COMPUTER NETWORKS LAB 7

~Pranay kumar karvi 23BDS1137

C CODE:-

```
        #include <stdio.h>
#include <math.h>

// Function to calculate the parity bit for a given position
int calculateParity(int data[], int n, int parityPosition) {
    int parity = 0;
    for (int i = 1; i <= n; i++) {
        if (i & (1 << (parityPosition - 1))) {
            parity ^= data[i - 1];
        }
    }
    return parity;
}

// Function to encode data using Hamming Code
void encodeHamming(int data[], int n, int encoded[]) {
    int parityBits = log2(n + 4);
    int j = 0, k = 0;

    // Insert data bits into the encoded array
    for (int i = 0; i < n + parityBits; i++) {
        if ((i + 1) & (i)) {
            encoded[i] = data[j++];
        } else {
            encoded[i] = -1; // Placeholder for parity bit
        }
    }

    // Calculate parity bits
    for (int i = 1; i <= parityBits; i++) {
        encoded[i - 1] = calculateParity(encoded, n + parityBits, i);
    }
}

// Function to decode the received bits and detect/correct errors
```

```
void decodeHamming(int encoded[], int n) {
    int parityBits = log2(n + 4);
    int errorPosition = 0;

    // Check for errors
    for (int i = 1; i <= parityBits; i++) {
        if (encoded[i - 1] != calculateParity(encoded, n, i)) {
            errorPosition += (1 << (i - 1));
        }
    }

    if (errorPosition) {
        printf("Error detected at position: %d\n", errorPosition);
        encoded[errorPosition - 1] ^= 1; // Correct the error
        printf("Corrected data: ");
        for (int i = 0; i < n; i++) {
            printf("%d ", encoded[i]);
        }
        printf("\n");
    } else {
        printf("No error detected.\n");
    }
}

// Function to print encoded data
void printEncodedData(int encoded[], int n) {
    printf("Encoded data: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", encoded[i]);
    }
    printf("\n");
}

int main() {
    int n;
    printf("Enter the number of data bits (e.g., 4): ");
    scanf("%d", &n);

    int data[n];  // Array to store user data bits
    printf("Enter the data bits (space-separated): ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &data[i]);
    }
```

```
    int encoded[n + 3]; // For 3 parity bits

    // Encode the data
    encodeHamming(data, n, encoded);
    printEncodedData(encoded, n + 3);

    // Optionally, introduce an error
    int introduceError;
    printf("Do you want to introduce an error? (1 for yes, 0 for no): ");
    scanf("%d", &introduceError);

    if (introduceError) {
        int errorPosition;
        printf("Enter the bit position to flip (1-%d): ", n + 3);
        scanf("%d", &errorPosition);
        if (errorPosition >= 1 && errorPosition <= n + 3) {
            encoded[errorPosition - 1] ^= 1; // Flip the bit to simulate error
            printf("Error introduced at position %d.\n", errorPosition);
        } else {
            printf("Invalid position! No error introduced.\n");
        }
    }

    // Decode the data and correct any errors
    decodeHamming(encoded, n + 3);

    return 0;
}
```

SCREENSHOTS-:

```
Enter the number of data bits (e.g., 4): 5
Enter the data bits (space-separated): 1 0 1 0 1
Encoded data: -2 -1 -2 -1 0 1 0 -1
Do you want to introduce an error? (1 for yes, 0 for no): 1
Enter the bit position to flip (1-8): 4
Error introduced at position 4.
Error detected at position: 7
Corrected data: -2 -1 -2 -2 0 1 1 -1


=== Code Execution Successful ===
```