EXPERIMENT NO. 5

SEMESTER: V **DATE OF PERFORMANCE: 10/08/2022**

SUBJECT: CN Lab **DATE OF SUBMISSION: 16/08/2022**

NAME OF THE STUDENT: Ashish Jha **ROLL NO.: 27**

AIM	Write a program to simulate Hamming code generation, detection and correction.
LEARNING OBJECTIVE	The student will demonstrate the working of (7,4) hamming code.
LEARNING OUTCOME	The student will be able to detect and correct errors using hamming code.
COURSE OUTCOME	CSL502.3: Simulate and explore networking algorithms and protocols.
PROGRAM OUTCOME	PO1,PO2,PO3,PO4,PO5,PO9,PO10,PSO1,PSO2,PSO3
BLOOM'S TAXONOMY LEVEL	Apply
THEORY	**Parity bits:** The bit which is appended to the original data of binary bits so that the total number of 1s is even or odd. **Even parity:** To check for even parity, if the total number of 1s is even, then the value of the parity bit is 0. If the total number of 1s occurrences is odd, then the value of the parity bit is 1. **Odd Parity:** To check for odd parity, if the total number of 1s is even, then the value of parity bit is 1. If the total number of 1s is odd, then the value of parity bit is 0. **Algorithm of Hamming code:** • An information of 'd' bits are added to the redundant bits 'r' to form d+r. • The location of each of the (d+r) digits is assigned a decimal value. • The 'r' bits are placed in the positions 1,2,.....2^{k-1}. • At the receiving end, the parity bits are recalculated. The decimal value of the parity bits determines the position of an error.
LAB EXERCISE	**Write program in C/C++/Java/Python (Write separate programs for odd**

	and even parity) 1. Ask the user whether the program will work for even parity (or for odd) parity. 2. The user can enter the 4-bit data . 3. Complete Code Word for this and can be generated by calculating for the parity bits: P1=(D3,D5,D7) P2=(D3,D6,D7) P4=(D5,D6,D7) 4.Enter the Received codeword (with or without error).Notify the user to introduce error at only one bit position. 5.Check bits 1,3,5,7.....to generate C1.Check bits 2,3,6,7.....to generate C2.Check bits 4,5,6,7.....to generate C3.Generate the error codeword =C3C2C1. 6.If error is there,it will be reflected at which position and will display the corrected code word by inverting the respective bit. 7.Decode the data bits. /* Sample Output This is hamming code error detection and correction using EVEN parity Enter 4 data bits.D7 D6 D5 D3 Enter the value of D7:1 Enter the value of D6:0 Enter the value of D5:1 Enter the value of D3:0 3 parity bits are required for the transmission of data bits. SENDER: The data bits entered are: 1 0 1 0

```
The Parity bits are:

Value of P4 is 0

Value of P2 is 1

Value of P1 is 0

The Hamming code is as follows :-

D7 D6 D5 P4 D3 P2 P1

1 0 1 0 0 1 0

Enter the hamming code with error at any position of your choice.

NOTE: ENTER A SPACE AFTER EVERY BIT POSITION.

Error should be present only at one bit position

1 0 1 0 1 1 0

RECEIVER:

Error is detected at position 3 at the receiving end.

Correcting the error....

The correct code is 1 0 1 0 0 1 0

The decoded data is:1010

*/

#include <iostream>

#include <cmath>

#include <vector>

#include <map>

using namespace std;
```

```
vector<int> data_stream(7, 0);
int parity;
int count = 0;
int error = 0;
vector<int> check_sum(3, 0);
string label[] = {"D7", "D6", "D5", "P4", "D3", "P2", "P1"};
string parityarr[] = {"Even", "Odd"};
int pos = 0;
int res = 0;


void print()
{


   cout << "\n";
   for (int i = 0; i < 7; i++)
   {


      cout << label[i] << " ";
   }
   cout << "\n";
   for (int i = data_stream.size() - 1; i >= 0; i--)
   {
```

```
        cout << data_stream[i] << "  ";
    }
    cout << "\n";
}

int parity_transmission(int a, int b, int c)
{
    int count = 0;

    count = data_stream[a] + data_stream[b] + data_stream[c];

    if (count % 2 == parity)
    {

        return 0;
    }
    else
    {

        return 1;
    }
}
```

```
int parity_receiving(int ind, int a, int b, int c)

{



   int count = 0;

   count = data_stream[ind] + data_stream[a] + data_stream[b] +
data_stream[c];



   if (count % 2 == parity)

   {



      return 0;

   }

   else

   {



      return 1;

   }

}



int main()

{



x:
```

```
cout << "Choose parity \n0.Even\n1.Odd  \n";


cin >> parity;


if (parity == 0)
{
   cout << "Even parity choosen \n";
}
else if (parity == 1)
{


   cout << "Odd parity Choosen \n";
}


else
{


   cout << "Invalid  choice\n";
   goto x;
}


cout << "Hamming code error detection using " << parityarr[parity] << "
```

```
parity \n";


   cout << "Enter 4 data bits  D7  D6  D5  D3\n";


   for (int i = 6; i >= 0; i--)
   {
      int x;


      if (i == 0 || i == 1 || i == 3)


      {
         continue;
      }
      cout << "Enter the value of D" << i + 1 << " ";
      cin >> x;


      data_stream[i] = x;
   }


   data_stream[0] = parity_transmission(2, 4, 6);
   data_stream[1] = parity_transmission(2, 5, 6);
   data_stream[3] = parity_transmission(4, 5, 6);
```

```
// print();
cout << "3 parity bits are required for the transmission of data bits.\n";

cout << "SENDER: \n";

cout << "The data bits entered are: ";

for (int i = 6; i >= 0; i--)
{

   if (i == 0 || i == 1 || i == 3)

   {
      continue;
   }

   cout << data_stream[i] << " ";
}

cout << "\n";
cout << "The parity bits are : ";

cout << "\n";
```

```
for (int i = 0; i < 7; i++)
{


   if (i == 3 || i == 5 || i == 6)
   {


      cout << label[i] << " ";
   }
}
cout << "\n";
for (int i = 6; i >= 0; i--)
{


   if (i == 0 || i == 1 || i == 3)


   {
      cout << data_stream[i] << "  ";
   }
}
cout << "\n";
cout << "The Hamming code is as follows :- \n";
cout << "\n";
print();
```

```
   cout << "Enter the hamming code with error at any position of your choice \n";

   cout << "\n";

   cout << "NOTE: ENTER 2 SPACES AFFTER EVERY BIT POSITION \n";

   cout << "\n";

   cout << "Error should be present at only one bit position \n";



   for (int i = 0; i < 7; i++)

   {

      cout << label[i] << " ";

   }

   cout << "\n";

   for (int i = 6; i >= 0; i--)

   {



      int x;

      cin >> x;



      data_stream[i] = x;

   }

   cout << "RECEIVER: \n";
```

```
cout << "Received data stream is: \n";


print();


cout << "\n";


check_sum[0] = parity_receiving(0, 2, 4, 6);

check_sum[1] = parity_receiving(1, 2, 5, 6);

check_sum[2] = parity_receiving(3, 4, 5, 6);


cout << "\n";


for (int i = 2; i >= 0; i--)

{


   res = check_sum[i] * pow(2, i);


   pos += res;

}

cout << "Error is detected at position " << pos << " at the receiving end.
\n";

cout << "Correcting the error..... \n";
```

```
int errorpos = pos - 1;
if (data_stream[errorpos] == 1)
{

   data_stream[errorpos] = 0;
}

else
{
   data_stream[errorpos] = 1;
}

cout << "\n";

cout << "The correct code is : \n";
print();

cout << "The decoded data is : ";

for (int i = 6; i >= 0; i--)
{

   if (i == 0 || i == 1 || i == 3)
```

```
    {
        continue;
    }

    cout << data_stream[i] << " ";
  }

  return 0;
}
```

```
PROBLEMS 5  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE                                                    Code

Choose parity
0.Even
1.Odd
cd "f:\MOODLE\Sem_5\Comp netw\codes\" ; if ($?) { g++ hamming_code2.cpp -o hamming_code2 } ; if ($?) { .\hamming_code2 }
Even parity choosen
Hamming code error detection using Even parity
Enter 4 data bits  D7  D6  D5  D3
Enter the value of D7 Enter the value of D6 Enter the value of D5 Enter the value of D3 3 parity bits are required for the transmission of data bits.
SENDER:
32758  32758  32758  32758  32758  32758  32758


Error is detected at position 0 at the receiving end.
Correcting the error.....

The correct code is :

D7 D6 D5 P4 D3 P2 P1
32758  32758  32758  32758  32758  32758  32758
The decoded data is : 32758 32758 32758 32758
PS F:\MOODLE\Sem_5\Comp netw\codes>
```

REFERENCES

- B.A. Forouzan, “Data Communications and Networking”, TMH, Fourth Edition.
- https://www.javatpoint.com/computer-network-error-correction