1. (a) Let \( A \) and \( B \) be two \( n \) element arrays, both of which have been sorted into increasing order. Prove that the number of ways in which \( A \) and \( B \) can be merged to form one sorted array of size \( 2n \) which is also in increasing order is \( \binom{2n}{n} \). (10 points)

(b) Use the result of Part (a) to show that any comparison-based algorithm that correctly merges two sorted arrays of size \( n \) into a single sorted array of size \( 2n \) must use at least \( 2n - o(n) \) comparisons in the worst-case. (15 points)

Note: For Part (b), you may use Stirling’s approximation: For any positive integer \( k \),
\[
\sqrt{2\pi k} \left(\frac{k}{e}\right)^k \leq k! \leq \sqrt{2\pi k} \left(\frac{k}{e}\right)^k + 1.
\]

2. Suppose you have a “black-box” median finding algorithm \( A \) that returns the lower median of any set of \( m \) distinct numbers in \( O(m) \) worst-case time. Let \( S \) be set containing \( n \) distinct numbers. Use \( A \) to develop an \( O(n) \) algorithm for finding the \( i \)th smallest number in \( S \), for a given \( i \) (1 \( \leq i \leq n \)).

Your answer should provide a high-level description of your algorithm, an explanation of why it is correct, a recurrence for its running time and a proof that the running time is indeed \( O(n) \). (25 points)

Note: Since \( A \) is a “black-box” algorithm, you cannot make any changes to it.

3. A device receives a sequence of numbers \( \langle x_1, x_2, x_3, \ldots \rangle \), one at a time. (Such a sequence is called a data stream.) As the device can only store one number at any time, we using the following strategy:

When the first value appears, it is stored in memory. For any \( k \geq 2 \), when the \( k \)th value appears, it replaces the value in memory with probability \( 1/k \).

Show that at each step \( k \), when the above strategy is used, the stored value is uniformly distributed over all the values seen so far. (That is, prove that at each step \( k \), \( \Pr\{\text{Stored value} = x_i\} = 1/k \), for any \( i, 1 \leq i \leq k \).) (20 points)
4. A fast food company is trying to decide locations for opening new stores along a highway (which can be assumed to be a straightline). The company has identified $n$ potential locations $v_1, v_2, \ldots, v_n$ for the restaurants. At each location, only one restaurant can be opened. For each pair of successive locations $v_i$ and $v_{i+1}$, the distance $d(v_i, v_{i+1})$ between those locations is available ($1 \leq i \leq n - 1$). Further, for each $i$, the profit $p_i$ that the company would make by placing a restaurant at $v_i$ is also known ($1 \leq i \leq n$). The company wants to choose a subset of $V = \{v_1, v_2, \ldots, v_n\}$ for new restaurants with the additional condition that each pair of restaurants is separated by a distance of at least $k$. (The value of $k$ is also known.) For a subset $S \subseteq V$ of chosen locations, the total profit is the sum of the profits of the locations in $S$.

**Example:** In the following example, there are four potential locations ($v_1$ through $v_4$). Distances between successive locations are shown on the edges. The profit for each location is shown below the location. Let the required minimum distance between two restaurants be 10.

\[
\begin{array}{cccc}
v_1 & 4 & v_2 & 5 \\
100 & & 150 & 75 \\
\end{array}
\]

For the above example, the maximum profit is 200 and this is obtained by placing restaurants at locations $v_2$ and $v_4$.

Give a dynamic programming algorithm that computes the maximum total profit that the company can achieve. The running time of your algorithm must be a polynomial in $n$.

You must clearly specify what information is stored in the table used for dynamic programming and how the entries in the table are computed. Be sure to include (high level) pseudocode for your algorithm and explain why the running time of your algorithm is a polynomial in $n$. *You won’t receive any credit if your algorithm is incorrect or its running time is not a polynomial in n.* (30 points)

**Hint:** Consider the $n$ subproblems obtained by choosing the first $i$ locations $v_1, v_2, \ldots, v_i$, where $1 \leq i \leq n$. 