOSITION, if exactly $k$ positions in the array $B[1, \ldots, M]$ are zero?
(b) (3pt) What is the expected running time of \texttt{RandomPermut}(2n, n) (as a function of \(n\))?

(c) (4pt) What is the expected running time of \texttt{RandomPermut}(n, n) (as a function of \(n\))?  

3. You are given an array \(A[1, \ldots, n]\) of real numbers, and a sorted array \(I[1, \ldots, k]\) of distinct integer numbers (i.e., indices) between 1 and \(n\). Describe an algorithm \texttt{MultiSelect}(\(A[1, \ldots, n], I[1, \ldots, k]\)) that outputs the \(I[1]\)-th smallest number in \(A\), the \(I[2]\)-th smallest number in \(A\), ..., \(I[k]\)-th smallest number in \(A\). Thus, for \(A = [1005, 3432, 234, 12, 90239]\) (i.e., \(n = 5\)), and \(I = [1, 3]\) (i.e., \(k = 2\)), \texttt{MultiSelect} would output 12, 1005.

In the following, you can assume that \(k < n\).

(a) (2pt) Describe an \(O(n \log n)\) algorithm for this problem.

(b) (8pt) Describe an \(O(n \log k)\) algorithm for this problem. You can assume that you have a procedure \texttt{Select}(\(A[1, \ldots, n], r\)) which returns the \(r\)-th smallest element in \(A[1, \ldots, n]\) in \(O(n)\) time. (If you have to use functions/algorithms we already saw in class, just mention it, do not provide pseudo-code) Hint: Use divide and conquer on the indices array \(I[1, \ldots, k]\).

4. You have \(n\) objects that you wish to put in order using the relations “<” and “=”. For example, with three objects 13 different orderings are possible.  
   \[
   \begin{align*}
   a &= b = c, & a &= b < c, & a &= b < c, & a &= c < b, & b &= a = c, & b &= a < c, & b &= c < a, \\
   b &= c < a, & c &= a = b, & c &= a < b, & c &= b < a.
   \end{align*}
   \]

Give and analyze a dynamic programming algorithm that can calculate, as a function of \(n\), the number of different possible orderings. Your algorithm should take \(O(n^2)\) time and \(O(n)\) space.

Hint: Consider how many distinct values do you have in your ordering. For example \(a = d < b = c\) has only two distinct values (i.e., \(a\) and \(c\). Consider how this number of distinct values changes when you insert \(c\)).

5. Describe how to implement a heap that supports \texttt{insert} and \texttt{delete-min}. You are being told in advance that the sequence of operations on this heap is going to be made out of \(n\) \texttt{inserts}, and (at most) \(k\) \texttt{delete-min} operations, where \(k \ll n\).

(a) (4pt) Show how to implement such a heap, such that \texttt{insert} operation and \texttt{delete-min} operation takes \(O(\log k)\) expected time.

(b) (6pt) Assume that you are given a data-structure \(X\) that supports:
   
   i. \texttt{insert} in \(O(1)\) worst-case time.
   
   ii. \texttt{delete-min} in \(O(\log n)\) worst-case time (where \(n\) is the number elements currently in \(X\)).
   
   iii. You can extract all the elements stored in \(X\), in time linear in the number of elements currently stored in it.

Show how to implement a new heap such that it performs \texttt{insert} in \(O(1)\) amortized time, and \texttt{delete-min} operation in \(O(\log k)\) amortized time, using \(X\).