* 1. 0000\_0000 0000\_01000 00010\_1000 0000\_0000 (flag is detected between stuffed data and end flag. Actual end flag is not noticed)
  2. Stuff a 1 after every 0. 50% efficiency.
  3. - not possible for a flag to occur between two stuffed bits  
     0011\_1100 0011\_11100 1111\_0011 11100\_1111 0011\_1100   
     This would not be possible because it would stuff this false flag  
       
     - By the same logic, it is not possible to create a false flag between stuffed boundaries  
       
     - It is not possible to create a false flag between the last few bits of the data and the first few bits of the end flag.  
     0011\_1100 0000\_1111 0011\_1100  
     This is impossible because the stuffer would add a 1 after the “00\_1111”
  4. Doesn’t work.  
     0011\_1100 1100\_1111 0011\_1100  
     This results in a false flag forming between the last few bits of the data and the first few bits of the end flag.

1. 1. CRC = 01011
   2. Sender: Total\_Message = Message + CRC = 10001011  
      Undetectable Error: Total\_Message + G(x) = 10111110
   3. This generator detects all 1-bit errors. The CRC polynomial has at least two terms, thus a single bit error will not be able to divide G(x).
   4. This generator does not detect all odd bit errors. X+1 is not a factor of G(x) and thus it is possible for the generator to not capture odd bit errors.
   5. To get the number 10-length burst error that is undetectable, we must find how many of these 10-length bursts are multiples of the Generator G(x). This means that we must multiply our generator with a term that is of the form where the middle few terms can be either a 1 or a 0. This means that there are different multiples that would result in a 10-length burst that is undetectable by the Generator G(x).
   6. There are different polynomial permutations that can be used to obtain the desired 10th powered polynomial from G(x). These are as follows (in bit representation):
      1. 10001
      2. 10011
      3. 10101
      4. 10111
      5. 11001
      6. 11011
      7. 11101
      8. 11111

1. 1. Yes, the sender needs to number the data frames.   
      Counterexample: Say the sender sends the first data frame and the receiver receives it. However, the ack that is sent is dropped. This will prompt the sender to resend that frame. If it is not numbered, the receiver will view it as a new frame and unknowingly have a duplicate.
   2. No, the receiver does not need to number the ack frames. This is because the frames are sent and received at predetermined discrete time intervals where confusion on which data frame the ack corresponds to is impossible.
   3. Once the sender recovers from the crash, it waits a single time unit to wait for all potential receiver acks to die out. Then, the sender should transmit a reset message that informs the receiver that a crash has occurred, and the transmission must be restarted. Upon reception of the reset message, the receiver should send a reset ack.
2. 1. The STATUS packet is needed in order to make sure that the receiver is receiving the message. For example, if the receiver has crashed, there is no way for the sender to know that the previous packets were received. The STATUS packet therefore ensures that the receiver is in fact still receiving all the sender’s packets by requiring a NACK.
   2. If there is unacknowledged data, it is the sender’s duty to poll the receiver’s status. This is obtained through the NACK message. If the status does not match up with the sender’s frame, then the sender will know to resend the unacknowledged packets.
   3. The timer should be stopped and restarted whenever a STATUS message is sent or the sender receives a NACK from the receiver.
   4. In the worst case, the sender will send the first packet, D, and the second packet. The second packet will be rejected because the first was dropped. The receiver will send NACK 0 to indicate that the sender must resend the first packet. This will reset the STATUS timer. After the timer ends, a STATUS message will be sent to the receiver, which should respond with a NACK. This NACK will inform the sender that the receiver got D.