**Problem: CRC**

You will be given two binary strings as input - the message and the divisor. You’ll need to calculate the Frame Check Sequence using [bitwise operators](https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/). Also, you’ll need to simulate how receiver using the CRC may detect any error in the frame. For this purpose, first calculate the frame and then randomly change 3 bits of it. Then show that receiver can detect the error using the same bitwise operation.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| Message: 1011011  Divisor: 1101 | Frame check sequence: 001  Frame: 1011011001  After randomly changing 3 bits:  1011110101  Remainder is 010. ERROR Detected. |

**Problem: Hamming Code**

You will be given a binary string - the message and parity type (odd or even) as inputs and will have to calculate the codeword for the message using Hamming code. You **must** use the [**bitwise operator**](https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/)**s** for this purpose. Also, you’ll need to simulaten how receiver using the hamming code may correct a single bit error in the fame. For this purpose, randomly change 1 bit of the frame and show that the receiver is able to correct it. Also change 2 bits of the actual frame and try to correct it in the same way. However, in this case, the main purpose will be showing that receiver can not correct errors of more than one bit using hamming code.

P.S: If someone is found to have implemented the code skeleton of previous semester, she/he will get **zero** in the project.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| Message: 10110110  Parity Type : e (for even) | Codeoword: 101110111000  Single Bit Error: 001110111000  Double Bit Error: 100110101000  For No Change in bit:  Error Detected in Position: 0  No Error in Codeword  For Single bit Error:  Error Detected in Position: 12  After Correcting, CODEWORD is: 101110111000  For Double bit Error:  Error Detected in Position: 15  After Correcting, CODEWORD is: 100110101000 |

**Problem: Sliding Window**

Hence, you will need to simulate a sequence of events of data transmission between one transmitter and one receiver using sliding window flow control with piggybacking. First take the frame sequence number length and window size as inputs. Then take the sequence of events as input. The sequence of events is expressed by several lines where each line denotes an event. It consists of sequence number of event, a character ‘X’ or ‘Y’ denoting the transmitting entity and another integer denoting the number of frames transmitted (if only acknowledgement is transmitted, then it will be 0). The input terminates with a -1. As output, you’ll need to show the sliding windows of X or Y(they will be identical and just the opposite) after each event. See the sample input and output for help.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 3 4  1 X 2  2 Y 1  3 X 3  4 Y 0  -1 | At the beginning  X Transmitting buffer: 0 1 2 3  X Transmitting Window: 0 1 2 3  X Receiving buffer: 0 1 2 3  X Receiving Window: 0 1 2 3  After event 1:  X Transmitting buffer: 0 1 2 3  X Transmitting Window: 2 3  X Receiving buffer: 0 1 2 3  X Receiving Window:0 1 2 3  After event 2:  X Transmitting buffer: 2 3 4 5  X Transmitting Window: 2 3 4 5  X Receiving buffer: 0 1 2 3  X Receiving Window: 1 2 3  After event 3:  X Transmitting buffer: 2 3 4 5  X Transmitting Window: 5  X Receiving buffer: 1 2 3 4  X Receiving Window: 1 2 3 4  After event 4:  X Transmitting buffer: 5 6 7 8  X Transmitting Window: 5 6 7 8  X Receiving buffer: 1 2 3 4  X Receiving Window: 1 2 3 4 |