1. Two standard six-sided dice are thrown. Let \( X_1 \) and \( X_2 \) denote the random variables corresponding to the values shown by the two dice.

(a) Let \( X = X_1 + X_2 \). Compute the conditional probability \( \Pr\{X = 7 \mid X_1 \text{ is even}\} \). (8 points)

(b) Let \( Y = (X_1 + X_2) \mod 2 \). Compute \( \mathbb{E}[Y] \). (12 points)

2. Let \( X \) and \( Y \) be independent geometric random variables with success probabilities \( p \) and \( q \) respectively. (Thus, \( X \) (\( Y \)) represents the number of trials until the first success in a Bernoulli trial with success probability \( p \) (\( q \)).) Compute \( \Pr\{X = Y\} \). Be sure to explain how you arrived at the result. (25 points)

3. Suppose we have \( b \) bins denoted by \( B_1, B_2, \ldots, B_b \) and we toss a total of \( b \) balls into these bins. Assume that for each toss, the ball is equally likely to land in any of the \( b \) bins. (That is, for any toss, the probability that the ball lands in \( B_i \) is \( \frac{1}{b} \), \( 1 \leq i \leq n \).)

(a) Consider bin \( B_1 \). Find the probability that exactly \( k \) of the \( b \) balls fall into \( B_1 \), where \( 1 \leq k \leq b \). (10 points)

(b) Compute the expected number of bins that contain exactly one ball. (15 points)

4. Let \( P[1..n] \) be an array each of whose elements contains an integer value (which may be positive, negative or zero). A subarray \( P[i .. j] \) of \( P \), where \( i \leq j \), consists of the elements \( P[i], P[i+1], \ldots, P[j] \). We say that subarray \( P[i .. j] \) is monotone if \( P[i] \leq P[i+1] \leq P[i+2] \ldots \leq P[j] \). Assume that a subarray consisting of just one element is monotone. This problem asks you to devise an \( O(n) \) algorithm that determines the length of a longest monotone subarray of a given array. The algorithm must also compute the indices of such a subarray in \( O(n) \) time.

Your answer must include the following:

(i) A clear description of the data structure used by the algorithm.

(ii) Pseudocode for the algorithm along with an explanation of why the running time is \( O(n) \).

(iii) A clear explanation of why your algorithm is correct. (Ideally, this should be a rigorous proof of correctness.)

You won't receive any credit if your algorithm is incorrect or its running time is asymptotically worse than \( O(n) \). (30 points)