



**DALHOUSIE
UNIVERSITY**

Faculty of Computer Science

CSCI 6704 – Advanced Topics in Networks

Name: Arka Ghosh

Banner ID: B00911033

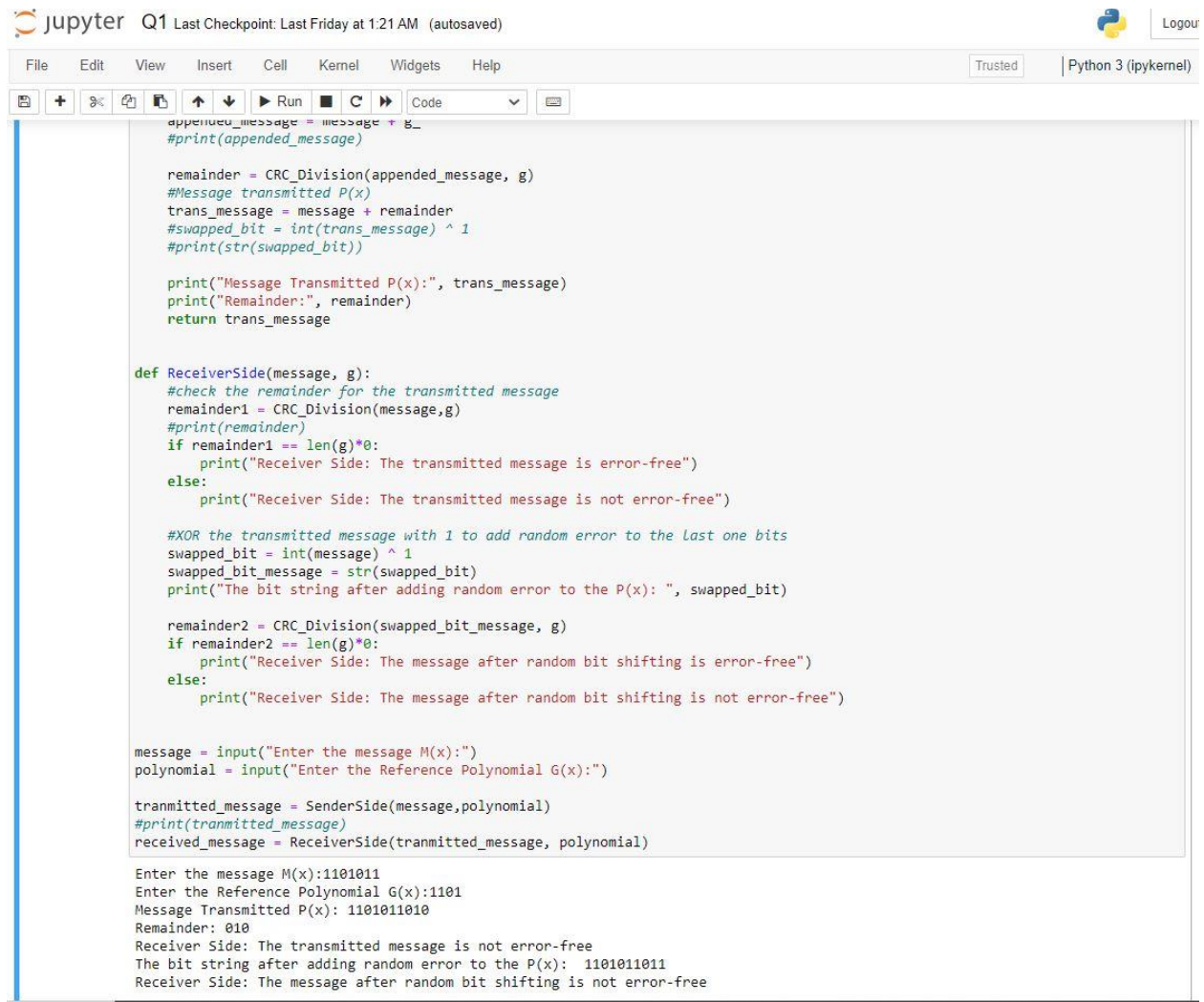
Assignment: 03 (Question 1)

Sample Input and Output for Question 1

The resulting outputs with the screenshot have been added below:

Test Case 1:

Enter the message M(x):1101011
Enter the Reference Polynomial G(x):1101
Message Transmitted P(x): 1101011010
Remainder: 010
Receiver Side: The transmitted message is not error-free
The bit string after adding random error to the P(x): 1101011011
Receiver Side: The message after random bit shifting is not error-free



The screenshot shows a Jupyter Notebook titled "Q1" with a last checkpoint from "Last Friday at 1:21 AM (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Trusted" and "Python 3 (ipykernel)".

The code in the notebook defines two functions: `SenderSide` and `ReceiverSide`. `SenderSide` appends the remainder of a message divided by a polynomial to the message. `ReceiverSide` checks if the received message is error-free by dividing it by the polynomial. If not, it flips the last bit of the message and checks again.

```
def SenderSide(message, g):
    appended_message = message + g_
    #print(appended_message)

    remainder = CRC_Division(appended_message, g)
    #Message transmitted P(x)
    trans_message = message + remainder
    #swapped_bit = int(trans_message) ^ 1
    #print(str(swapped_bit))

    print("Message Transmitted P(x):", trans_message)
    print("Remainder:", remainder)
    return trans_message

def ReceiverSide(message, g):
    #check the remainder for the transmitted message
    remainder1 = CRC_Division(message, g)
    #print(remainder)
    if remainder1 == len(g)*0:
        print("Receiver Side: The transmitted message is error-free")
    else:
        print("Receiver Side: The transmitted message is not error-free")

    #XOR the transmitted message with 1 to add random error to the last one bits
    swapped_bit = int(message) ^ 1
    swapped_bit_message = str(swapped_bit)
    print("The bit string after adding random error to the P(x): ", swapped_bit)

    remainder2 = CRC_Division(swapped_bit_message, g)
    if remainder2 == len(g)*0:
        print("Receiver Side: The message after random bit shifting is error-free")
    else:
        print("Receiver Side: The message after random bit shifting is not error-free")

message = input("Enter the message M(x):")
polynomial = input("Enter the Reference Polynomial G(x):")

transmitted_message = SenderSide(message, polynomial)
#print(transmitted_message)
received_message = ReceiverSide(transmitted_message, polynomial)
```

The output of the code is as follows:

```
Enter the message M(x):1101011
Enter the Reference Polynomial G(x):1101
Message Transmitted P(x): 1101011010
Remainder: 010
Receiver Side: The transmitted message is not error-free
The bit string after adding random error to the P(x): 1101011011
Receiver Side: The message after random bit shifting is not error-free
```

Test Case 2:

Enter the message $M(x)$: 10110011

Enter the Reference Polynomial $G(x)$: 11001

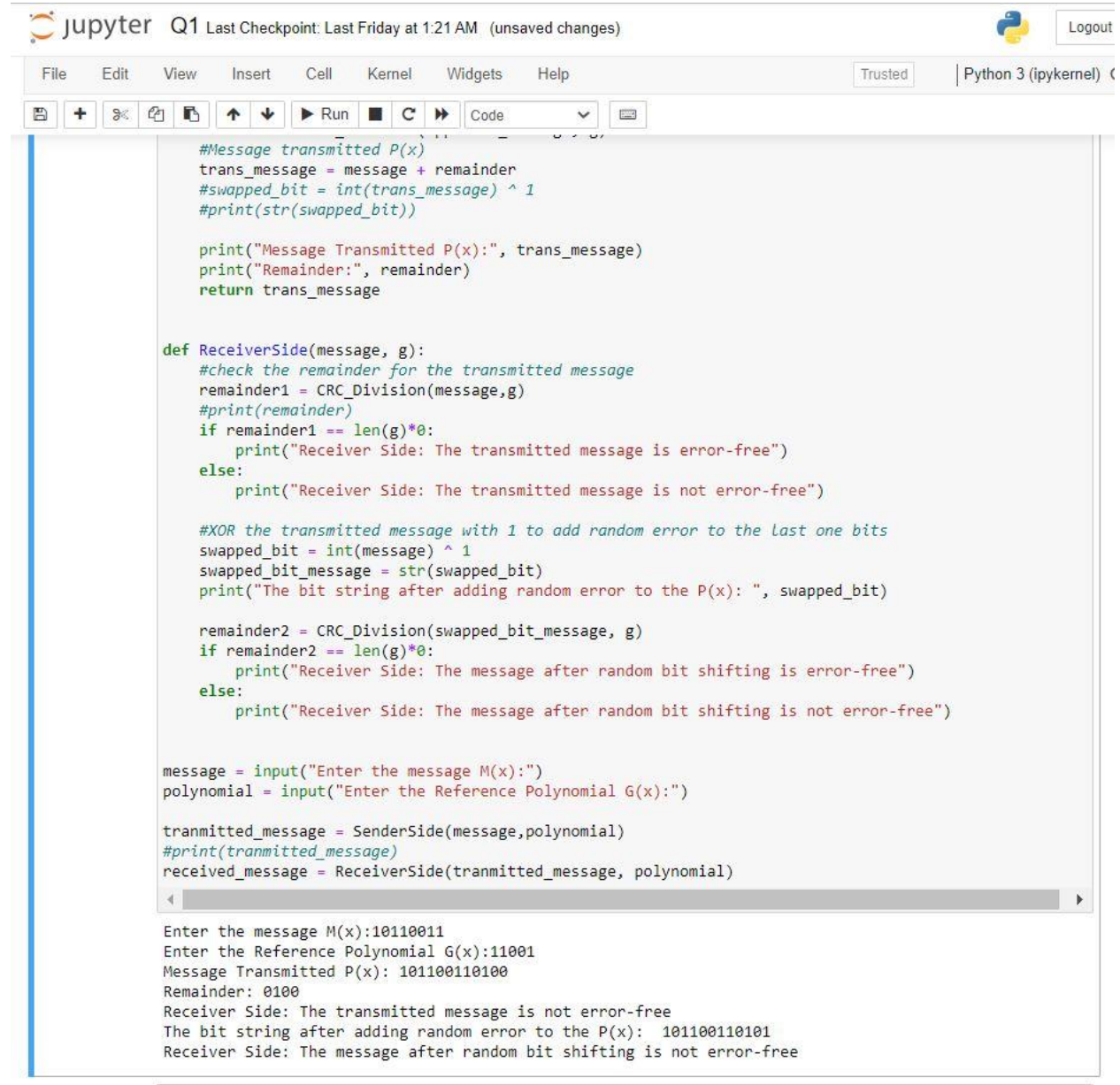
Message Transmitted $P(x)$: 101100110100

Remainder: 0100

Receiver Side: The transmitted message is not error-free

The bit string after adding random error to the $P(x)$: 101100110101

Receiver Side: The message after random bit shifting is not error-free



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and code execution, and a code editor. The code in the editor implements a CRC error detection algorithm. It defines a `SenderSide` function that calculates the remainder of a message divided by a polynomial and appends it to the message. It also defines a `ReceiverSide` function that checks if the received message's remainder is zero. If not, it flips the last bit of the message and recalculates the remainder. The notebook shows the execution of this code with the test case inputs: message $M(x)$ = 10110011 and polynomial $G(x)$ = 11001. The output shows that the transmitted message $P(x)$ is 101100110100 with a remainder of 0100. When the receiver side processes this, it finds the message is not error-free, flips the last bit to get 101100110101, and finds that the message after random bit shifting is still not error-free.

```
#Message transmitted P(x)
trans_message = message + remainder
#swapped_bit = int(trans_message) ^ 1
#print(str(swapped_bit))

print("Message Transmitted P(x):", trans_message)
print("Remainder:", remainder)
return trans_message

def ReceiverSide(message, g):
    #check the remainder for the transmitted message
    remainder1 = CRC_Division(message,g)
    #print(remainder)
    if remainder1 == len(g)*0:
        print("Receiver Side: The transmitted message is error-free")
    else:
        print("Receiver Side: The transmitted message is not error-free")

    #XOR the transmitted message with 1 to add random error to the Last one bits
    swapped_bit = int(message) ^ 1
    swapped_bit_message = str(swapped_bit)
    print("The bit string after adding random error to the P(x): ", swapped_bit)

    remainder2 = CRC_Division(swapped_bit_message, g)
    if remainder2 == len(g)*0:
        print("Receiver Side: The message after random bit shifting is error-free")
    else:
        print("Receiver Side: The message after random bit shifting is not error-free")

message = input("Enter the message M(x):")
polynomial = input("Enter the Reference Polynomial G(x):")

tranmitted_message = SenderSide(message,polynomial)
#print(tranmitted_message)
received_message = ReceiverSide(tranmitted_message, polynomial)

Enter the message M(x):10110011
Enter the Reference Polynomial G(x):11001
Message Transmitted P(x): 101100110100
Remainder: 0100
Receiver Side: The transmitted message is not error-free
The bit string after adding random error to the P(x): 101100110101
Receiver Side: The message after random bit shifting is not error-free
```

Test Case 3:

Enter the message M(x):1001100

Enter the Reference Polynomial G(x):1100

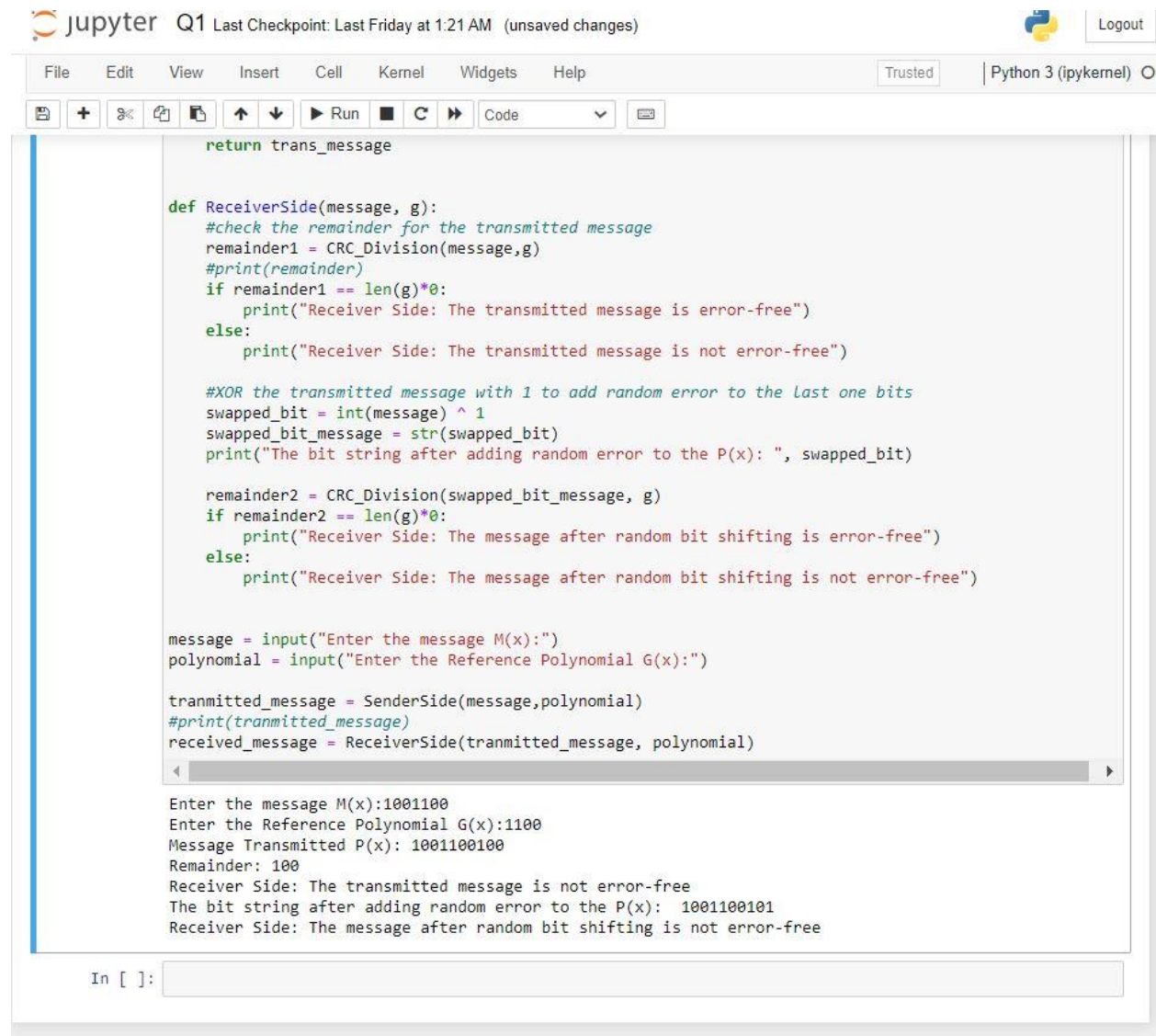
Message Transmitted P(x): 1001100100

Remainder: 100

Receiver Side: The transmitted message is not error-free

The bit string after adding random error to the P(x): 1001100101

Receiver Side: The message after random bit shifting is not error-free



```
return trans_message

def ReceiverSide(message, g):
    #check the remainder for the transmitted message
    remainder1 = CRC_Division(message,g)
    #print(remainder)
    if remainder1 == len(g)*0:
        print("Receiver Side: The transmitted message is error-free")
    else:
        print("Receiver Side: The transmitted message is not error-free")

    #XOR the transmitted message with 1 to add random error to the last one bits
    swapped_bit = int(message) ^ 1
    swapped_bit_message = str(swapped_bit)
    print("The bit string after adding random error to the P(x): ", swapped_bit)

    remainder2 = CRC_Division(swapped_bit_message, g)
    if remainder2 == len(g)*0:
        print("Receiver Side: The message after random bit shifting is error-free")
    else:
        print("Receiver Side: The message after random bit shifting is not error-free")

message = input("Enter the message M(x):")
polynomial = input("Enter the Reference Polynomial G(x):")

tranmitted_message = SenderSide(message,polynomial)
#print(tranmitted_message)
received_message = ReceiverSide(tranmitted_message, polynomial)

Enter the message M(x):1001100
Enter the Reference Polynomial G(x):1100
Message Transmitted P(x): 1001100100
Remainder: 100
Receiver Side: The transmitted message is not error-free
The bit string after adding random error to the P(x): 1001100101
Receiver Side: The message after random bit shifting is not error-free
```

In []: