HOMEWORK No. 2 [50 points]

Textbook: Problems 8.7, 9.3 **[10 points each]**
Note: In problem 8.7, in each time slot, the two opponents throw their dices at the same time, *i.e.,* the battleships fire in parallel and not sequentially.

Problem H2.1 [10 points] Consider a wireless channel that alternates between "Good" and "Bad" states, and that is being used to transmit a digital signal. From monitoring the channel over an extended period of time, we determine that whenever ever the channel is in its "Bad" state, the transmitted bit is corrupted with probability 1, while the transmitted bit is received correctly with probability 1 when the channel is in its "Good" state. In addition, we also determine that channel transitions between "Good" and "Bad" states in consecutive bit transmission slots depend only on the current state, and obey the following statistics:

$$
\begin{aligned}
P(\text{channel is in "Bad" state in slot } i+1 \mid \text{ channel is in "Bad" state in slot } i) &= \beta \\
P(\text{channel is in "Good" state in slot } i+1 \mid \text{ channel is in "Bad" state in slot } i) &= 1-\beta \\
P(\text{channel is in "Good" state in slot } i+1 \mid \text{ channel is in "Good" state in slot } i) &= \alpha \\
P(\text{channel is in "Bad" state in slot } i+1 \mid \text{ channel is in "Good" state in slot } i) &= 1-\alpha
\end{aligned}
$$

1. Construct a Markov chain that captures the evolution over time of the channel, and use it to compute as a function of α and β the stationary probabilities that the channel is in a "Good" state and a "Bad" state, respectively.
2. Assuming that data is transmitted in words of 8 bits, compute as a function of α and β the probabilities $P(0), P(1), P(>1)$, of $0, 1$, and more than 1 errors, respectively, in a random 8-bit word. For the purpose of this derivation, assume further that the channel is in a random state at the start of a codeword, *i.e.,* the probability that the channel is in a given state when the first bit is transmitted is the stationary probability for that state.
   Hint: Enumerate for each possible starting state the channel patterns needed to generate the required number of errors.

Problem H2.2 [20 points] Consider a packet processing facility that consists of two packet header processors in series. The first performs a CRC computation on the packet header, while the second does an address lookup on the destination address carried in the packet header.

A new packet arrives in each clock cycle with probability p, and is queued if either of the two processors is already busy processing a packet. In other words the "server" consisting of the two processors can only handle one packet at the time. When both processors become idle, the first packet waiting at the head of the packet queue (note that this could be an arriving packet if the system was empty) is sent first to the CRC processor, and next to the address lookup processor if the CRC is correct.

From observation, we know that the CRCs of incoming packets are correct with probability q, independently of each other. CRC computations take only one clock cycle, and the number of clock cycles (memory access) that an address lookup takes is geometrically distributed with an average of L clock cycles.

1. What is the distribution of a packet processing time, *i.e.,* the probability that the processing of a packet takes $k \geq 1$ clock cycles?
2. Give a condition, function of p, q, and L for the system to be stable.
3. Derive expressions, function of p, q, and L for the expected number of packets in the CRC processor and in the address lookup processor, respectively. In other words, what is the fraction of the time that either processor is busy processing a packet.

4. Formulate a Markov chain model that captures the system behavior. Identify explicitly what you define as the system state as well as the transition probabilities between all states.
   Hint: The state cannot be just the number of packets in the system since you also need to remember whether a packet is in the CRC or in the address lookup processor. But this can be readily handled by keeping track of both the number of packets in the queue, and which processor is currently handling the packet in service.