Assignment 2 Write Up:

Assignment - 02.

* Title: Hamming code.
* Problem statement: write a C/C++ program for error detection and correction for 7/8 bits ASCII codes using Hamming code or CRC. Demonstrate the packets captured traces using wireshark packet analyzer tool for peer to peer mode.
* S/W & H/W Requirements:
  Windows 10 (64 bit), C/C++ compiler, Wireshark Packet Analyzer tool
* Theory:
  - When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems
  - The corrupted bits lead to spurious data being received by the receiver & are called errors

ECCs

- Error Correcting Codes are sequence of numbers generated by specific algorithms for detecting and removing errors in data that has been transmitted over noisy channels

Teacher's Sign:

Page No.
Date

Types of errors:

i) Single bit data error:
The change in one bit in the whole data sequence is called 'serial communication system'.

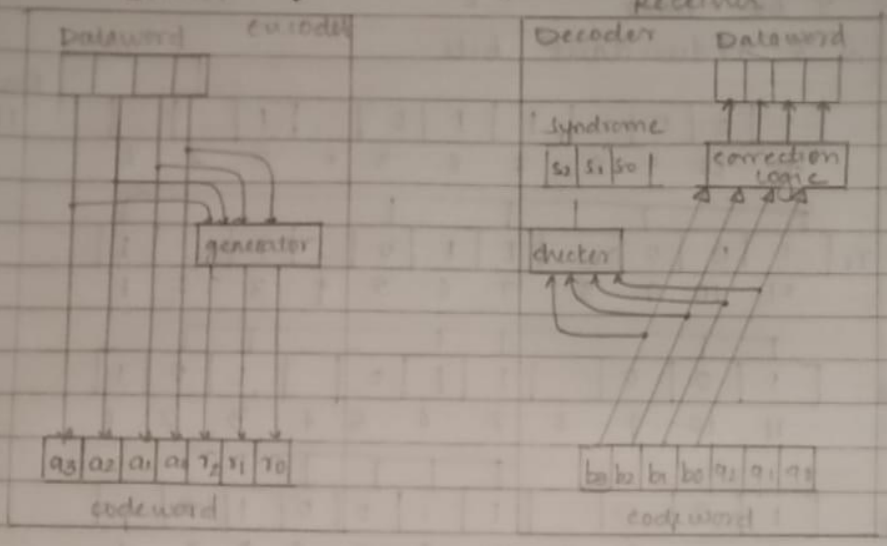
ii) Multiple bit errors:
If there is chance in two or more bits of data sequence

iii) Burst error:
The change of set of bits in data sequence of transmission to receiver is called multiple bit error.

- Hamming code:
→ This error detecting and correcting codes technique is developed by R.W Hamming
- This code not only identifies the error bit in the whole data sequence and it also corrects it
- This code uses number of parity bits located at certain positions in the codeword
- The number of parity bits depend upon the number of information bits
- The Hamming code uses the relation between redundancy bits & the data bits and this code can be applied to any number of bits.

Teacher's Sign.:

* Implementation of Hamming code:



* Algorithm:

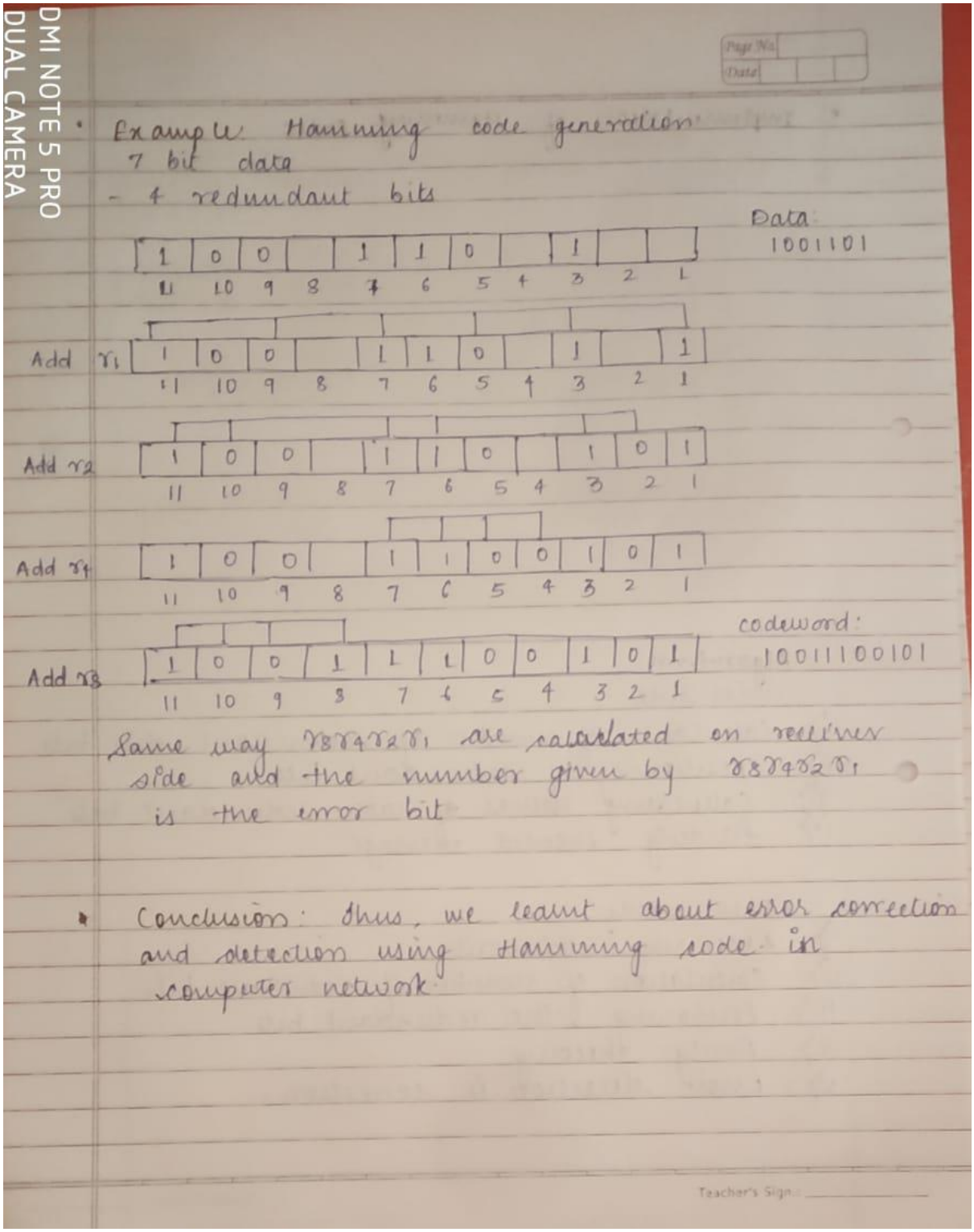
Sender side:

i) Calculation of number of redundant disk bits
ii) Positioning of redundant bits.
iii) Calculating values of each redundant bits
iv) Sending encoded message

Receiver side:

i) Receiving codeword.
ii) Calculation of number of redundant bits.
iii) Positioning the redundant bits
iv) Parity checking
v) Error detection & correction.

Teacher's Sign.: _____



Hamming_Code Program:

```
#include<iostream>

#include<math.h>

#include<time.h>
```

```
using namespace std;
class Hamming_code
{
	int msg[30];
	int code_sent[30];
	int code_received[30];
	char parity;
	int data_bits,parity_bits;
	public:
		Hamming_code()
		{
			for(int i=0;i<30;i++)
			{
				msg[i]=code_sent[i]=code_received[i]=0;
			}
			data_bits=parity_bits=0;
			parity='E';
		}
		void cal_pbits()
		{
			cout<<"Enter type of Parity(E/O):";
			cin>>parity;
			cout<<"Enter number of data bits:";
			cin>>data_bits;
			while(data_bits+parity_bits+1>pow(2,parity_bits))
			{
				parity_bits++;
			}
			cout<<"No of parity bits: "<<parity_bits<<endl;
			cout<<"Total number of data bits:"<<parity_bits+data_bits<<endl;
		}
```

```
void read_message()
{
	cout<<"Enter Message:";
	for(int i=1;i<=data_bits;i++)
		cin>>msg[i];
	cout<<"Message entered is:";
	for(int i=1;i<=data_bits;i++)
		cout<<msg[i];
	cout<<endl;
}
void encode_msg()
{
	int d=0,p=1;
	for(int i=1;i<=data_bits+parity_bits;i++)
	{
		if(i==pow(2,d))
		{
			code_sent[i]=0;
			d++;
		}
		else
		{
			code_sent[i]=msg[p];
			p++;
		}
	}
	p=0;
	int min,max=0,bit_sum,k,j;
	for(int i=1;i<=data_bits+parity_bits;i=pow(2,p))
	{
		p++;
```

```
                bit_sum=0;
                k=i;
                min=1;
                max=i;
                for(j=i;j<=data_bits+parity_bits;j=k+i)
                {
                        for(k=j;max>=min&&k<=data_bits+parity_bits;++min,++k)
                        {
                                if(code_sent[k] == 1)
                                bit_sum++;
                        }
                        min=1;
                }
                if(parity=='E')
                {
                        if(bit_sum%2==0)
                                code_sent[i]=0;
                        else
                                code_sent[i]=1;
                }
                else
                {
                        if(bit_sum%2!=0)
                                code_sent[i]=0;
                        else
                                code_sent[i]=1;
                }
        }
}
void sent_print()
{
```

```
	cout<<"Code Sent with";
	if(parity=='E')
		cout<<" Even Parity:";
	else
		cout<<" Odd Parity:";
	for(int i=1;i<=data_bits+parity_bits;i++)
		{
			cout<<code_sent[i];
		}
	cout<<endl;
}
void get_received()
{
	for(int i=1;i<=data_bits+parity_bits;i++)
		code_received[i]=code_sent[i];
}
void disturbance()
{
	srand(time(0));
	int i = rand()%(data_bits+parity_bits)+1;
	if(code_received[i]==1)
		code_received[i]=0;
	else
		code_received[i]=1;
}
void print_received()
{
	cout<<"Code received with:";
	if(parity=='E')
		cout<<" Even Parity:";
	else
```

```
                cout<<" Odd Parity:";
        for(int i=1;i<=data_bits+parity_bits;i++)
                cout<<code_received[i];
}
bool equal()
{
        for(int i=1;i<=data_bits+parity_bits;i++)
                if(code_sent[i]!=code_received[i])
                        return false;
        return true;
}
void error_checking()
{
        int p=0;
        int min,max=0,bit_sum,j,k;
        int code[10]={0};
        int q=1;
        for(int i=1;i<=data_bits+parity_bits;i=pow(2,p))
        {
                p++;
                bit_sum=0;
                j=i;
                k=i;
                min=1;
                max=i;
                for(j;j<=data_bits+parity_bits;)
                {
                        for(k=j;max>=min&&k<=data_bits+parity_bits;++min,++k)
                        {
                                if(code_received[k]==1)
                                        bit_sum++;
```

```
                        }
                        j=k+i;
                        min=1;
                }
                if(parity=='E')
                {
                        if(bit_sum%2==0)
                                code[q]=0;
                        else
                                code[q]=1;
                }
                else
                {
                        if(bit_sum%2!=0)
                                code[q]=0;
                        else
                                code[q]=1;
                }
                q++;
        }
        int error=0;
        int p1=0;
        for(int l=1,p1=0;l<q;l++,p1++)
        {
                error+=code[l]*pow(2,p1);
        }
        cout<<"Error is in bit no.:"<<error<<endl;
    }
};
int main()
{
```

```
int choice;
Hamming_code obj;
do
{
	cout<<"1.Sender Side\n2.Receiver Side\n3.Exit\n";
	cin>>choice;
	switch(choice)
	{
		case 1:
			cout<<"1.Without Disturbance\n2.With Disturbance\n";
			cin>>choice;
			obj.cal_pbits();
			obj.read_message();
			obj.encode_msg();
			obj.get_received();
			switch(choice)
			{
				case 1:
					obj.sent_print();
					break;
				case 2:
					obj.disturbance();
					obj.sent_print();
					break;
			}
			break;
		case 2:
			obj.print_received();
			if(obj.equal())
				cout<<"\nNo Error in received code."<<endl;
			else
```

```
                                        obj.error_checking();
                                    break;
                        }
            }while(choice!=3);
}
```

Ouput:

```
1.Sender Side
2.Receiver Side
3.Exit
1
1.Without Disturbance
2.With Disturbance
1
Enter type of Parity(E/O):O
Enter number of data bits:4
No of parity bits: 3
Total number of data bits:7
Enter Message:0 1 0 1
Message entered is:0101
Code Sent with Odd Parity:1001101
1.Sender Side
2.Receiver Side
3.Exit
2
Code received with: Odd Parity:1001101
No Error in received code.
1.Sender Side
2.Receiver Side
3.Exit
1
1.Without Disturbance
```

2.With Disturbance

2

Enter type of Parity(E/O):E

Enter number of data bits:7

No of parity bits: 4

Total number of data bits:11

Enter Message:1 1 0 1 0 0 1

Message entered is:1101001

Code Sent with Even Parity:01101011001

1.Sender Side

2.Receiver Side

3.Exit

2

Code received with: Even Parity:01101010001

Error is in bit no.:8

1.Sender Side

2.Receiver Side

3.Exit