Program 13

Write a program for error detecting code using CRC-CCITT (16 bits).

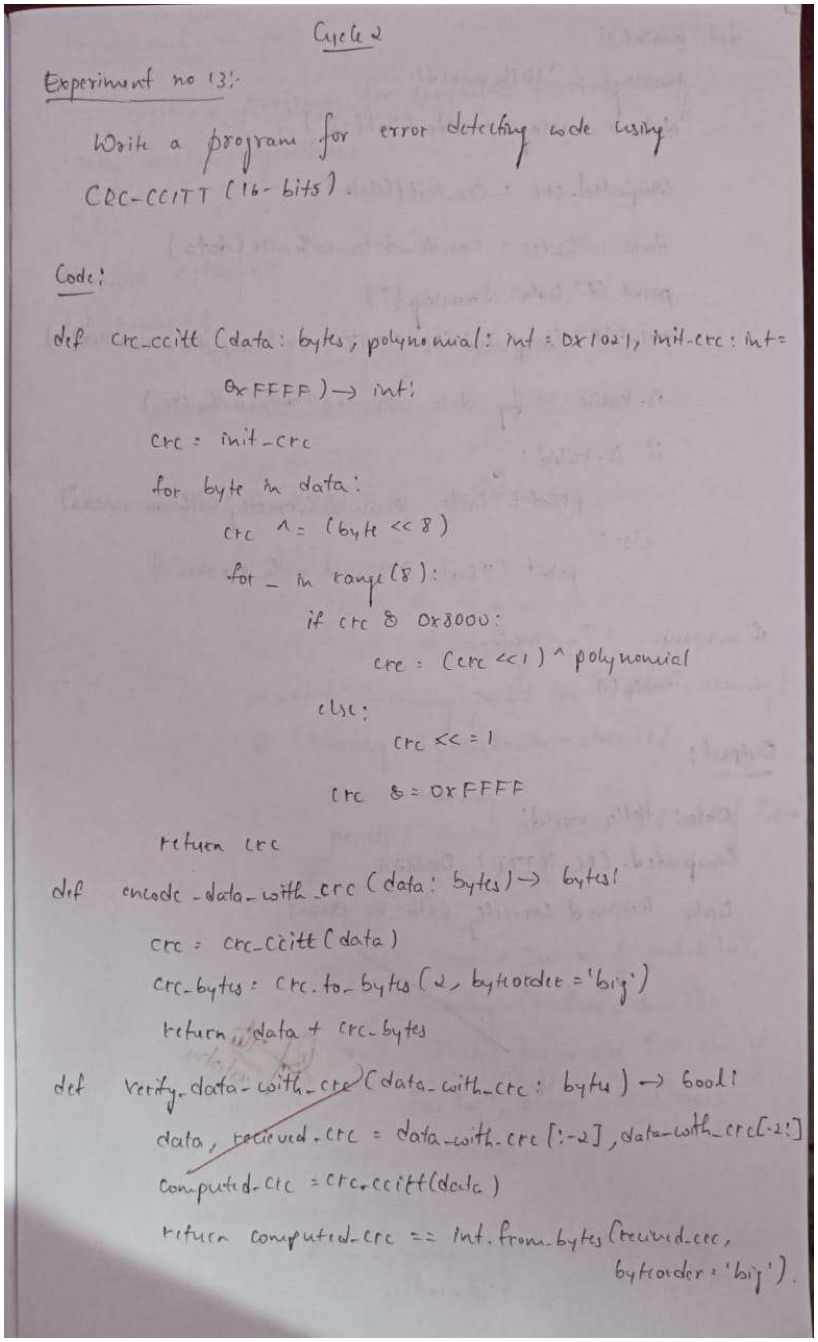
Code:

```
#include <stdio.h>
#include <stdint.h>
#define CRC_POLY 0x11021
#define INITIAL_CRC 0xFFFF

uint16_t compute_crc(uint8_t *data, size_t length) {
    uint16_t crc = INITIAL_CRC;
    for (size_t i = 0; i < length; i++) {
        crc ^= (data[i] << 8);
        for (int j = 0; j < 8; j++) {
            if (crc & 0x8000) {
                crc = (crc << 1) ^ CRC_POLY;
            } else {
                crc <<= 1;
            }
        }
    }
    return crc & 0xFFFF;
}

int check_crc(uint8_t *data, size_t length, uint16_t expected_crc) {
    uint16_t computed_crc = compute_crc(data, length);
    return (computed_crc == expected_crc);
}

int main() {
    uint8_t data[] = "Hello, World!";
    size_t data_length = sizeof(data) - 1;
    printf("Data: %s\n", data);
    uint16_t crc = compute_crc(data, data_length);
    printf("Computed CRC-CCITT: 0x%04X\n", crc);
    uint8_t received_data[] = "Hello, World!";
    size_t received_length = sizeof(received_data) - 1;
    if (check_crc(received_data, received_length, crc)) {
        printf("Data received correctly with no errors.\n");
    } else {
        printf("Error detected in received data!\n");
    }
    return 0;
}
```

Output:

```
Data: Hello, World!
Computed CRC-CCITT: 0x67DA
Data received correctly with no errors.
```

Figure 77: Output of CRC-CCIT



Cycle 2

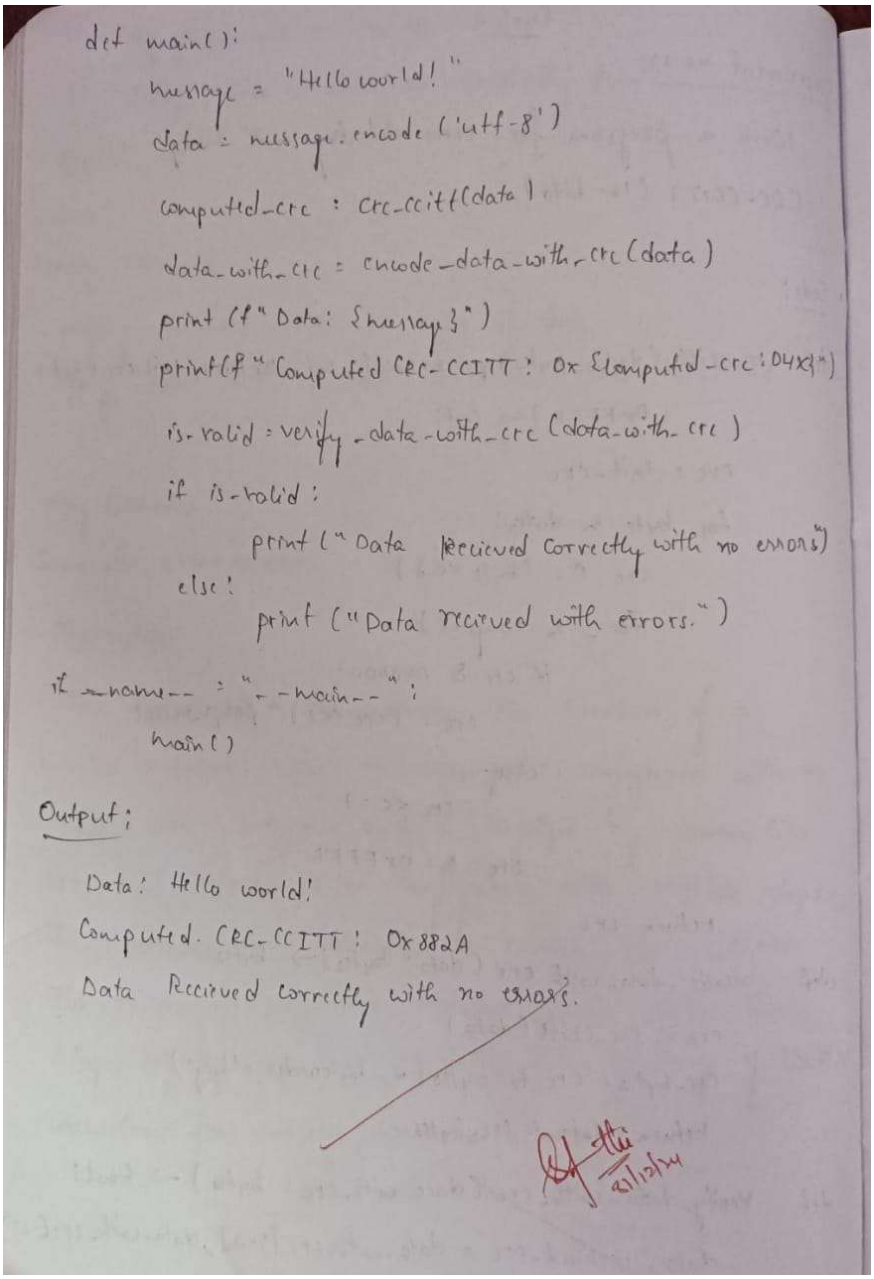
Experiment no 13:-

Write a program for error detecting code using CRC-CCITT (16-bits).

Code:

```
def crc_ccitt(data: bytes, polynomial: int = 0x1021, init_crc: int=
        0xFFFF) -> int:
    crc = init_crc
    for byte in data:
        crc ^= (byte << 8)
        for _ in range(8):
            if crc & 0x8000:
                crc = (crc << 1) ^ polynomial
            else:
                crc <<= 1
            crc &= 0xFFFF
    return crc
def encode_data_with_crc(data: bytes) -> bytes:
    crc = crc_ccitt(data)
    crc_bytes = crc.to_bytes(2, byteorder='big')
    return data + crc_bytes
def verify_data_with_crc(data_with_crc: bytes) -> bool:
    data, received_crc = data_with_crc[:-2], data_with_crc[-2:]
    computed_crc = crc_ccitt(data)
    return computed_crc == int.from_bytes(received_crc,
                                          byteorder='big').
```

Figure 78: Observation Book 1



```
def main():
    message = "Hello world!"
    data = message.encode('utf-8')
    computed_crc = crc_ccitt(data)
    data_with_crc = encode_data_with_crc(data)
    print(f"Data: {message}")
    print(f"Computed CRC-CCITT: 0x{computed_crc:04X}")
    is_valid = verify_data_with_crc(data_with_crc)
    if is_valid:
        print("Data Received correctly with no errors")
    else:
        print("Data received with errors.")
if __name__ == "__main__":
    main()
```

Output:

```
Data: Hello world!
Computed CRC-CCITT: 0x882A
Data Received correctly with no errors.
```

Figure 79: Observation Book 3