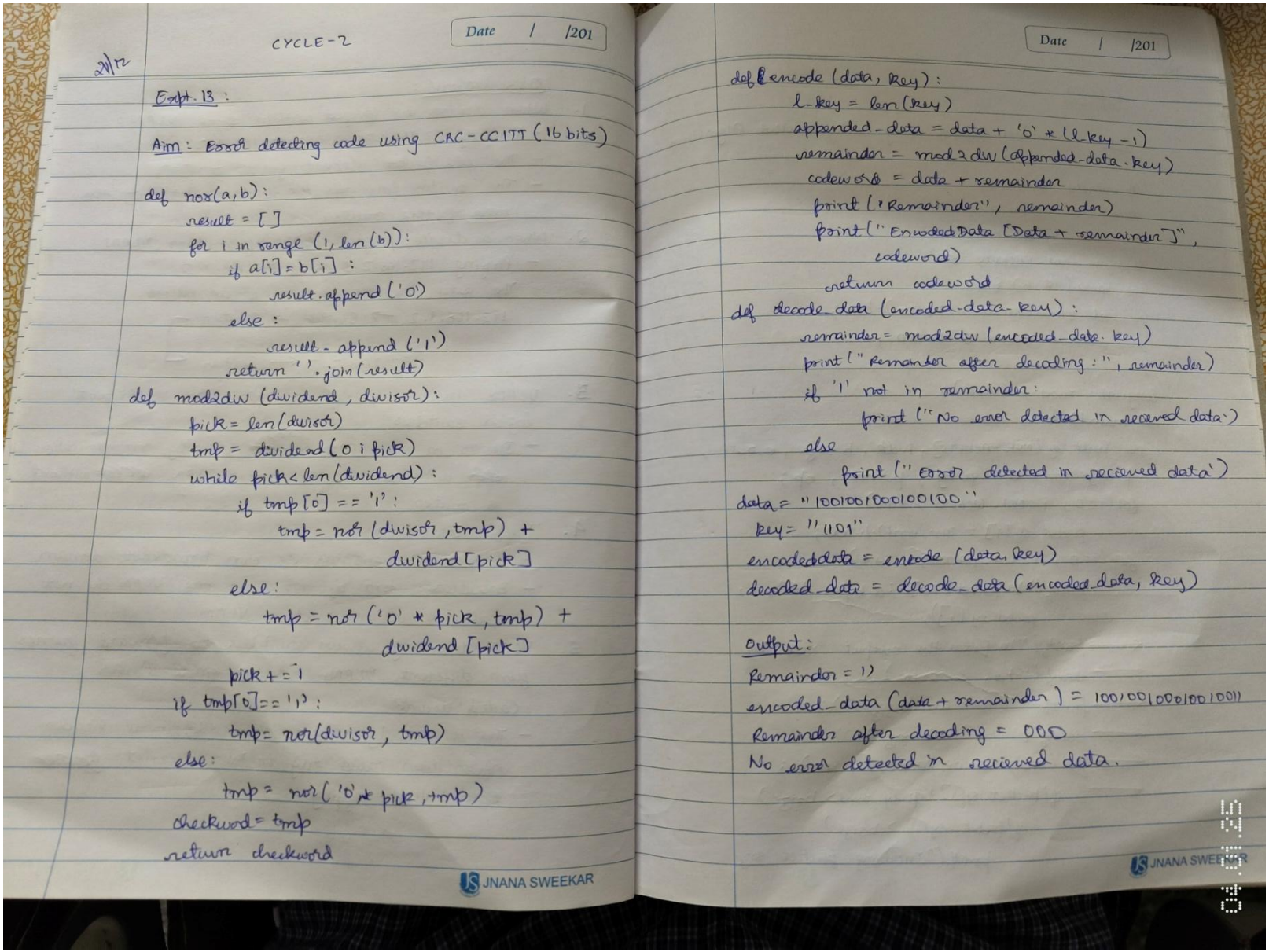
Expt. 13 – 21/12/2024



CYCLE-2

Date / /201

21/12

Expt. 13 :

Aim : Error detecting code using CRC-CCITT (16 bits)

```
def xor(a,b):
    result = []
    for i in range (1, len(b)):
        if a[i] = b[i] :
            result.append ('0')
        else :
            result.append ('1')
    return ''.join(result)
def mod2div (dividend, divisor):
    pick = len(divisor)
    tmp = dividend(0 : pick)
    while pick< len(dividend) :
        if tmp[0] == '1':
            tmp = xor (divisor, tmp) +
                        dividend[pick]
        else:
            tmp = xor ('0' * pick, tmp) +
                        dividend [pick]
        pick += 1
    if tmp[0]== '1' :
        tmp= xor(divisor, tmp)
    else:
        tmp = xor( '0'* pick, tmp)
    checkword = tmp
    return checkword
```

JNANA SWEEKAR

Date / /201

```
def encode (data, key) :
    l_key = len (key)
    appended_data = data + '0' * (l_key -1)
    remainder = mod2div (appended_data, key)
    codeword = data + remainder
    print ("Remainder", remainder)
    print ("Encoded Data [Data + remainder]",
            codeword)
    return codeword
def decode_data (encoded_data, key) :
    remainder = mod2div (encoded_data, key)
    print ("Remainder after decoding :", remainder)
    if '1' not in remainder:
        print ("No error detected in received data")
    else
        print ("Error detected in received data")
data = "100100100100100"
key = "1101"
encodedData = encode (data, key)
decoded_data = decode_data (encoded_data, key)
```

Output :

Remainder = 111

encoded_data (data + remainder) = 100100100100111

Remainder after decoding = 000

No error detected in received data.

JNANA SWEEKAR

Code:

```
#include <stdio.h>
#include <string.h>
#define N strlen(gen_poly)
char data[28], gen_poly[10], check[28];
int data_len, i, j;
void XOR() {
for (j = 0; j < N; j++) {
check[j] = (check[j] == gen_poly[j]) ? '0' : '1';
}
}
void crc() {
for (i = 0; i < N; i++) {
check[i] = data[i];
}
do {
if (check[0] == '1') {
XOR();
}
for (j = 0; j < N - 1; j++) {
check[j] = check[j + 1];
}
check[j] = data[i++];
} while (i <= data_len + N - 1);
}
void receiver() {
printf("\nData received: ");
scanf("%s", data);
crc();
for (i = 0; i < N - 1; i++) {
```

```
if (check[i] == '1') {
break;
}
}
if (i < N - 1) {
printf("\nERROR!");
} else {
printf("\nNO ERROR!");
}
}
5
1
int main() {
printf("\nEnter data: ");
scanf("%s", data);
printf("\nEnter generator: ");
scanf("%s", gen_poly);
data_len = strlen(data);
// Append N-1 zeros to the data
for (i = data_len; i < data_len + N - 1; i++) {
data[i] = '0';
}
data[data_len + N - 1] = '\0'; // Null-terminate the string
printf("\nData with padded 0's: %s", data);
crc();
printf("\nCheck sum: ");
for (i = 0; i < N - 1; i++) {
printf("%c", check[i]);
}
// Append checksum to data
```

```
for (i = data_len; i < data_len + N - 1; i++) {
data[i] = check[i - data_len];
}
data[data_len + N - 1] = '\0'; // Null-terminate the string
printf("\nFinal data to be transmitted: %s", data);
receiver();
return 0;
}
```

Output: