EXPERIMENT NO. 4

SEMESTER: V	**DATE OF PERFORMANCE: 07th August 2024**
SUBJECT: CN Lab	**DATE OF SUBMISSION: 08th August 2024**
NAME OF THE STUDENT: Dwayne George Nixon	**ROLL NO.: 21**

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.Note-If error is introduced at more than one position then display "Hamming code is just 1-bit error detection and correction mechanism".
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
*/
```

Append the code here with all possible sample outputs.

Code for Even Parity:

```
#include <stdio.h>

// Function to calculate parity bits for even parity
void calculateParityBits(int code[]) {
    code[6] = code[0] ^ code[2] ^ code[4]; // P1 = D7 ^ D5 ^ D3
    code[5] = code[0] ^ code[1] ^ code[4]; // P2 = D7 ^ D6 ^ D3
    code[3] = code[0] ^ code[1] ^ code[2]; // P4 = D7 ^ D6 ^ D5
}

// Function to detect and correct any single-bit error for even parity
void detectAndCorrectError(int code[]) {
    int c1 = code[6] ^ code[4] ^ code[2] ^ code[0]; // C1 = P1 ^ D3 ^ D5 ^ D7
    int c2 = code[5] ^ code[4] ^ code[1] ^ code[0]; // C2 = P2 ^ D3 ^ D6 ^ D7
    int c4 = code[3] ^ code[2] ^ code[1] ^ code[0]; // C4 = P4 ^ D5 ^ D6 ^ D7

    int errorPosition = c4 * 4 + c2 * 2 + c1; // Determine the error position

    if (errorPosition != 0) {
        printf("Error is detected at position %d\n", errorPosition);
        code[7 - errorPosition] ^= 1; // Correct the error by flipping the bit
        printf("Correcting the error....\n");
    } else {
        printf("No error detected.\n");
    }
}

// Function to display the code
void displayCode(const int code[]) {
    for (int i = 0; i < 7; ++i) {
        printf("%d ", code[i]);
    }
    printf("\n");
}

int main() {
    printf("This is Hamming code error detection and correction using EVEN parity\n");

    int code[7];
    printf("Enter 4 data bits D7 D6 D5 D3:\n");
    printf("Enter the value of D7: ");
    scanf("%d", &code[0]);
    printf("Enter the value of D6: ");
    scanf("%d", &code[1]);
    printf("Enter the value of D5: ");
    scanf("%d", &code[2]);
    printf("Enter the value of D3: ");
    scanf("%d", &code[4]);

    printf("3 parity bits are required for the transmission of data bits.\n");

    calculateParityBits(code); // Calculate parity bits

    printf("SENDER:\n");
    printf("The data bits entered are: %d %d %d %d\n", code[0], code[1], code[2], code[4]);
    printf("The Parity bits are:\n");
    printf("Value of P4 is %d\n", code[3]);
    printf("Value of P2 is %d\n", code[5]);
    printf("Value of P1 is %d\n", code[6]);
    printf("The Hamming code is as follows: -\n");
    printf("D7 D6 D5 P4 D3 P2 P1\n");
    displayCode(code); // Display the Hamming code

    printf("Enter the hamming code with error at any position of your choice.\n");
    printf("NOTE: ENTER A SPACE AFTER EVERY BIT POSITION.\n");
    printf("Error should be present only at one bit position\n");

    for (int i = 0; i < 7; ++i) {
        scanf("%d", &code[i]);
    }

    printf("RECEIVER:\n");
    detectAndCorrectError(code); // Detect and correct any error
    printf("The correct code is ");
    displayCode(code); // Display the corrected code

    printf("The decoded data is: %d%d%d%d\n", code[0], code[1], code[2], code[4]);

    return 0;
}
```

Code for Odd Parity:

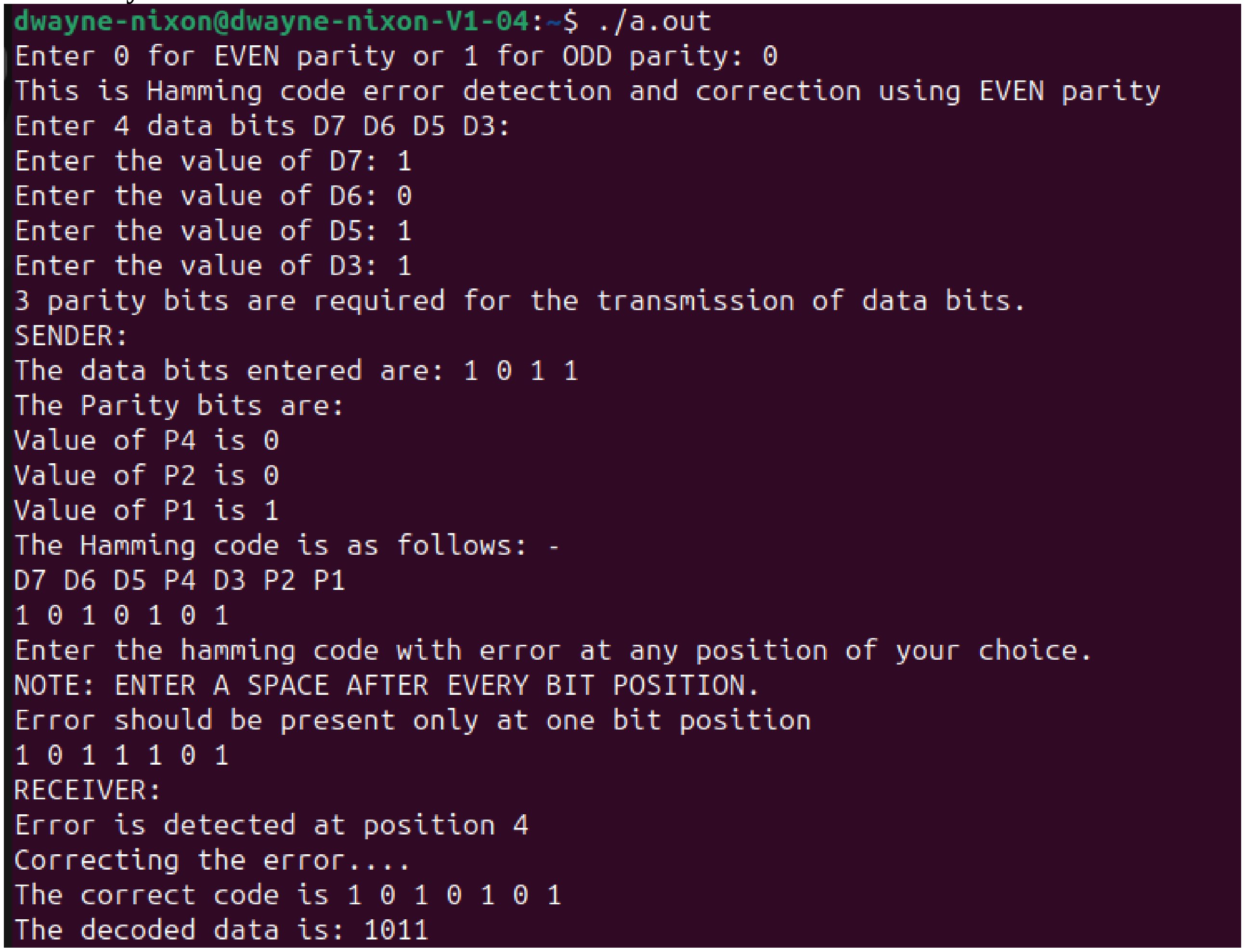
```c
#include <stdio.h>

// Function to calculate parity bits for odd parity
void calculateParityBits(int code[]) {
    code[6] = !(code[0] ^ code[2] ^ code[4]); // P1 = !(D7 ^ D5 ^ D3)
    code[5] = !(code[0] ^ code[1] ^ code[4]); // P2 = !(D7 ^ D6 ^ D3)
    code[3] = !(code[0] ^ code[1] ^ code[2]); // P4 = !(D7 ^ D6 ^ D5)
}

// Function to detect and correct any single-bit error for odd parity
void detectAndCorrectError(int code[]) {
    int c1 = !(code[6] ^ code[4] ^ code[2] ^ code[0]); // C1 = !(P1 ^ D3 ^ D5 ^ D7)
    int c2 = !(code[5] ^ code[4] ^ code[1] ^ code[0]); // C2 = !(P2 ^ D3 ^ D6 ^ D7)
    int c4 = !(code[3] ^ code[2] ^ code[1] ^ code[0]); // C4 = !(P4 ^ D5 ^ D6 ^ D7)

    int errorPosition = c4 * 4 + c2 * 2 + c1; // Determine the error position

    if (errorPosition != 0) {
        printf("Error is detected at position %d\n", errorPosition);
        code[7 - errorPosition] ^= 1; // Correct the error by flipping the bit
        printf("Correcting the error....\n");
    } else {
        printf("No error detected.\n");
    }
}

// Function to display the code
void displayCode(const int code[]) {
    for (int i = 0; i < 7; ++i) {
        printf("%d ", code[i]);
    }
    printf("\n");
}

int main() {
    printf("This is Hamming code error detection and correction using ODD parity\n");

    int code[7];
    printf("Enter 4 data bits D7 D6 D5 D3:\n");
    printf("Enter the value of D7: ");
    scanf("%d", &code[0]);
    printf("Enter the value of D6: ");
    scanf("%d", &code[1]);
    printf("Enter the value of D5: ");
    scanf("%d", &code[2]);
    printf("Enter the value of D3: ");
    scanf("%d", &code[4]);

    printf("3 parity bits are required for the transmission of data bits.\n");

    calculateParityBits(code); // Calculate parity bits

    printf("SENDER:\n");
    printf("The data bits entered are: %d %d %d %d\n", code[0], code[1], code[2], code[4]);
    printf("The Parity bits are:\n");
    printf("Value of P4 is %d\n", code[3]);
    printf("Value of P2 is %d\n", code[5]);
    printf("Value of P1 is %d\n", code[6]);
    printf("The Hamming code is as follows: -\n");
    printf("D7 D6 D5 P4 D3 P2 P1\n");
    displayCode(code); // Display the Hamming code

    printf("Enter the hamming code with error at any position of your choice.\n");
    printf("NOTE: ENTER A SPACE AFTER EVERY BIT POSITION.\n");
    printf("Error should be present only at one bit position\n");

    for (int i = 0; i < 7; ++i) {
        scanf("%d", &code[i]);
    }

    printf("RECEIVER:\n");
    detectAndCorrectError(code); // Detect and correct any error
    printf("The correct code is ");
    displayCode(code); // Display the corrected code

    printf("The decoded data is: %d%d%d%d\n", code[0], code[1], code[2], code[4]);

    return 0;
}
```

Combined Code with Screenshots:

```
#include <stdio.h>

// Function to calculate parity bits for even or odd parity
void calculateParityBits(int code[], int isOddParity) {
    if (isOddParity) {
        code[6] = !(code[0] ^ code[2] ^ code[4]); // P1 = !(D7 ^ D5 ^ D3)
        code[5] = !(code[0] ^ code[1] ^ code[4]); // P2 = !(D7 ^ D6 ^ D3)
        code[3] = !(code[0] ^ code[1] ^ code[2]); // P4 = !(D7 ^ D6 ^ D5)
    } else {
        code[6] = code[0] ^ code[2] ^ code[4]; // P1 = D7 ^ D5 ^ D3
        code[5] = code[0] ^ code[1] ^ code[4]; // P2 = D7 ^ D6 ^ D3
        code[3] = code[0] ^ code[1] ^ code[2]; // P4 = D7 ^ D6 ^ D5
    }
}

// Function to detect and correct any single-bit error for even or odd parity
void detectAndCorrectError(int code[], int isOddParity) {
    int c1, c2, c4;

    if (isOddParity) {
        c1 = !(code[6] ^ code[4] ^ code[2] ^ code[0]); // C1 = !(P1 ^ D3 ^ D5 ^ D7)
        c2 = !(code[5] ^ code[4] ^ code[1] ^ code[0]); // C2 = !(P2 ^ D3 ^ D6 ^ D7)
        c4 = !(code[3] ^ code[2] ^ code[1] ^ code[0]); // C4 = !(P4 ^ D5 ^ D6 ^ D7)
    } else {
        c1 = code[6] ^ code[4] ^ code[2] ^ code[0]; // C1 = P1 ^ D3 ^ D5 ^ D7
        c2 = code[5] ^ code[4] ^ code[1] ^ code[0]; // C2 = P2 ^ D3 ^ D6 ^ D7
        c4 = code[3] ^ code[2] ^ code[1] ^ code[0]; // C4 = P4 ^ D5 ^ D6 ^ D7
    }

    int errorPosition = c4 * 4 + c2 * 2 + c1; // Determine the error position

    if (errorPosition != 0) {
        printf("Error is detected at position %d\n", errorPosition);
        code[7 - errorPosition] ^= 1; // Correct the error by flipping the bit
        printf("Correcting the error....\n");
    } else {
        printf("No error detected.\n");
    }
}

// Function to display the code
void displayCode(const int code[]) {
    for (int i = 0; i < 7; ++i) {
        printf("%d ", code[i]);
    }
    printf("\n");
}

int main() {
    int isOddParity;
    printf("Enter 0 for EVEN parity or 1 for ODD parity: ");
    scanf("%d", &isOddParity);

    if (isOddParity) {
        printf("This is Hamming code error detection and correction using ODD parity\n");
    } else {
        printf("This is Hamming code error detection and correction using EVEN parity\n");
    }

    int code[7];
    printf("Enter 4 data bits D7 D6 D5 D3:\n");
    printf("Enter the value of D7: ");
    scanf("%d", &code[0]);
    printf("Enter the value of D6: ");
    scanf("%d", &code[1]);
    printf("Enter the value of D5: ");
    scanf("%d", &code[2]);
    printf("Enter the value of D3: ");
    scanf("%d", &code[4]);

    printf("3 parity bits are required for the transmission of data bits.\n");

    calculateParityBits(code, isOddParity); // Calculate parity bits

    printf("SENDER:\n");
    printf("The data bits entered are: %d %d %d %d\n", code[0], code[1], code[2], code[4]);
    printf("The Parity bits are:\n");
    printf("Value of P4 is %d\n", code[3]);
    printf("Value of P2 is %d\n", code[5]);
    printf("Value of P1 is %d\n", code[6]);
    printf("The Hamming code is as follows: -\n");
    printf("D7 D6 D5 P4 D3 P2 P1\n");
    displayCode(code); // Display the Hamming code

    printf("Enter the hamming code with error at any position of your choice.\n");
    printf("NOTE: ENTER A SPACE AFTER EVERY BIT POSITION.\n");
    printf("Error should be present only at one bit position\n");

    for (int i = 0; i < 7; ++i) {
        scanf("%d", &code[i]);
    }
    printf("RECEIVER:\n");
    detectAndCorrectError(code, isOddParity); // Detect and correct any error
    printf("The correct code is ");
    displayCode(code); // Display the corrected code

    printf("The decoded data is: %d%d%d%d\n", code[0], code[1], code[2], code[4]);

    return 0;
}
```

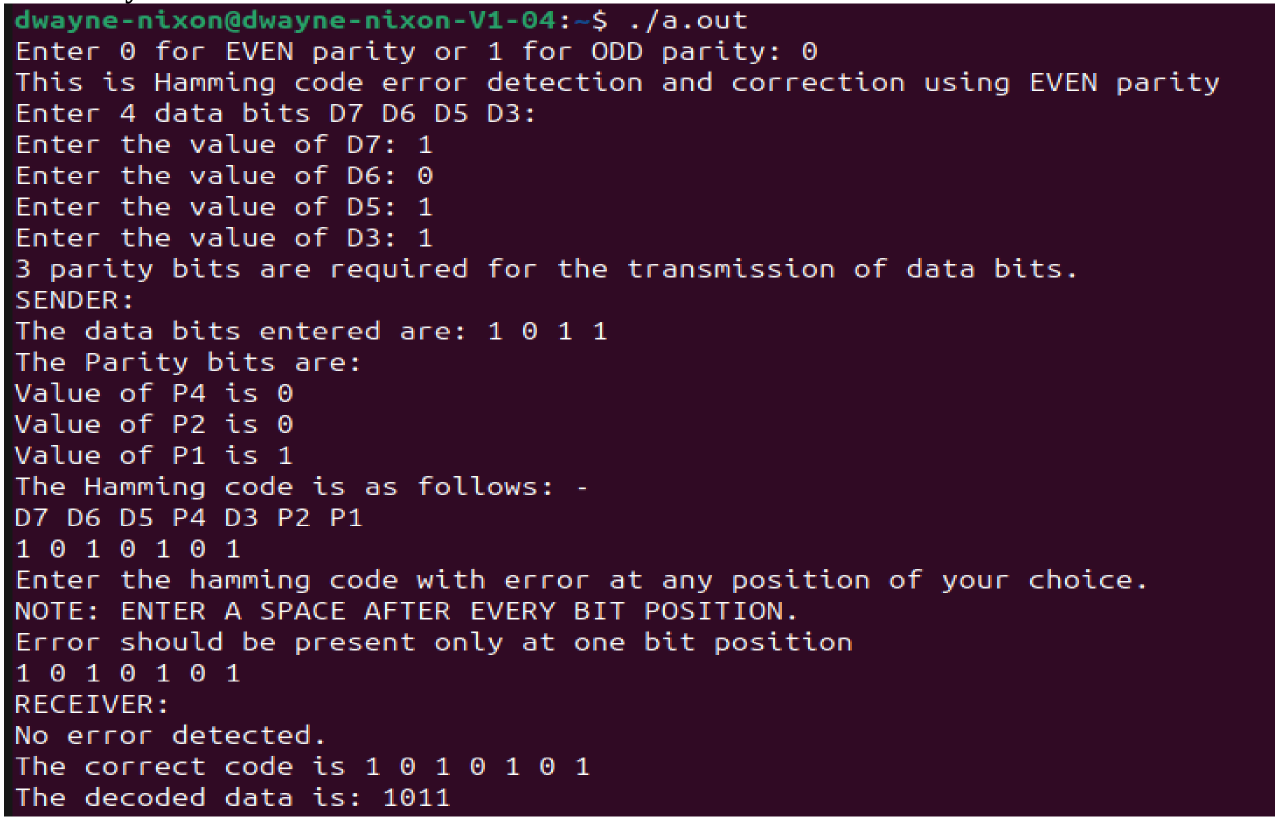
Screenshots:

Even Parity With Error:



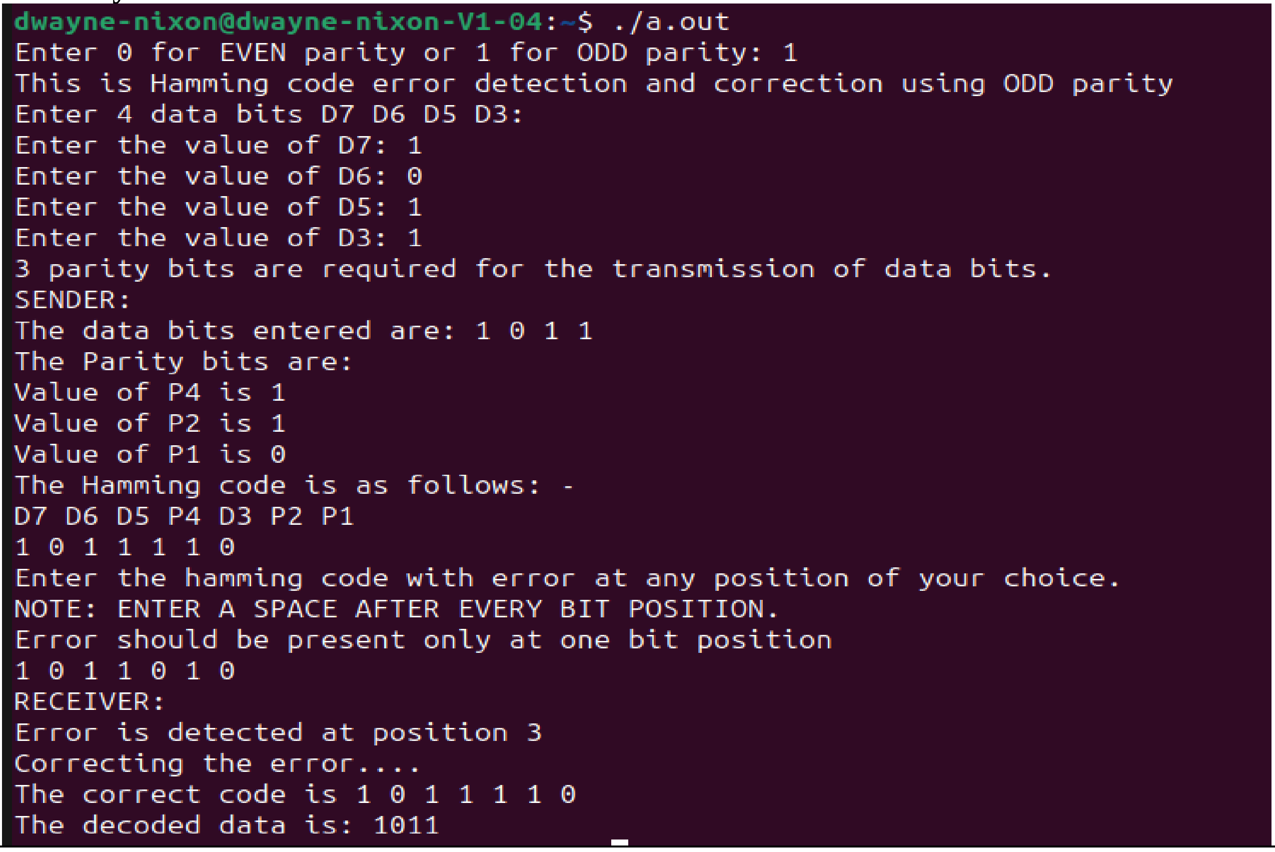
```
dwayne-nixon@dwayne-nixon-V1-04:~$ ./a.out
Enter 0 for EVEN parity or 1 for ODD parity: 0
This is Hamming code error detection and correction using EVEN parity
Enter 4 data bits D7 D6 D5 D3:
Enter the value of D7: 1
Enter the value of D6: 0
Enter the value of D5: 1
Enter the value of D3: 1
3 parity bits are required for the transmission of data bits.
SENDER:
The data bits entered are: 1 0 1 1
The Parity bits are:
Value of P4 is 0
Value of P2 is 0
Value of P1 is 1
The Hamming code is as follows: -
D7 D6 D5 P4 D3 P2 P1
1 0 1 0 1 0 1
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 1 1 0 1
RECEIVER:
Error is detected at position 4
Correcting the error....
The correct code is 1 0 1 0 1 0 1
The decoded data is: 1011
```

Even Parity With No Error:

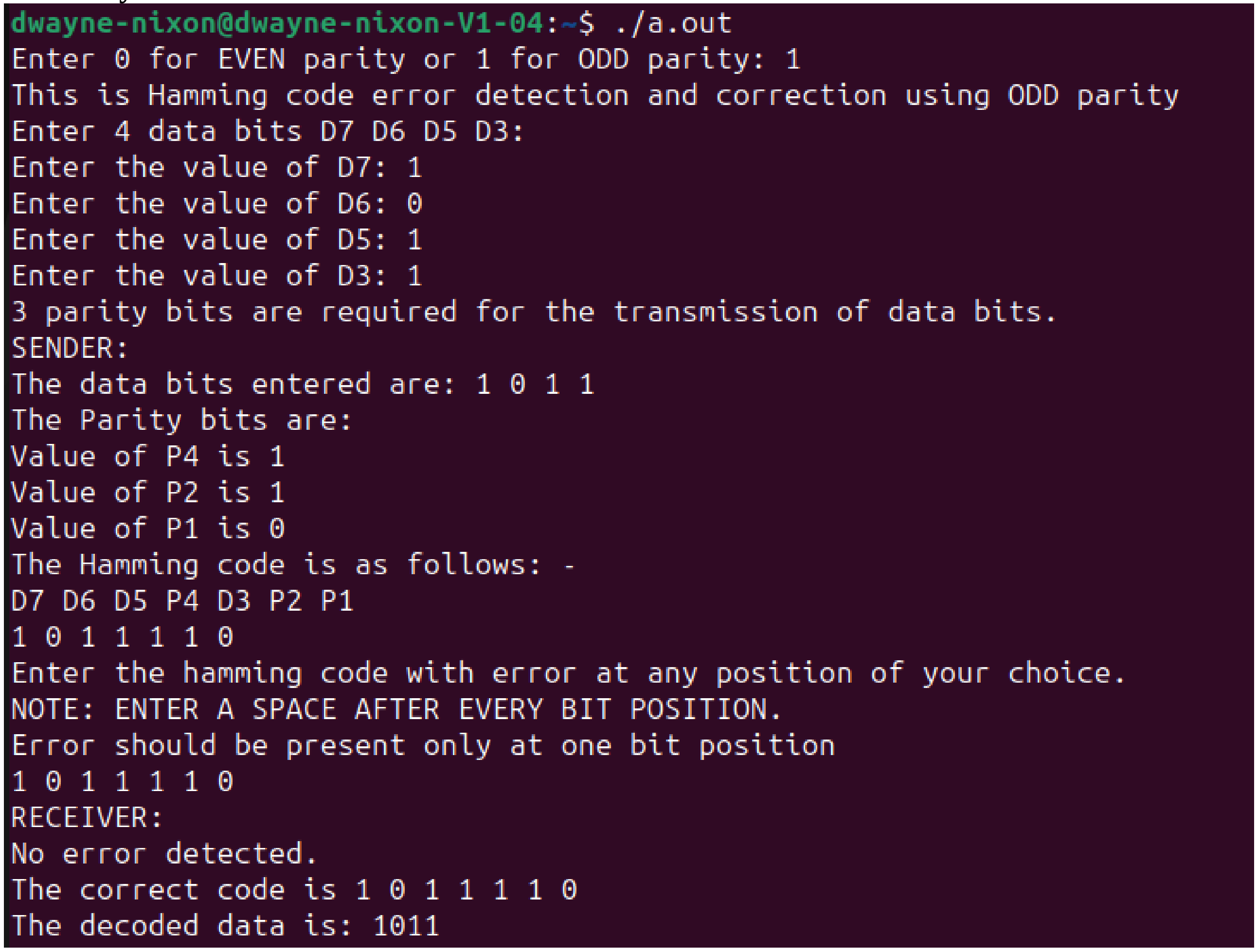


```
dwayne-nixon@dwayne-nixon-V1-04:~$ ./a.out
Enter 0 for EVEN parity or 1 for ODD parity: 0
This is Hamming code error detection and correction using EVEN parity
Enter 4 data bits D7 D6 D5 D3:
Enter the value of D7: 1
Enter the value of D6: 0
Enter the value of D5: 1
Enter the value of D3: 1
3 parity bits are required for the transmission of data bits.
SENDER:
The data bits entered are: 1 0 1 1
The Parity bits are:
Value of P4 is 0
Value of P2 is 0
Value of P1 is 1
The Hamming code is as follows: -
D7 D6 D5 P4 D3 P2 P1
1 0 1 0 1 0 1
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 0 1
RECEIVER:
No error detected.
The correct code is 1 0 1 0 1 0 1
The decoded data is: 1011
```

Odd Parity With Error:



```
dwayne-nixon@dwayne-nixon-V1-04:~$ ./a.out
Enter 0 for EVEN parity or 1 for ODD parity: 1
This is Hamming code error detection and correction using ODD 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: 1
3 parity bits are required for the transmission of data bits.
SENDER:
The data bits entered are: 1 0 1 1
The Parity bits are:
Value of P4 is 1
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 1 1 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 1 0 1 0
RECEIVER:
Error is detected at position 3
Correcting the error....
The correct code is 1 0 1 1 1 1 0
The decoded data is: 1011
```

	Odd Parity With No Error: 
REFERENCES	• B.A. Forouzan, "Data Communications and Networking", TMH,Fourth Edition. • https://www.javatpoint.com/computer-network-error-correction

```
dwayne-nixon@dwayne-nixon-V1-04:~$ ./a.out
Enter 0 for EVEN parity or 1 for ODD parity: 1
This is Hamming code error detection and correction using ODD 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: 1
3 parity bits are required for the transmission of data bits.
SENDER:
The data bits entered are: 1 0 1 1
The Parity bits are:
Value of P4 is 1
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 1 1 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 1 1 1 0
RECEIVER:
No error detected.
The correct code is 1 0 1 1 1 1 0
The decoded data is: 1011
```