Goal: The goal of this project is to get started with simulating communication systems and to experiment a little bit with simple and error-detecting and error-correcting codes.

Assignment: Write a program to simulate a binary communication system that uses a simple error-control coding scheme. The problem can be broken down into several discrete but interconnecting parts:

- **The input stream:** The input stream is a binary information source with equal probability of 0 and 1. The bit values are independent from one bit to the next. Bits will be input in 3-tuples to the encoder, so it is most convenient to generate them in sets of 3.

- **The encoder:** We are going to generate random codes here. In order to constrain the code space, we only consider systematic codes, which means that the codeword contains the input word. For uniformity with other students’ work, let the input word be the last 3 bits of the codeword. There will be eight codewords in each code (corresponding to inputs 000, 001, ..., 111). Generate the remaining two coded bits of each codeword at random for each code.

- **The channel:** We will test the codes for 2 binary symmetric channels with crossover probabilities 0.2 and 0.05. An easy way to generate the errors is to generate a stream of 0s and 1s where the probability of a 1 is equal to the crossover probability. Then add them to the codeword (which is also 0s and 1s) using modulo 2 arithmetic or exclusive-or the two streams.

- **The decoder:** The decoder chooses the maximum likelihood codeword based on Hamming distance. Hamming distance is just the number of bits that differ between the received word and a potential codeword. To do the decoding, compute the Hamming distance for every codeword in the code. This decoder should provide two outputs. The first is an error detection signal. If there are no codewords at Hamming distance 0, then an error is detected; otherwise no error is detected. The second output is the error-corrected output. Choose one of the codewords with minimum Hamming distance. Resolve ties in whatever way you prefer. Since it is a systematic code, drop the first two bits of the codeword to yield the information sequence to be output from the decoder.

- **The performance metrics:** We want to evaluate the codes based on two performance metrics: probability of undetected error (note the change) and probability of bit error. **For probability of undetected error, count the total number of received words that are in error that are a codeword (i.e., are at Hamming distance 0 from a codeword). Also count the total number of received words that have at least one error caused by the channel.**
Divide the first by the second to estimate the probability of undetected error. For probability of bit error, compare the output of the decoder to the original information sequence. Count the number of bit errors. Divide by the total number of bits to get an estimate of the probability of bit error at the output of the decoder. Compare to the channel error rate, which is the crossover probability.

Evaluate the performance of 20 different codes for each value of the crossover probability. Use at least 100,000 information bits for each simulation. Identify the best code(s), along with their probability of error detection and probability of bit error.

Addendum: Also evaluate the performance of (at least) 20 linear codes. For linear codes, the sum (modulo-2, i.e., 1+1=0) of any other two codewords is a codeword. For a linear code, it is sufficient to design the three codewords of the form xx001, xx010, and xx100, and then compute all the other codewords as the linear sums of these. For instance, if two codewords are 11001 and 01100, then the codeword for information sequence 101 is 10101. Note that 00000 is always a codeword because xx001+xx001=00000.