Aim: Algorithm to implement data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

Algorithm

1. **Input** the data (binary digits).
2. **Choose** the generator polynomial depending on the CRC type.
3. **Append** the required number of zero bits (equal to degree of polynomial-1) to the end of data.
4. **Perform binary division** (modulo-2) using XOR.
5. The **remainder** after division is the **CRC checksum**.
6. **Append the CRC checksum** to the original binary data to get the final codeword.

Cyclic Redundancy Check for 12 bit Polynomial

Cyclic Redundancy Check for 16 bit Polynomial

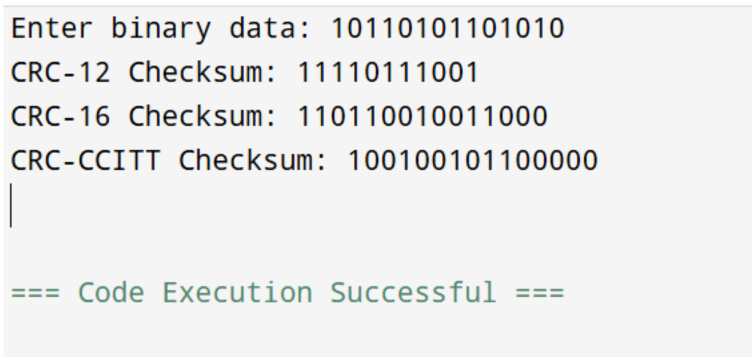
Cyclic Redundancy Check for Consultative Committee for International Telegraphy and Telephony 16 bit Polynomial

PROGRAM:

```
#include <stdio.h>

#include <string.h>

void xorOperation(char *dividend, char *divisor, int length)          #XOR Operation

{

   for (int i = 1; i < length; i++) {

      dividend[i - 1] = ((dividend[i] - '0') ^ (divisor[i] - '0')) + '0';

   }

}

// Perform Modulo-2 division

void computeCRC(char *data, char *poly, char *remainder, int polyLen) {

   int dataLen = strlen(data);                                        #XOR division

   char temp[128];

   strncpy(temp, data, polyLen);

   temp[polyLen] = '\0';


   for (int i = polyLen; i <= dataLen; i++) {

      if (temp[0] == '1')
```

```
            xorOperation(temp, poly, polyLen);
        else
            xorOperation(temp, "0000000000000000", polyLen); // **Dummy zero divisor**

        temp[polyLen - 1] = (i < dataLen) ? data[i] : '0';
    }

    strncpy(remainder, temp, polyLen - 1);                        #polyLen – 1 is CRC checksum
    remainder[polyLen - 1] = '\0';
}

int main() {
    char binData[128];
    printf("Enter binary data: ");
    scanf("%s", binData);

    char dataPadded[160], remainder[64];

    // CRC-12
    char crc12[] = "1100000001111";  // degree 12
    strcpy(dataPadded, binData);
    strcat(dataPadded, "00000000000"); // append 11 zeros
    computeCRC(dataPadded, crc12, remainder, 12);
    printf("CRC-12 Checksum: %s\n", remainder);

    // CRC-16
    char crc16[] = "11000000000000101"; // degree 16
    strcpy(dataPadded, binData);
    strcat(dataPadded, "000000000000000"); // append 15 zeros
    computeCRC(dataPadded, crc16, remainder, 16);
    printf("CRC-16 Checksum: %s\n", remainder);

    // CRC-CCITT
    char crcCCITT[] = "10001000000100001"; // degree 16
    strcpy(dataPadded, binData);
    strcat(dataPadded, "000000000000000"); // append 15 zeros
    computeCRC(dataPadded, crcCCITT, remainder, 16);
    printf("CRC-CCITT Checksum: %s\n", remainder);

    return 0;
}
```

OUTPUT



```
Enter binary data: 10110101101010
CRC-12 Checksum: 11110111001
CRC-16 Checksum: 110110010011000
CRC-CCITT Checksum: 100100101100000


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