Practical-3

Name: Abhijeet Vidwan Vyavhare
Roll No: 232
PRN: 202202040012

Problem statement:

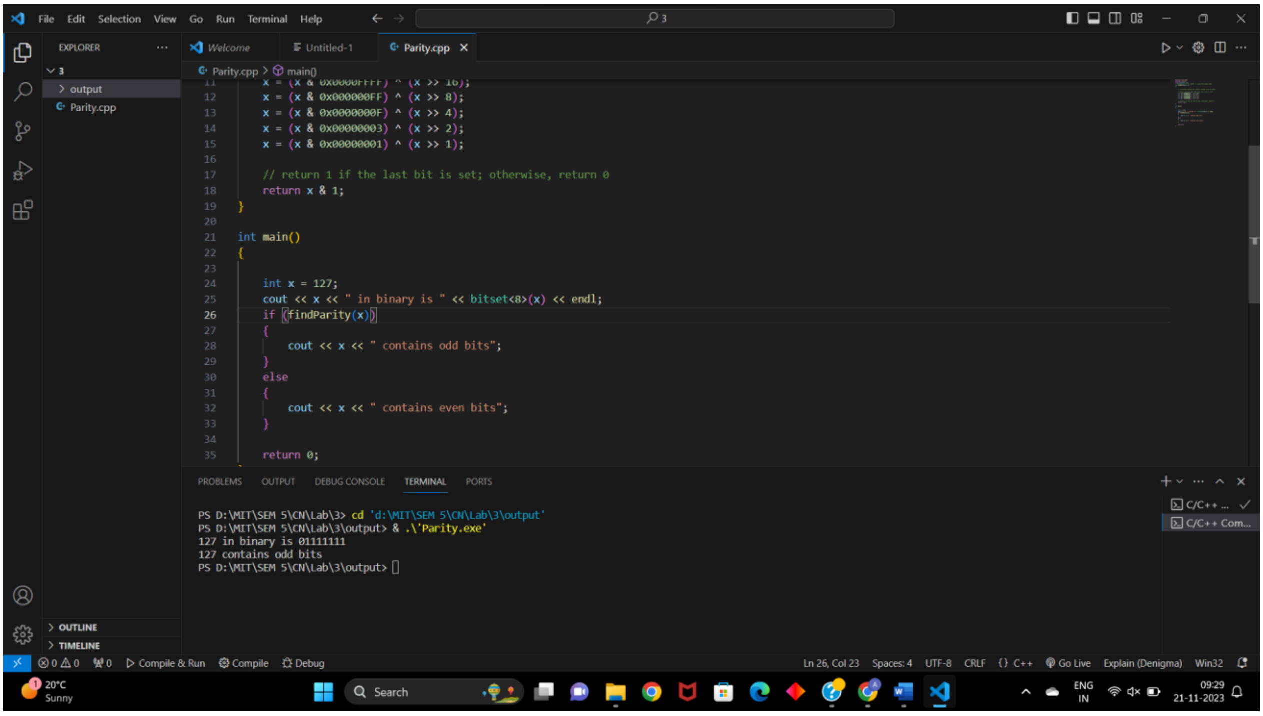
Write a program to detect and correct single-bit error using

1. Parity Check 2. Hamming Code and 3. Cyclic Redundancy Check

1) Parity check:

```
#include <iostream>
#include <bitset>
using namespace std;
// Compute parity of a number `x` using the lookup table
int findParity(int x)
{

   // recursively divide the (32–bit) integer into two equal

   // halves and take their XOR until only 1 bit is left
   x = (x & 0x0000FFFF) ^ (x >> 16);
   x = (x & 0x000000FF) ^ (x >> 8);
   x = (x & 0x0000000F) ^ (x >> 4);
   x = (x & 0x00000003) ^ (x >> 2);
   x = (x & 0x00000001) ^ (x >> 1);

   // return 1 if the last bit is set; otherwise, return 0
   return x & 1;
}

int main()
{

   int x = 127;
   cout << x << " in binary is " << bitset<8>(x) << endl;
   if (findParity(x))
   {
      cout << x << " contains odd bits";
   }
   else
   {
      cout << x << " contains even bits";
   }

   return 0;
}
```

OUTPUT:


File Edit Selection View Go Run Terminal Help
Welcome
Untitled-1
Parity.cpp
EXPLORER
output
Parity.cpp
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Compile & Run
Compile
Debug
Ln 26, Col 23 Spaces: 4 UTF-8 CRLF C++ Go Live Explain (Denigma) Win32
20°C Sunny
Search
ENG IN
09:29 21-11-2023

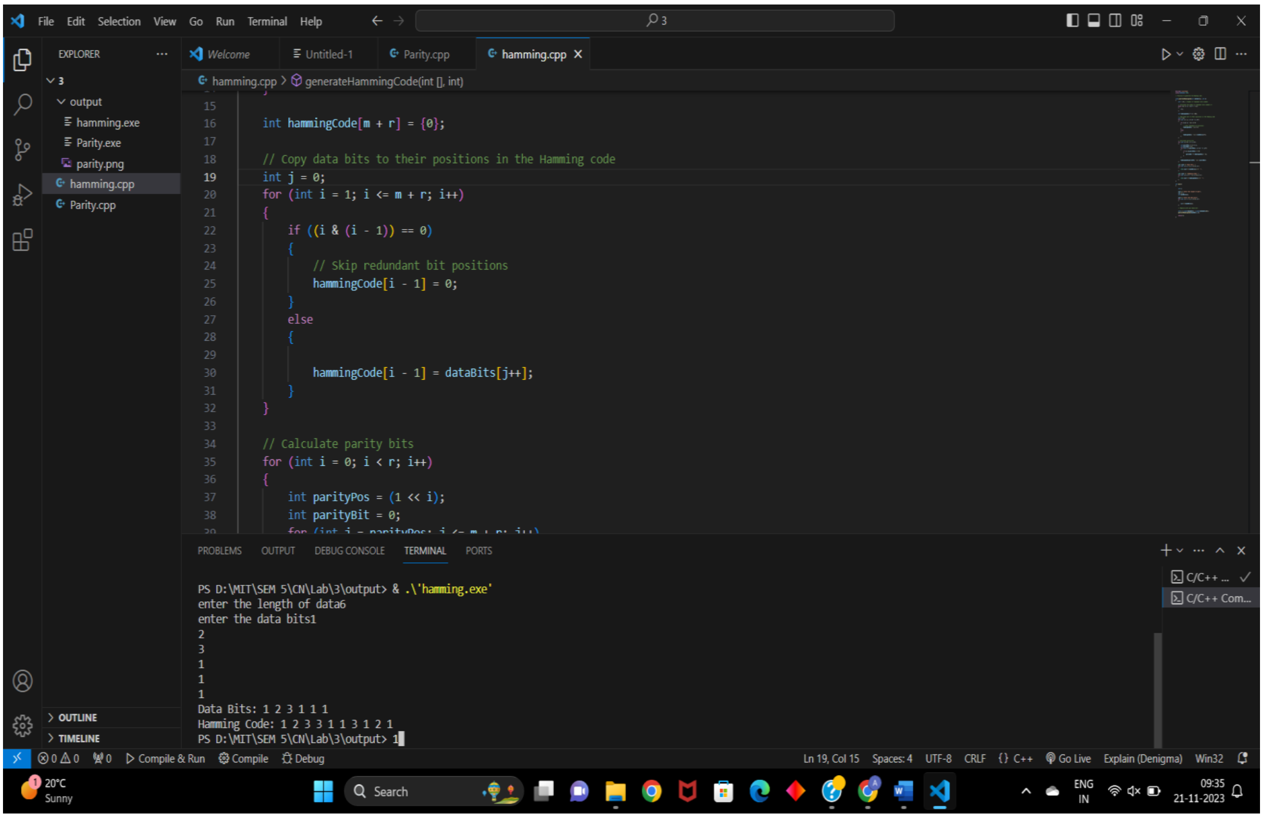
```
    x = (x & 0x0000FFFF) ^ (x >> 16);
    x = (x & 0x000000FF) ^ (x >> 8);
    x = (x & 0x0000000F) ^ (x >> 4);
    x = (x & 0x00000003) ^ (x >> 2);
    x = (x & 0x00000001) ^ (x >> 1);

    // return 1 if the last bit is set; otherwise, return 0
    return x & 1;
}

int main()
{

    int x = 127;
    cout << x << " in binary is " << bitset<8>(x) << endl;
    if (findParity(x))
    {
        cout << x << " contains odd bits";
    }
    else
    {
        cout << x << " contains even bits";
    }

    return 0;
```

```
PS D:\MIT\SEM 5\CN\Lab\3> cd 'd:\MIT\SEM 5\CN\Lab\3\output'
PS D:\MIT\SEM 5\CN\Lab\3\output> & .\'Parity.exe'
127 in binary is 01111111
127 contains odd bits
PS D:\MIT\SEM 5\CN\Lab\3\output>
```

2) Hamming code:

```
#include <iostream>
using namespace std;

// Function to generate the Hamming code
void generateHammingCode(int dataBits[], int m)
{
    int r = 0; // Number of redundant bits needed

    // Calculate the number of redundant bits needed (r)
    while ((1 << r) < (m + r + 1))
    {
        r++;
    }
    int hammingCode[m + r] = {0};

    // Copy data bits to their positions in the Hamming code
    int j = 0;
    for (int i = 1; i <= m + r; i++)
    {
        if ((i & (i - 1)) == 0)
        {
            // Skip redundant bit positions
            hammingCode[i - 1] = 0;
        }
        else
        {

            hammingCode[i - 1] = dataBits[j++];
        }
    }

    // Calculate parity bits
    for (int i = 0; i < r; i++)
    {
        int parityPos = (1 << i);
        int parityBit = 0;
        for (int j = parityPos; j <= m + r; j++)
        {
            if ((j & parityPos) != 0)
            {
                parityBit ^= hammingCode[j - 1];
            }
        }

        hammingCode[parityPos - 1] = parityBit;
    }

    std::cout << "Data Bits: ";
    for (int i = m - 1; i >= 0; i--)
    {
        std::cout << dataBits[i] << " ";
    }

    std::cout << "\nHamming Code: ";
    for (int i = m + r - 1; i >= 0; i--)
    {
        std::cout << hammingCode[i] << " ";
    }
}

int main()
{

    int n;

    cout << "enter the length of data";
    cin >> n;
    int dataBits[n];

    cout << "enter the data bits";
    for (int i = n - 1; i >= 0; i--)
    {
        cin >> dataBits[i];
    }

    // Replace with your data bits

    int m = sizeof(dataBits) / sizeof(dataBits[0]);
    generateHammingCode(dataBits, m);

    return 0;
}
```

OUTPUT:


File Edit Selection View Go Run Terminal Help
EXPLORER
3
output
hamming.exe
Parity.exe
parity.png
hamming.cpp
Parity.cpp
Welcome
Untitled-1
Parity.cpp
hamming.cpp
hamming.cpp > generateHammingCode(int [], int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
C/C++ ...
C/C++ Com...
OUTLINE
TIMELINE
Ln 19, Col 15 Spaces: 4 UTF-8 CRLF {} C++ Go Live Explain (Denigma) Win32
Compile & Run Compile Debug
20°C Sunny
Search
ENG IN
09:35 21-11-2023

```cpp
    int hammingCode[m + r] = {0};

    // Copy data bits to their positions in the Hamming code
    int j = 0;
    for (int i = 1; i <= m + r; i++)
    {
        if ((i & (i - 1)) == 0)
        {
            // Skip redundant bit positions
            hammingCode[i - 1] = 0;
        }
        else
        {

            hammingCode[i - 1] = dataBits[j++];
        }
    }

    // Calculate parity bits
    for (int i = 0; i < r; i++)
    {
        int parityPos = (1 << i);
        int parityBit = 0;
```

```
PS D:\MIT\SEM 5\CN\Lab\3\output> & .\'hamming.exe'
enter the length of data6
enter the data bits1
2
3
1
1
1
Data Bits: 1 2 3 1 1 1
Hamming Code: 1 2 3 3 1 1 3 1 2 1
PS D:\MIT\SEM 5\CN\Lab\3\output> 1
```

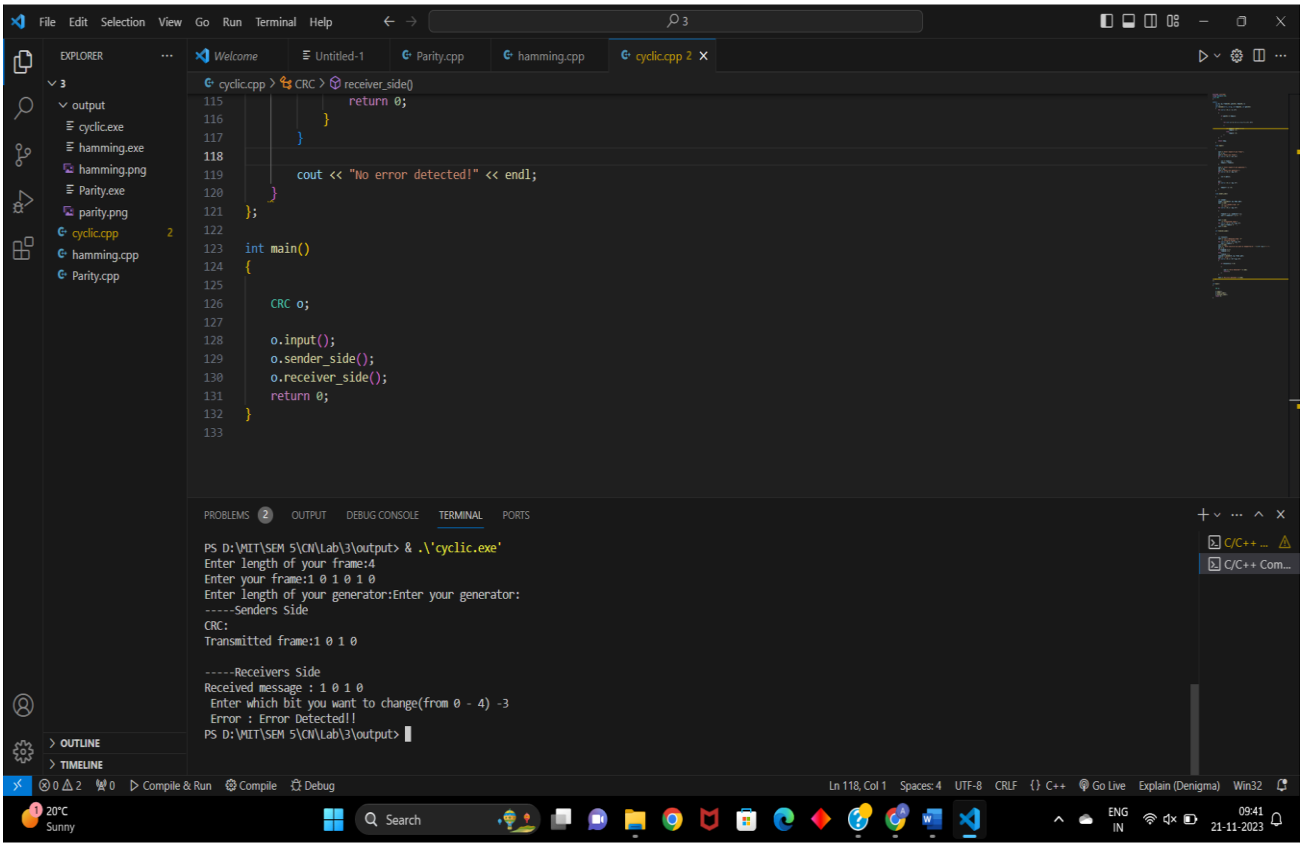
3) Cyclic Redundancy Check

```
#include <iostream>
using namespace std;
class CRC
{

public:
   int nf, ng, frame[20], gen[10], temp[20], b;
   char a;
   int *divide(int n, int g, int temp[10], int gen[10])
   {
      for (int i = 0; i < n; i++)

      {

         if (gen[0] == temp[i])

         {

            for (int j = 0, k = i; j < g + 1; j++, k++)

            {

               if (temp[k] ^ gen[j] == 1)
                  temp[k] = 1;
```

```
                else
                    temp[k] = 0;
            }
        }
    }

    return temp;
}

void input()

{

    cout << "Enter length of your frame:";
    cin >> nf;
    cout << "Enter your frame:";
    for (int i = 0; i < nf; i++)
    {

        cin >> frame[i];
        temp[i] = frame[i];
    }
    cout << "Enter length of your generator:";
    cin >> ng;
    cout << "Enter your generator:";
    for (int i = 0; i < ng; i++)
    {

        cin >> gen[i];
    }

    ng--;
    for (int i = 0; i < ng; i++)
    {

        temp[nf + i] = 0;
    }
}

void sender_side()

{

    int *sender;
    sender = divide(nf, ng, temp, gen);
    cout << endl
         << "-----Senders Side  \n"
         << "CRC:";
    for (int i = 0; i < ng; i++)

    {

        frame[nf + i] = sender[nf + i];
```

```
            cout << sender[nf + i] << ' ';
        }

        cout << endl
             << "Transmitted frame:";
        for (int i = 0; i < nf + ng; i++)
            cout << frame[i] << ' ';
        cout << endl;
    }

    int receiver_side()

    {

        int *receiver;
        cout << "\n-----Receivers Side  \n"
             << "Received message : ";
        for (int i = 0; i < nf + ng; i++)
            cout << frame[i] << ' ';
        cout << endl;
        cout << " Enter which bit you want to change(from 0 - " << nf + ng << ") -";
        cin >> b;
        if (frame[b] == 1)
            frame[b] = 0;
        else
            frame[b] = 1;
        receiver = divide(nf, ng, frame, gen);
        cout << " Error : ";
        for (int i = 0; i < nf + ng; i++)
        {

            if (receiver[i] != 0)

            {

                cout << "Error Detected!!" << endl;
                return 0;
            }
        }

        cout << "No error detected!" << endl;
    }
};

int main()
{

    CRC o;
    o.input();
    o.sender_side();
    o.receiver_side();
    return 0;
}
```

OUTPUT:



```cpp
                return 0;
            }
        }

        cout << "No error detected!" << endl;
    }
};

int main()
{

    CRC o;

    o.input();
    o.sender_side();
    o.receiver_side();
    return 0;
}
```

```
PS D:\MIT\SEM 5\CN\Lab\3\output> & .\'cyclic.exe'
Enter length of your frame:4
Enter your frame:1 0 1 0 1 0
Enter length of your generator:Enter your generator:
-----Senders Side
CRC:
Transmitted frame:1 0 1 0

-----Receivers Side
Received message : 1 0 1 0
 Enter which bit you want to change(from 0 - 4) -3
 Error : Error Detected!!
PS D:\MIT\SEM 5\CN\Lab\3\output> 
```