CSE 350 Project Outline

* Intended for Spring 2020 Section 2 and 3

1. You'll have to implement and simulate data transmission using **hamming code.**
2. The usable programming languages are C++, Java, and python. You may implement your project using any one of these.
3. Do the implementation according to the code skeleton provided. **If you understand the skeleton you don't need to read the following instructions.**

4. Your implementation must contain the following:
    a. Two classes: one for the sender, the other for the receiver
    b. In the main program, there will be a sender object and a receiver object
    c. Take the message to be sent by the sender (will be a binary string) and type of parity (odd/even) to be used as inputs. Pass it to the sender.
    d. Sender object will generate the codeword (a binary string) using the inputs. While generating the codeword, the sender will do the following correctly.
        i. Compute the number of parity bits
        ii. Positioning the parity bits
        iii. Calculating the value of the parity bits
    e. Save the codeword generated by Sender in a variable *correctCW*
    f. Change one bit of the variable randomly. Save it in a new variable. *singleErrorCW*
    g. Change two bits of the variable randomly. Save it in another new variable *doubleErrorCW*
    h. Pass the size of the codeword and parity type to the receiver.
    i. Pass the *correctErrorCW* to the receiver object. The receiver must understand that the bit string is without any error and print "No error"
    j. Pass the *singleErrorCW* to the same receiver object. The receiver must detect the error and correct it and print "There is an error. If single-bit it is corrected"
    k. Pass the *doubleErrorCW* to the same receiver object. The receiver must detect the error and attempt to correct it and print "There is an error. If single-bit it is corrected. However, in this scenario, the corrected message will also be erroneous.
    l. You must use bitwise operators while generating the codeword
    m. If someone is found to have done the exact codes from any one of the previous semesters, she/he will get **zero** in the project.

```
        }

/***** This is a skeleton of how your code should be. You are free to do your code in any
language among C++, Java, and python. Just make sure that your code follows this skeleton
*******/

class Sender {
        string message;
        int codewordsize;
        char parityType;

public:

        int getCodeWordSize(){
                //calculate the codewordsize
        }
        void setMessage(string message){
        }
        void setParityType(int parityType){
        }
        String generateCodeword(){
                //generate the codeword from the message and parityType
        }
}

class Receiver {
        int codewordsize;
        char parityType;

public:
        void setCodeWordSize(){
        }
        void setParityType(){
        }
        void receiveCodeword(string codeword){
              /*** Check whether the codeword contains an error
                  If it does, correct it irrespective of the number of bits in error
                  However, more than one bit gets altered, the corrected stream will also be
erroneous. ***/
        }
}
```

```
string randomChange(string codeword, int num)
{
        // randomly alter 'num' number of bits in the codeword and return it
}

int main()
{
        Sender sender;
        Receiver receiver;

        string message;
        char parityType; // 'o' for odd parity, 'e' for even parity

        // Take the message and parityType as input

        sender.setMessage(message);
        sender.setParityType(parityType);


        string codeword = sender.generateCodeword();

        correctCW = codeword;
        singleErrorCW = randomChange(codeword, 1);
        doubleErrorCW = randomChange(codeword, 2);


        int codewordsize = sender.getCodeWordSize();
        receiver.setCodeWordSize(codewordsize);
        receiver.setParityType(parityType);

        receiver.receiveCodeword(correctCW);
        receiver.receiveCodeword(singleErrorCW);
        receiver.receiveCodeword(doubleErrorCW);

}
```

Sample Input	Sample Output
Message: 10110110 Parity Type: e (for even)	Codeword: 101110111000 Single Bit Error: 001110111000 Double Bit Error: 100110101000 For No Change in codeword: 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