CS2031
Telecommunication

Assignment #1

Student Number : 12307269

INTRODUCTION:

Our task was to implement a program that performs Cyclic Redundancy Check on a string for both the sending and receiving side of the communication channel. CRC is an error-detecting code designed to detect accidental changes to data. CRC is implemented by dividing the data repeatedly. The remainder is then send to get verified by performing the division calculation again, this time with the remainder added instead of zeroes. The remainder should equal zero if there are no detectable errors.

DESIGN:

I began this assignment by writing a pseudo-code for it.

```
//CRC pseudo-code
divider[] = number of bits want to enter;
data[1...n] = data entered;
//create array with added zeros;
 totalLength = data.length + divisor.length - 1;
crc[totalLength];              //crc[] will contain the data entered plus extra zeros.
for(i = 0; i < data.length; i++)
{
      crc[i] = data[i];
}

//division method

int current =1;                //start "dividing" at crc[1] because crc[0] contains the size
                               //of the original data entered.
while(crc.length -  current>= divisor.length)
{
    for(i=0; i<divisor.length;i++)
    {
       if(crc[i+current] == 0){
           divide by Zero[], an array containing zeros;
        }else{
            crc[i+current] = crc[i+current] ^ divisor[i]; // binary division use exclusive-OR
        }
        current++;     // division starts on the next array
    }
}
```

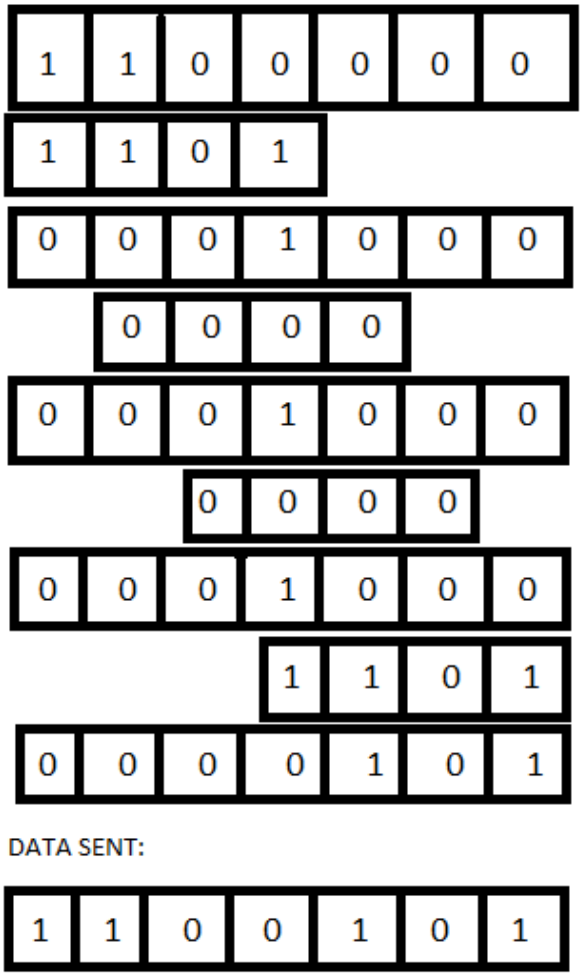
IMPLEMENTATION:

My program has a set divisor(byte divisor[] = { 1, 1 0, 1 }) so the only user input we want is the data that is to be divided by the default divisor. In the sender class, we use the divisor to divide the new data (crc[]) which contains the user input plus the extra zeros.

DIVISION CALCULATION:

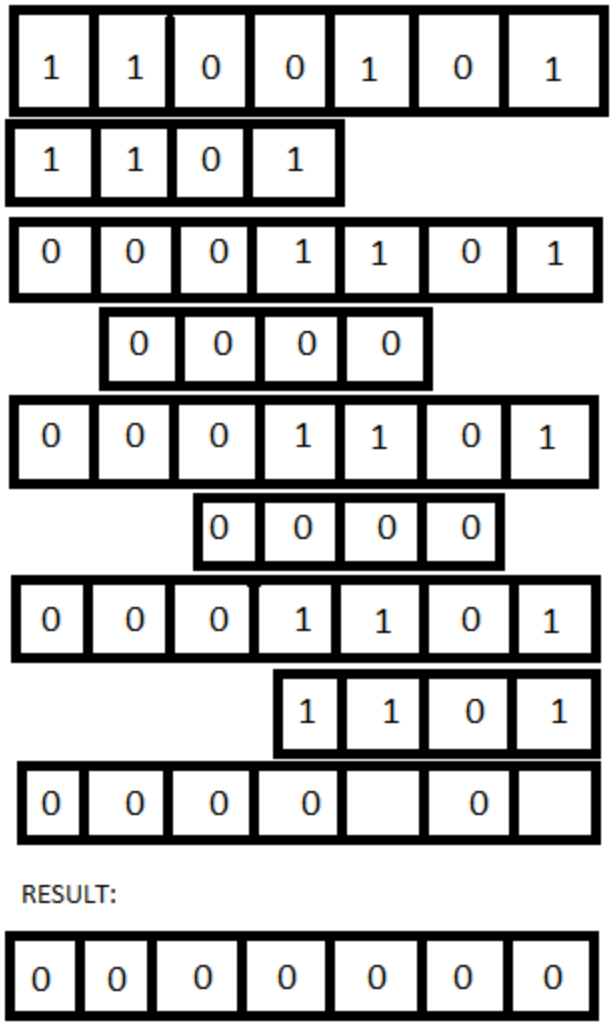
```
static byte[] divide(byte zeros[],byte divisor[], byte rem[])
{
      int cur = 1;       //start “dividing” at crc[1] because crc[0] contains the size
                                 //of the original data entered.
            while ((rem.length - cur) >= divisor.length) {

                  for (int i = 0; i < divisor.length; i++) {

                        if (i == 0 && (rem[cur + i] == 0)) {
                              rem[cur + i] = (byte) (rem[cur + i] ^ zeros[i]);
                        //divide by zeros using XOR operator
                        } else {
                              rem[cur + i] = (byte) (rem[cur + i] ^ divisor[i]);
                        //divide by divisor 1101  using XOR operator
                        }
                  }

                  cur++;          // division starts on the next array
            }
            return rem;
}
```

SENDER CALCULATION diagram:



-as you can see above, if crc[i] is equal to zero, we use an array of zeros as our divisor instead of 1101.

RECEIVER CALCULATION diagram:



Once we get the remainder, the data which contains the original input plus remainder is sent to the "Receiver" class. The same division method is then used and you should always get 0 as a result if there are no detectable errors.

Some examples of data I've tested:

Data entered: 1 Data+extra zeros = 1000
Remainder: 0101 Data sent: 1101
Result: 0000

Data entered: 100100000 Data+extra zeros = 100100000000
Remainder: 000000000001 Data sent: 100100000001
Result: 000000000000

LIMITATION:

One limitation my program has is that a user can enter any characters that aren't 1 or 0. Another limitation is, I have a fixed divisor, it'd be easier to corrupt the data entered. And also CRC is not suitable for protecting against intentional alteration of data.

I believe that my program is program quite well for it will calculate any data size even its just one. Even if a user input 0, the result will still be zero, hence no error detected in the data.