

**AIM: Write a program for Hamming Code generation for error detection and correction.**

**PROGRAM:**

```
#include <stdio.h>

int main() {
    int data[7], received[7], i;
    int p1, p2, p4, syndrome;

    printf("Hamming(7,4) code using EVEN parity at positions 1, 2, and 4.\n");

    printf("Enter 4 data bits one by one (D1, D2, D3, D4):\n");

    // Insert data bits into correct positions:
    // Positions: [1] P1 [2] P2 [3] D1 [4] P4 [5] D2 [6] D3 [7] D4
    scanf("%d", &data[2]); // D1 at position 3
    scanf("%d", &data[4]); // D2 at position 5
    scanf("%d", &data[5]); // D3 at position 6
    scanf("%d", &data[6]); // D4 at position 7

    // Compute parity bits for even parity
    // P1 covers bits 1,3,5,7 → positions: 0,2,4,6
    data[0] = data[2] ^ data[4] ^ data[6];

    // P2 covers bits 2,3,6,7 → positions: 1,2,5,6
    data[1] = data[2] ^ data[5] ^ data[6];

    // P4 covers bits 4,5,6,7 → positions: 3,4,5,6
    data[3] = data[4] ^ data[5] ^ data[6];

    printf("\nEncoded 7-bit Hamming code:\n");
    for (i = 0; i < 7; i++) {
        printf("%d ", data[i]);
    }

    // Input received message (can be erroneous)
    printf("\n\nEnter received 7-bit message one by one:\n");
    for (i = 0; i < 7; i++) {
        scanf("%d", &received[i]);
    }

    // Recalculate parity checks for even parity
    p1 = received[0] ^ received[2] ^ received[4] ^ received[6]; // P1 group
    p2 = received[1] ^ received[2] ^ received[5] ^ received[6]; // P2 group
    p4 = received[3] ^ received[4] ^ received[5] ^ received[6]; // P4 group

    // Syndrome = binary value of parity errors → tells error position
```

```

syndrome = p4 * 4 + p2 * 2 + p1;

if (syndrome == 0) {
    printf("\nNo error detected. Message is correct.\n");
} else {
    printf("\nError detected at bit position: %d\n", syndrome);
    // Correct the bit by flipping
    received[syndrome - 1] ^= 1;

    printf("Corrected 7-bit message:\n");
    for (i = 0; i < 7; i++) {
        printf("%d ", received[i]);
    }
    printf("\n");
}

return 0;
}

```

## OUTPUT