1. (6 points) The time taken to INSERTIONSORT an array is proportional to the number of inversions in the array. Assume that our sample space is the set of permutations of the $n$ numbers, and assume that each permutation is equally likely (it is drawn from a uniform distribution over the set of permutations).

   a. What is the probability that $A$ has $\frac{n(n-1)}{2}$ inversions?

   b. What is the probability that $A$ has 0 inversions?

   c. What is the probability that $A$ has exactly 1 inversion?

2. Consider the following code fragment to find the Biggest and Second biggest elements of an array $A[1..n]$ of distinct integers (assume that $n>1$):

   ```
   else {Big$\leftarrow A[1]$; Second$\leftarrow a[2]$;}
   for $i \leftarrow 3$ to $n$ do
     if Second$<A[i]$ then if Big$<A[i]$ then {Second$\leftarrow$ Big; Big$\leftarrow A[i]$;}
   else Second$\leftarrow A[i]$
   ```


   b. (6 points) Develop a recurrence for $w_n$, the number of worst-case instances (instances forcing the code fragment to execute the worst-case number of pairwise comparisons). Note that $w_2 = 2$ and $w_3 = 4$ (corresponding to the instances $(1,2,3), (2,1,3), (1,3,2)$ and $(3,1,2)$). Solve the recurrence.

   c. (3 points) What is the best-case number of pairwise comparisons executed by this code fragment? For arbitrary $n$, what input instances yield this best case number of comparisons?

   For parts $d$, $e$ and $f$, assume the sample space is the set of all permutations of $n$ distinct numbers and each input instance is equally likely. Verify your answers for $n=3$. 
d. (5 points) What is the probability that an input instance causes the code fragment to use exactly the best-case number of pairwise comparisons?
e. (4 points) What is the probability that an input instance causes the code fragment to use the worst-case number of pairwise comparisons?
f. (8 points) What is the expected number of pairwise comparisons?

3. (10 points) Given the input \( A[1..n] \) and integer \( k, 1 \leq k \leq n \), we seek the \( k \text{-th} \) smallest member of \( A \). Consider the following "algorithm" to find the \( k \text{-th} \) smallest member of \( A[lo..hi] \):

1) Partition \( A[lo..hi] \) around some element \( x \) at \( A[pivot] \),
   satisfying \( lo \leq pivot \leq hi \), \( A[i] < x \) for \( lo \leq i < pivot \), and \( A[i] \geq x \) for \( pivot < i \leq hi \).
2) Comparing \( k, lo, hi, \) and \( pivot \) it can be determined if the
    \( k \text{-th} \) smallest member of \( A[lo..hi] \) lies in \( A[lo..pivot-1] \)
    or is equal to \( A[pivot] \) or lies in \( A[pivot+1..hi] \). Recurse to 1).
A) Program the above "algorithm".
B) Check that your program works on a small problem like
   \[
   A = \begin{bmatrix}
   3 & 1 & 4 & 1 & 5 & 9 & 2 & 6 \\
   1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
   \end{bmatrix}
   \]
   by verifying that your program returns that the 3rd smallest member is 2
   and that the 8th smallest member is 9.
C) Profile your program to determine its average case run-time for
   finding the \( k \text{-th} \) smallest member of \( A[1..n] \) where the members of \( A \)
   are \( n \) randomly generated integers (drawn from a uniform distribution)
   and \( k \) is drawn from a uniform distribution over \{1,...,n\}.
   \textbf{Hint:} Your program will probably have average-case run time
   \( T(n) = c_1 n + c_2 \), where \( c_1 \) and \( c_2 \) are to be determined by you.

4. (8 points) Prove that \( \Omega(\lg n) \) comparisons are necessary to test if an element \( x \)
   belongs to an ordered list \( (a_1, a_2, ..., a_n) \).
1. **a.** \( A \) has \( \frac{n(n-1)}{2} \) inversions when it is sorted in decreasing order. Since exactly 1 of the \( n! \) permutations is sorted in decreasing order, its probability is \( \frac{1}{n!} \).

**b.** \( A \) has 0 inversions when it is sorted in increasing order. Since exactly 1 of the \( n! \) permutations is sorted in increasing order, its probability is \( \frac{1}{n!} \).

**c.** If the elements of \( A \) are \( \{1, 2, \ldots, n\} \), then the \( n-1 \) permutations with exactly 1 inversion are \( \langle 2, 1, 3, 4, \ldots, n-1, n \rangle, \langle 1, 3, 2, 4, \ldots, n-1, n \rangle, \langle 1, 2, 4, 3, \ldots, n-1, n \rangle, \ldots, \langle 1, 2, 3, 4, \ldots, n, n-1 \rangle \).

The probability of drawing one of these permutations is \( \frac{n-1}{n!} \).

2. **a.** \( A[1] < A[2] \) is always executed exactly one time, and \( \text{Second} < A[i] \) is always executed exactly \( n-2 \) times. The comparison \( \text{Big} < A[i] \) is executed \( n-2 \) times (every time through the \( \text{for} \)-loop) if \( A \) is sorted in increasing order. Hence the worst-case number of pairwise comparisons is \( 2n-3 \).

**b.** For \( n > 2 \), there are \( w_{n-1} \) worst-case arrangements of the first \( n-1 \) numbers. For \( \text{Second} < A[n] \) to be satisfied, then either \( A[n] = n \) or \( A[n] = n-1 \). Hence,

\[
w_n = \begin{cases} 
2, & \text{if } n = 2 \\
2w_{n-1}, & \text{if } n > 2 
\end{cases}
\]

which has the closed form solution \( w_n = 2^{n-1} \).

**c.** Combined, \( A[1] < A[2] \) and \( \text{Second} < A[i] \) are always executed exactly \( n-1 \) times. In the best-case, \( \text{Second} < A[i] \) is always violated (so \( \text{Big} < A[i] \) is never executed), so the best-case number of pairwise comparisons is \( n-1 \). This happens exactly when \( A[1] \) and \( A[2] \) are the two largest elements of \( A \).

**d.** The fewest pairwise comparisons occur when \( A[1] \) and \( A[2] \) are the two largest elements of \( A \), which is exactly when \( A[1] \) is the largest element and \( A[2] \) is the second largest (with probability \( 1/n(n-1) \)) or when \( A[2] \) is the largest element and \( A[1] \) is the second largest (with probability \( 1/n(n-1) \)). These two events are mutually exclusive (hence we can add their probabilities) to yield the probability of a best-case number of pairwise comparisons to be \( \frac{2}{n(n-1)} \).

**e.** Of the \( n! \) input instances, there are \( 2^{n-1} \) worst-case instances. Hence, the probability of an input instance being worst-case is \( \frac{2^{n-1}}{n!} \).
Let random variable $X$ denote the number of pairwise comparisons, and let random variables $X_k$, $n \geq k \geq 3$, denote the number of pairwise comparisons involving $A[k]$. Note that for input instance $s$, $X_k(s) \in \{1, 2\}$, and $X = 1 + \sum_{n \geq k \geq 3} X_k$. The solution to the problem is $E[X] = E\left[1 + \sum_{n \geq k \geq 3} X_k\right] = 1 + E\left[\sum_{n \geq k \geq 3} X_k\right] = 1 + \sum_{n \geq k \geq 3} E[X_k]$. For $n \geq k \geq 3$, $E[X_k] = 1 \cdot \Pr\{X_k = 1\} + 2 \cdot \Pr\{X_k = 2\} = 1 \cdot \frac{k-2}{k} + 2 \cdot \frac{2}{k} = \frac{k+2}{k}$. Finally, $E[X] = 1 + \sum_{n \geq k \geq 3} \frac{k+2}{k} = 1 + \sum_{n \geq k \geq 3} 1 + 2 \sum_{n \geq k \geq 3} \frac{1}{k} = 1 + (n-2) + 2\left(\sum_{n \geq k \geq 1} \frac{1}{k} - \frac{3}{2}\right) = n + 2H_n - 4$.

4. Consider an adversary which responds to an algorithm’s query in the following manner: Let $(a_1, a_2, \ldots, a_n)$ be the elements of $(a_1, a_2, \ldots, a_n)$ which may equal $x$

- Initially $m=n$, finally $m \leq 1$
- When the algorithm asks to compare $x$ to $a_j$, the adversary replies

  - if $\left|\{a_1, \ldots, a_{j-1}\}\right| > \left|\{a_{j+1}, \ldots, a_m\}\right|$ then $x < a_j$
  - else $x > a_j$

We note that with the adversary’s algorithm, $m$ the size of the set of candidates to be equal to $x$, can not shrink too quickly. Initially $m=n$, finally $m \leq 1$, and with each query from the algorithm, the new size of $m$ is at least $\frac{n}{2}$ of the old size of $m$. The algorithm will need at least $\lg n$ steps to reduce $m$ from $n$ to $1$. 