Group A – Assignment 4

Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes or CRC.

Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer-to-peer mode

Outcomes:

Demonstrate Hamming Codes and CRC with example.

Theory:

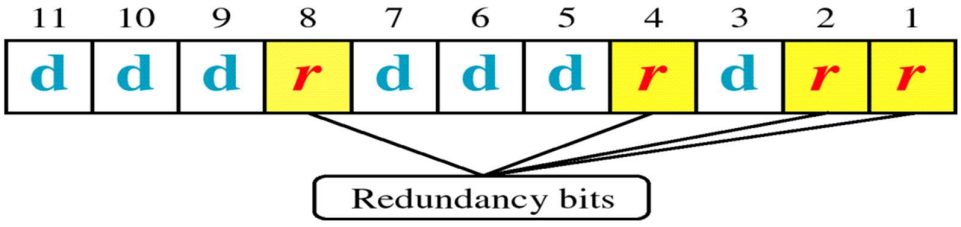
Hamming code

- Hamming codes are a family of linear error-correcting codes that generalize the Hamming (7,4)-code
- Invented by Richard Hamming in 1950

Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors.

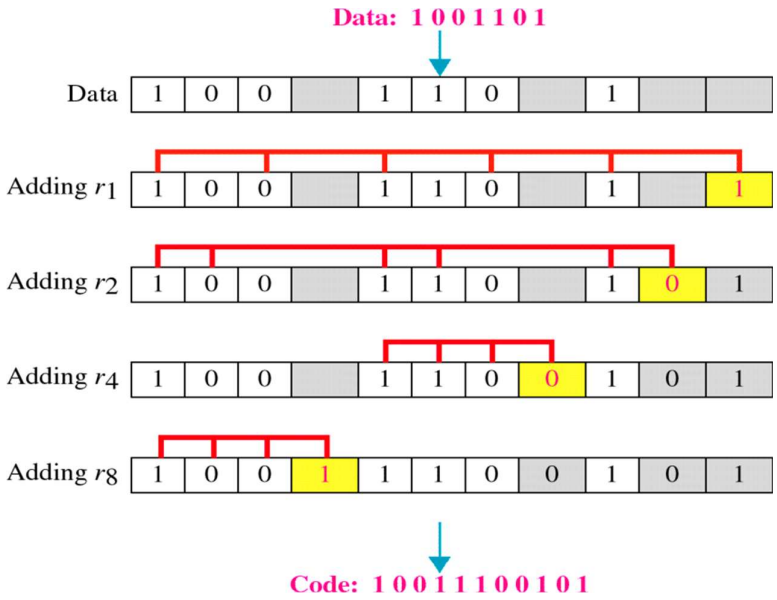
General algorithm

- The following general algorithm generates a single-error correcting (SEC) code for any number of bits.
- Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.
- Write the bit numbers in binary: 1, 10, 11, 100, 101, etc.
- All bit positions that are powers of two (have only one 1 bit in the binary form of their position) are parity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000)
- All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.
- Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.
- Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.
- Parity bit 1 covers all bit positions which have the least significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.
- Parity bit 2 covers all bit positions which have the second least significant bit set: bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.
- Parity bit 4 covers all bit positions which have the third least significant bit set: bits 4–7, 12–15, 20–23, etc.
- Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 8–15, 24–31, 40–47, etc.
- In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero**.**

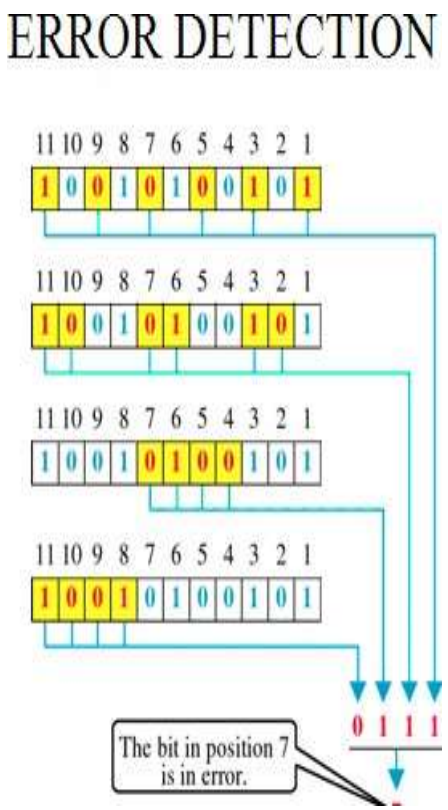


Example

Error detection



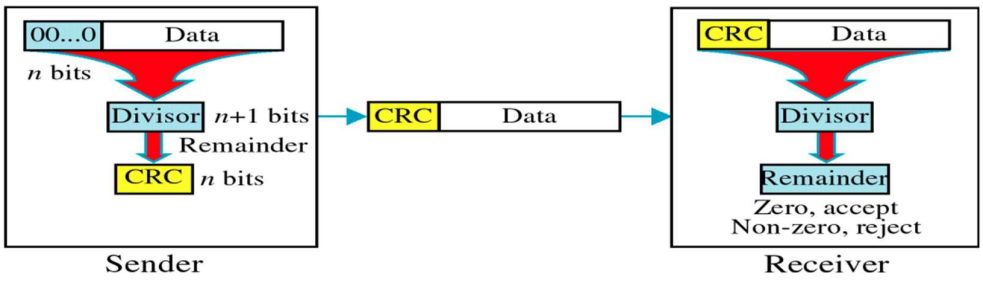
Error correction



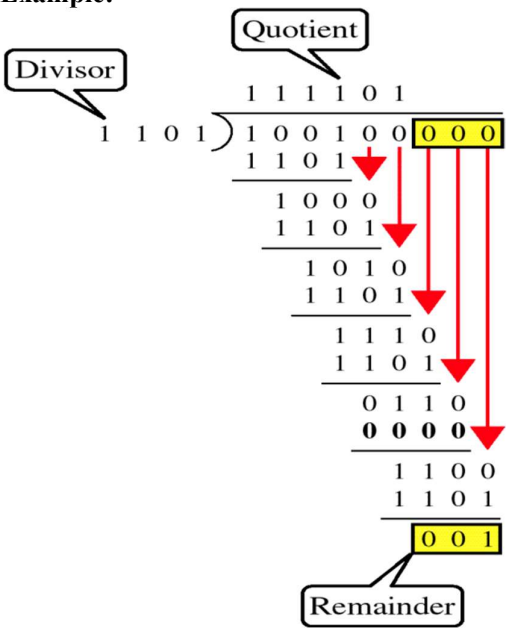
Cyclic Redundancy Check: CRC

- Given a k-bit frame or message, the transmitter generates an n-bit sequence, known as a *frame check sequence (FCS),* so that the resulting frame, consisting of (k+n) bits, is exactly divisible by some predetermined number.

- The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error.



Example:



Conclusion:
Hence, we studied and implement program for error detection and correction for 7/8 bits ASCII codes using CRC.

Frequently Asked Question

Q1. What is Parity check?

Q2. What is Hamming code?

Q3. How many redundancy bits are required for 16 bits of data?

Q4. List the Advantages of Hamming Code?

Q5. What is CRC ?

Humming Code program: -

```
#include<iostream>
using namespace std;
void reciever_Side()
{
	int n;
	int ar[7];
	cout<<"Enter a 7 digit codeword that is recieved: ";
	cin>>n;
	int x=n;
	int i=0;
	while(x!=0)
	{
		ar[i] = x%10;
		x=x/10;
		i++;
	}
	cout<<"The code entered is: ";
	for(int j=0; j<7; j++)
	{
		cout<<ar[j];
	}

	cout<<endl<<endl;

	int p1, p2, p4;
	p1 =  ar[2] ^ ar[4] ^ ar[6];
	p2 =  ar[2] ^ ar[5] ^ ar[6];
	p4 =  ar[4] ^ ar[5] ^ ar[6];

//	cout<<" p1: "<<p1<<endl;
```

```
//          cout<<" p2: "<<p2<<endl;
//          cout<<" p4: "<<p4<<endl;

            if(p1== ar[0] and p2==ar[1] and p4 ==ar[3])
            {
             cout<<" The given word has no errors"<<endl;
            }
            else
            {
                        cout<<" The given code has errors!"<<endl;
                        int z;
                        z= ar[0] + (ar[1]*2) + (ar[3]*4);
                        cout<<" The error is at "<<z<<" position."<<endl;

            }

}

int main()
{
 block1:
            int x;
            cout<<" Press 1 to send codeword \n Press 2 to recieve the code word and check for errors\n
Enter here: ";
            cin>>x;

            if(x==1)
            {
             int arr[7];
            cout<<"Enter data bits"<<endl;
            cout<<"First byte: ";
            cin>>arr[4];
            cout<<"Second byte: ";
            cin>>arr[2];
            cout<<"Third byte: ";
            cin>>arr[1];
            cout<<"Fourth byte: ";
            cin>>arr[0];

            arr[6] = arr[4] ^ arr[2] ^ arr[0];
            arr[5] = arr[4] ^ arr[1] ^ arr[0];
            arr[3] = arr[2] ^ arr[1] ^ arr[0];
            cout<<"The codeword transmitted is: ";
            for(int i =0; i<7; i++)
              {
                        cout<<arr[i];
              }

              cout<<endl<<endl;
              goto block1;
            }
            else if(x==2)
            {
                        reciever_Side();
            }
```

```
else{
        cout<<" Wrong option entered!";
        cout<<endl<<endl;
        goto block1;
}


return 0;
}
```

Output

```
SENDER SIDE:
Enter 7 data bits (space-separated, e.g., 1 0 1 1 0 0 1): 1 1 1 0 0 0 0
Generated 11-bit Hamming Code (to send): 0 0 1 0 1 1 0 0 0 0 0

RECEIVER SIDE:
Enter the 11-bit Hamming code received (space-separated): 1 1 1 0 1 1 0 1 1
    1 1

Error detected at position: 3
Corrected Code: 1 1 0 0 1 1 0 1 1 1 1


=== Code Execution Successful ===
```