Name: Raghav Maheshwari
Roll NO: 53
Panel: A

Lab Assignment - 3 (CN)

ERROR DETECTION & CORRECTION

AIM: To write a program for error detection and correction using Hamming code.

OBJECTIVES:
1. To encode and decode original original data bits with help of parity bits.
2. To demonstrate use of error control protocols.

THEORY:
(1) Types of errors:
Errors can be classified into two categories, they are

i) Single - Bit Error:
In this only one bit of a given data unit is changed. It does not appear more likely in serial data transmission but mainly in parallel Data Transmission.

ii) Burst Error: In this two or more bits are changed and determined from first corrupted bit to last corrupted bit. The duration of noise in Burst Error is more than duration of noise in single bit as it depends on duration of noise and data rate.

② Concept of parity bits:
A parity bit is a check bit, which is added to a block of data for error detection purpose. It is used to validate integrity of data. Value of parity bit is assigned either 0 or 1 that makes it suitable for single bit error detection only.

③ Hamming code:
Hamming code is used to detect and correct error in transmitting data. So, it is an error detection and correction code. In this transmission, message is encoded with redundant bits. Redundant bits are extra bits that are placed at certain location of data bits to detect error.
There are 3 steps to encode data bits with hamming code:

i) Selecting number of redundant bits
→ $2^r \geq m+r+1$ (r = no. of redundant bits, m = no. of data bits)

ii) Choosing location of redundant bits
→ redundant bits are placed at position that are numbered corresponding to power of 2 i.e. 1, 2, 4, 8

iii) Assigning value of redundant bits.
→ values assign to redundant bits are XOR of those significant data bit position.
Eg- r_1 checks 1, 3, 5 - - - r_2 check 2, 3, 6 - - etc.

We have seen how hamming code technique works for error detection & correction in a data packet transmitted over a network.

* FAQ

Q1 What is the difference b/w flow and error control?

Ans

Flow Control	Error Control
→ Flow control is meant only for transmission of data from sender to receiver.	→ Error control is meant for transmission of error from sender to receiver
→ It prevents loss of data and avoid over running of receive buffers.	→ It is used to detect and correct error occured in code.
→ for flow control there are two approaches: feedback - based flow control and rate based flow control.	→ To detect error in data, approaches are: Parity checking, Checksum.

Q2 Explain in brief two types of error control mechanisms.

Ans i) Forward error control mechanisms:
In forward error control, additional reductant information is also transmitted along with data.

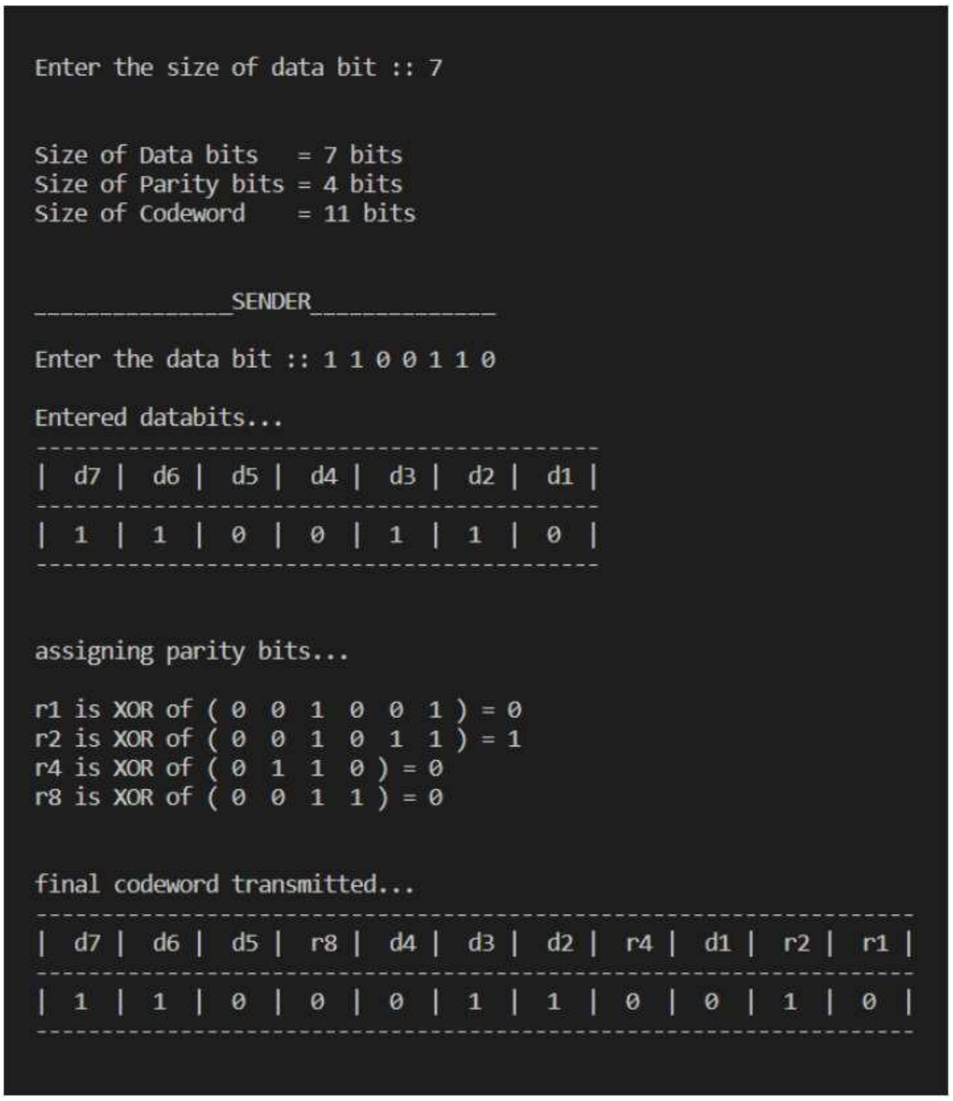
ii) Backward or feedback error control mechanism:
In this, along with each character, a little additional information is provided for detection of errors; the receiver in this technique performs no error correction.

CODE:

```
#include <iostream>
#include <math.h>
using namespace std;
void display (int n,int r,int arr[]);
void display (int m,int arr[]);
void assign_Rbit(int n,int code_word[]);
void gen_codeword(int m,int n,int data_bit[],int code_word[]);
void error_check(int n,int a[],int b[]);
int main()
{
 int m=0, n=0, r=0;
 printf("\n\nEnter the size of data bit :: ");
 cin>>m;
 while(pow(2,r)<=m+r+1){
 r++;
 }
 n=m+r;
 printf("\n\nSize of Data bits = %d bits\nSize of Parity bits = %d bits\nSize
of
 Codeword = %d bits\n\n",m,r,n);
 printf("\n________________SENDER_______________\n\nEnter the data
bit ::
");
 int code_word[n] , data_bit[m] ;
 gen_codeword(m,n,data_bit,code_word);
 cout<<"\nEntered databits...";
 display(m,data_bit);
 cout<<"\nassigning parity bits...\n";
 assign_Rbit(n,code_word);
 cout<<"\nfinal codeword transmitted...";
 display(n,r,code_word);
 cout<<"\n________________RECEIVER_______________\n\nEnter desired
data bits :: ";
 int receiver_word[n] , receiver_data[m] ;
 gen_codeword(m,n,receiver_data,receiver_word);
 cout<<"\nEntered databits...";
 display(m,receiver_data);
 cout<<"\nassigning parity bits...\n";
 assign_Rbit(n,receiver_word);
 cout<<"\nReceiver expected codeword...";
 display(n,r,receiver_word);
 cout<<"\nchecking all of the parity bits...";
 error_check(n,code_word,receiver_word);
 return 0;
 }
void display (int n,int r,int arr[]){
 cout<<endl;
 int count=r-1,count2=n-r,count3=r-1;
 cout<<"-";
 for (int i = 0; i < n; i++){
 cout<<"------";
 }
 cout<<endl<<"|";
 for(int i =n-1;i>=0;i--){
 if (i==pow(2,count)-1){
 count--;
 cout<<" r"<<pow(2,count3)<<" |";
 count3--;
 }
 else{
 cout<<" d"<<count2<<" |";
 count2--;
 }
 }
 cout<<endl<<"-";
 for (int i = 0; i < n; i++){
 cout<<"------";
 }
 cout<<endl<<"|";
 for (int i = 0; i < n; i++){
 cout<<" "<<arr[n-1-i]<<" |";
 }
 cout<<endl<<"-";
 for (int i = 0; i < n; i++){
 cout<<"------";
 }
 cout<<endl<<endl;
}
void display (int m,int arr[]){
 cout<<endl<<"-";
 for (int i = 0; i < m; i++){
 cout<<"------";
 }
 cout<<endl<<"|";
 for(int i =m;i>0;i--){
 cout<<" d"<<i<<" |";
 }
 cout<<endl<<"-";
 for (int i = 0; i < m; i++){
 cout<<"------";
 }
 cout<<endl<<"|";
 for (int i = 0; i < m; i++){
 cout<<" "<<arr[i]<<" |";
 }
 cout<<endl<<"-";
 for (int i = 0; i < m; i++){
 cout<<"------";
 }
 cout<<endl<<endl;
}
void assign_Rbit(int n,int code_word[]){
 int c=0,it=0,count;
 for(int j=0;j<n;it++){
 count=pow(2,it);
 c=0;
 cout<<endl;
 cout<<"r"<<count<<" is XOR of (";
 for(int k=j;k<n;k++){
 if(count==0){
 k+=pow(2,it)-1;
 count=pow(2,it);
 continue;
 }
 count--;
 cout<<" "<<code_word[k]<<" ";
 c=c^code_word[k];
 code_word[(int)(pow(2,it)-1)]=c;
 }
 cout<<") = "<<c;
 j=pow(2,it+1)-1;
 }
 cout<<endl<<endl;
}
void gen_codeword(int m,int n,int data_bit[],int code_word[]){
 int count = 0 , count2 = 0;
 for (int i = 0;i<m;i++){
 cin>>data_bit[i];
 }
 for(int i = 0; i<n; i++){
```

```
 code_word[i]=0;
 if (i==pow(2,count)-1){
 code_word[i]=0;
 count++;
 continue;
 }
 code_word[i]=data_bit[m-1-count2];
 count2++;
 }
}
void error_check(int n,int a[],int b[]){
 int count=-1 ,flag =0,err_bit=0,r_bit=0;
 cout<<endl;
 for (int i = 0; i <n;i+=pow(2,count))
 {
 if(a[i]==b[i]){
 cout<<"\nr"<<pow(2,count+1)<<" matched...";
 count++;
 }
 else{
 cout<<"\nr"<<pow(2,count+1)<<" not matched...";
 count++;
 r_bit=count+1;
 err_bit+=pow(2,count);
 flag = 1;
 }
 }
 cout<<"\n\n\n";
 if (flag == 1){
 cout<<"transmitted codeword and expected codeword does NOT
match...\n\nerror detected ::
 'd"<<err_bit-r_bit<<"' data bit is flipped...";
 }
 else{
 cout<<"transmitted codeword and expected codeword
match...\n\ncodeword received
 successfully...";
 }
 cout<<"\n\n\n\n"
}
```

OUTPUT:



```
Enter the size of data bit :: 7


Size of Data bits   = 7 bits
Size of Parity bits = 4 bits
Size of Codeword    = 11 bits


_______________SENDER______________

Enter the data bit :: 1 1 0 0 1 1 0

Entered databits...
-------------------------------------------
|  d7 |  d6 |  d5 |  d4 |  d3 |  d2 |  d1 |
-------------------------------------------
|  1  |  1  |  0  |  0  |  1  |  1  |  0  |
-------------------------------------------


assigning parity bits...

r1 is XOR of ( 0  0  1  0  0  1 ) = 0
r2 is XOR of ( 0  0  1  0  1  1 ) = 1
r4 is XOR of ( 0  1  1  0 ) = 0
r8 is XOR of ( 0  0  1  1 ) = 0


final codeword transmitted...
-------------------------------------------------------------------
|  d7 |  d6 |  d5 |  r8 |  d4 |  d3 |  d2 |  r4 |  d1 |  r2 |  r1 |
-------------------------------------------------------------------
|  1  |  1  |  0  |  0  |  0  |  1  |  1  |  0  |  0  |  1  |  0  |
-------------------------------------------------------------------
```