Write a program (in any programming language) to understand and demonstrate how error correcting works under different BER values. Specifically, use the \((n,k)\) block coding, with \(n=5\) and \(k=2\) as we discussed in class (this example is in the textbook, too. See page 198).

You program will have three major parts (functions) and you will call these functions back-to-back in your main program:

**Sender:** asks user to enter BER and a message (i.e., a stream of characters), for example

```
> Enter BER (or Pb): 0.0001
> Enter your message: this is my hw
```

This function encodes that message using \((n,k)\) block coding and creates a frame (say `sent_frame`). Basically, you will deal with the bit representation of the message and replace every two bits with the corresponding 5-bit codeword.

**Transmitter:** takes `sent_frame` as a parameter and creates another same size frame called `received_frame` by flipping every bit in the `sent_frame` with the given probability \(P_b\) or BER. Then this function prints out the `sent_frame` and `received_frame` on the screen.

**Receiver:** takes `received_frame` as a parameter and tries to correct/detect errors so that it can figure out what was the original message sent... this function should print out which errors are corrected or just detected but cannot be corrected. At the end, by comparing the original message and the received message, determine wrongly corrected parts of the message if any.

**What to return (Submission will be done using WebCT):**

Run your program for several cases with different BER (or Pb). Specifically try \(10^{-6}\), \(10^{-5}\), \(10^{-4}\), \(10^{-3}\), \(10^{-2}\), and comment on how the increase in BER affects the performance of \((n=5,k=2)\) block coding.

*Using WebCT, submit your source code and the output file with your comments on the performance of \((n,k)\) block coding as BER increases.*