Assignment 4

Name: Hitesh Tolani

Roll no: 73

Class: SY-AIDS-A

Title: Write a program for error detection and correction for 7/8 bits ASCII codes using CRC.

Theory:

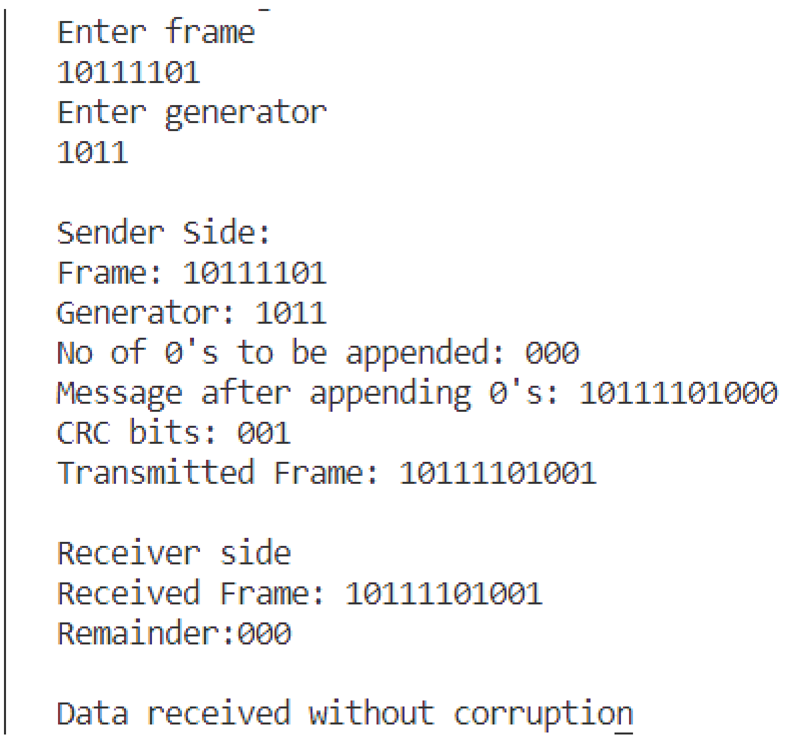
Cyclic Redundancy Check (CRC) is a widely used error-checking technique employed in digital communication and data storage systems. It is a type of hash function that detects errors in transmitted or stored data by generating a fixed-size checksum or hash value. CRC operates by dividing the data into blocks and calculating a remainder, which is then appended to the original data. During transmission or storage, this checksum is recalculated at the receiving end, and any discrepancy in the checksum indicates the presence of errors

The strength of CRC lies in its ability to detect common types of errors, such as single-bit errors and burst errors, making it a robust and efficient method for ensuring data integrity. CRC is particularly favoured in networking protocols, such as Ethernet and Wi-Fi, where data integrity is crucial. It offers a balance between simplicity and effectiveness, as the algorithm is straightforward to implement in hardware or software.

Code:

```python
def xor(divisor:str,dividend:str):
    tmp = ""
    for idx in range(0,len(divisor)):
        if(divisor[idx] != dividend[idx]):
            tmp += "1"
        else:
            tmp += "0"

    return tmp

def mod2div(frame: str, generator: str) -> str:

    GEN_LEN = len(generator)
    FRAME_LEN = len(frame)

    tmp = frame[0:GEN_LEN]
    nextBitIdx = GEN_LEN

    while nextBitIdx < FRAME_LEN :

        if len(tmp) == GEN_LEN:
            tmp = xor(generator,tmp) + frame[nextBitIdx]
        else:
            tmp += frame[nextBitIdx]

        tmp = tmp.lstrip('0')
        nextBitIdx+=1

    if len(tmp) == GEN_LEN:
        tmp = xor(generator,tmp)

    return tmp[1:]


def senderSide(frame: str, generator: str) -> str:

    print("\nSender Side:")

    GEN_LEN = len(generator)
    endBits = "0" * (GEN_LEN - 1)

    appendedFrame = ''+frame

    appendedFrame += endBits

    result = mod2div(appendedFrame, generator)
    transmittedFrame = frame + result

    print(
        f"Frame: {frame}\nGenerator: {generator}\nNo of 0's to be appended:
{endBits}"
    )
    print(f"Message after appending 0's: {appendedFrame}\nCRC bits:
{result}\nTransmitted Frame: {transmittedFrame}")

    return transmittedFrame


# CRC checker -> receiver
def receiverSide(frame: str, generator:str) -> None:
    result = mod2div(frame, generator)

    print(f"\nReceiver side \nReceived Frame: {frame}\nRemainder:{result}\n ")

    if "1" not in result:
        print("Data received without corruption")
    else:
        print("Data has been corrupted")


if __name__ == "__main__":
    frame = input("Enter frame\n")
    generator = input("Enter generator\n")

    data = senderSide(frame, generator)
    receiverSide(data, generator)
```

Output:



```
Enter frame
10111101
Enter generator
1011

Sender Side:
Frame: 10111101
Generator: 1011
No of 0's to be appended: 000
Message after appending 0's: 10111101000
CRC bits: 001
Transmitted Frame: 10111101001

Receiver side
Received Frame: 10111101001
Remainder:000

Data received without corruption
```