Aim: write a program for Hamming code generation for error detection and correction

#include <stdio.h>

#include <math.h>

// Function to check if a number is a power of 2

int isPowerOfTwo(int x) {

return x && !(x & (x - 1));

}

int main() {

int data[20], hamming[30]; // data[] stores input bits, hamming[] stores final code

int dataBits, parityBits = 0, totalBits, i, j, k = 0;

// Input number of data bits

printf("Enter number of data bits: ");

scanf("%d", &dataBits);

// Input data bits from user (LSB to MSB)

printf("Enter data bits (LSB to MSB): ");

for (i = dataBits - 1; i >= 0; i--) {

scanf("%d", &data[i]); // Storing in reverse for MSB-first output later

}

// Calculate how many parity bits are needed:

// Total length = data bits + parity bits

// Condition: 2^p >= m + p + 1

while ((1 << parityBits) < (dataBits + parityBits + 1)) {

parityBits++;

}

totalBits = dataBits + parityBits;

// Fill hamming[] with data and parity placeholders

j = 0; // Index for data bits

for (i = 1; i <= totalBits; i++) {

if (isPowerOfTwo(i)) {

hamming[i] = 0; // Parity bit placeholder

} else {

hamming[i] = data[j++]; // Insert data bit

}

}

// Calculate parity bits using even parity rule

for (i = 0; i < parityBits; i++) {

int p = 1 << i; // Position of parity bit (1, 2, 4, 8, ...)

int parity = 0;

for (j = 1; j <= totalBits; j++) {

if (j & p) { // Check if p-th bit is set in position j

parity ^= hamming[j]; // XOR to calculate even parity

}

}

hamming[p] = parity; // Set calculated parity bit

}

// Print final Hamming code (MSB to LSB)

printf("Hamming code (MSB to LSB): ");

for (i = totalBits; i >= 1; i--) {

printf("%d", hamming[i]);

}

printf("\n");

return 0;

}

Output:

