# Name: Srujan Patwardhan

# Class: TY CS D

# Roll No: 2

# PRN: 12210847

**Code:**

**Hamming Code:**

import java.util.Scanner;

public class HammingCode {

    public static int[] generateHammingCode(int[] data) {

        int m = data.length;

        int r = 0;

        while (Math.pow(2, r) < (m + r + 1)) {

            r++;

        }

        int[] hammingCode = new int[m + r];

        for (int i = 0; i < r; i++) {

            hammingCode[(int) Math.pow(2, i) - 1] = -1;

        }

        int j = 0;

        for (int i = 0; i < hammingCode.length; i++) {

            if (hammingCode[i] != -1) {

                hammingCode[i] = data[j];

                j++;

            }

        }

        for (int i = 0; i < r; i++) {

            int position = (int) Math.pow(2, i);

            int parity = 0;

            for (int k = 1; k <= hammingCode.length; k++) {

                if (((k >> i) & 1) == 1 && k != position) { // right shift k by i positions and then check MSB is 1

                    parity ^= hammingCode[k - 1];

                }

            }

            hammingCode[position - 1] = parity;

        }

        return hammingCode;

    }

    public static int[] detectAndCorrect(int[] hammingCode) {

        int r = 0;

        while (Math.pow(2, r) < (hammingCode.length + 1)) {

            r++;

        }

        int errorPosition = 0;

        for (int i = 0; i < r; i++) {

            int position = (int) Math.pow(2, i);

            int parity = 0;

            for (int k = 1; k <= hammingCode.length; k++) {

                if (((k >> i) & 1) == 1) {

                    parity ^= hammingCode[k - 1];

                }

            }

            if (parity != 0) {

                errorPosition += position;

            }

        }

        if (errorPosition > 0) {

            System.out.println("Error detected at position: " + errorPosition);

            hammingCode[errorPosition - 1] ^= 1;

            System.out.println("Error corrected.");

        } else {

            System.out.println("No error detected.");

        }

        return hammingCode;

    }

    public static int[] extractData(int[] hammingCode) {

        int r = 0;

        int m = 0;

        while (Math.pow(2, r) < (hammingCode.length + 1)) {

            r++;

        }

        m = hammingCode.length - r;

        int[] data = new int[m];

        int j = 0;

        for (int i = 0; i < hammingCode.length; i++) {

            if ((i + 1 & i) != 0) {

                data[j++] = hammingCode[i];

            }

        }

        return data;

    }

    public static void printArray(int[] arr) {

        for (int i : arr) {

            System.out.print(i);

        }

        System.out.println();

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the ASCII code (7 or 8 bits): ");

        String input = scanner.next();

        int[] data = new int[input.length()];

        for (int i = 0; i < input.length(); i++) {

            data[i] = Character.getNumericValue(input.charAt(i));

        }

        int[] hammingCode = generateHammingCode(data);

        System.out.print("Generated Hamming code: ");

        printArray(hammingCode);

        System.out.print("Enter the received Hamming code for error detection: ");

        String receivedInput = scanner.next();

        int[] receivedCode = new int[receivedInput.length()];

        for (int i = 0; i < receivedInput.length(); i++) {

            receivedCode[i] = Character.getNumericValue(receivedInput.charAt(i));

        }

        int[] correctedCode = detectAndCorrect(receivedCode);

        System.out.print("Corrected Hamming code: ");

        printArray(correctedCode);

        int[] originalData = extractData(correctedCode);

        System.out.print("Extracted original data: ");

        printArray(originalData);

        scanner.close();

    }

}

**CRC:**

import java.util.Scanner;

public class CRC {

    public static String xor(String a, String b) {

        String result = "";

        int n = b.length();

        for (int i = 1; i < n; i++) {

            result += (a.charAt(i) == b.charAt(i)) ? "0" : "1";

        }

        return result;

    }

    public static String mod2div(String dividend, String divisor) {

        int pick = divisor.length();

        String tmp = dividend.substring(0, pick);

        int n = dividend.length();

        while (pick < n) {

            if (tmp.charAt(0) == '1') {

                tmp = xor(divisor, tmp) + dividend.charAt(pick);

            } else {

                tmp = xor("0".repeat(pick), tmp) + dividend.charAt(pick);

            }

            pick += 1;

        }

        if (tmp.charAt(0) == '1') {

            tmp = xor(divisor, tmp);

        } else {

            tmp = xor("0".repeat(pick), tmp);

        }

        return tmp;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the data: ");

        String data = scanner.next();

        System.out.print("Enter the divisor: ");

        String divisor = scanner.next();

        int len = divisor.length();

        String appendedData = data + "0".repeat(len - 1);

        String remainder = mod2div(appendedData, divisor);

        String codeword = data + remainder;

        System.out.println("The transmitted codeword is: " + codeword);

        System.out.print("Enter the received codeword: ");

        String receivedData = scanner.next();

        String checkRemainder = mod2div(receivedData, divisor);

        if (checkRemainder.contains("1")) {

            System.out.println("Error detected in the received codeword.");

        } else {

            System.out.println("No error detected in the received codeword.");

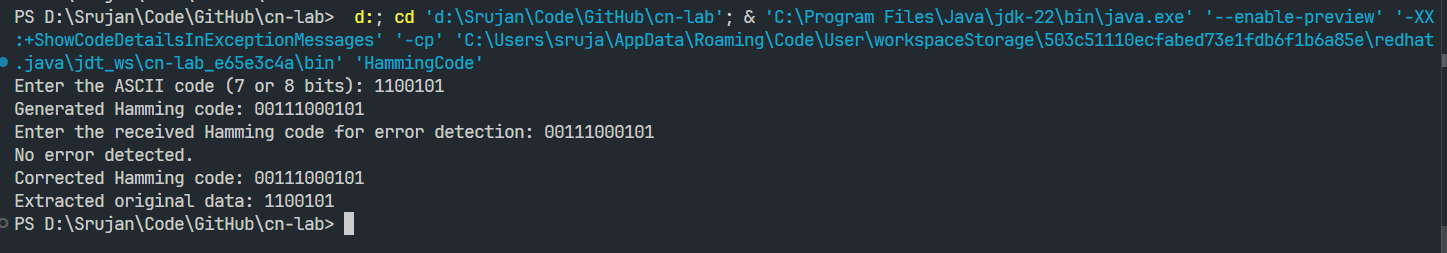
        }

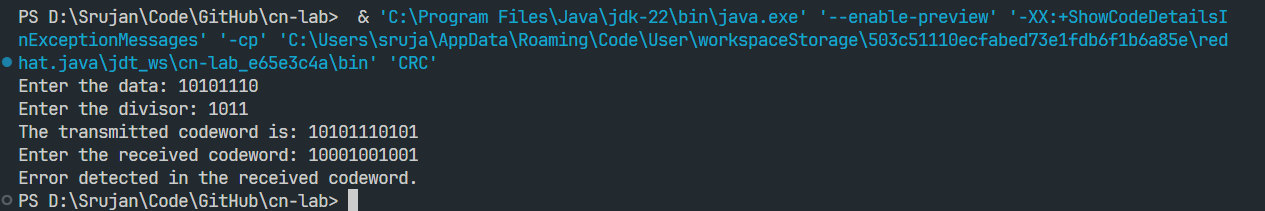
        scanner.close();

    }

}

**Output:**

****

****