ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ:

ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ

ΘΕΜΑ:ΥΛΟΠΟΙΗΣΗ CRC (ΜΕ ΧΡΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON)

Σύμφωνα με την εκφώνηση της εργασίας, καλλούμαστε να κατασκευάσουμε ενα πρόγραμμα το οποίο θα υλοποιεί τον αλγόριθμο ανίχνευσης λαθών CRC

1->ΔΕΔΟΜΕΝΑ

Στην παρακάτω εικόνα παρουσιάζονται τα δεδομένα και οι μεταβλητές που χρησιμοποιήθηκαν για την υλοποίηση του αλγορίθμου

```
# Main program
k = 20  # number of bits in D sequence
messages = 1000000  # number of messages that should be transmitted
D = []  # list of D parameters
T = []  # list of T parameters
P = input("Enter bits of P parameter(0 or 1 bits only accepted): ")
BER = float(input("Enter bit error(floating number between (0,1)): "))
count_true = 0 # variable that counts how many messages have been altered due to bit error rate
count_spotted_by_crc = 0
```

μεταβλητή k - > υποδεικνύει τον αριθμό των bits για τις ακολουθίες D

μεταβλητή messages - > συνολικός αριθμός τον μηνυμάτων που πρόκειται να μεταδοθούν (ακολουθία D)

λίστα D - > λίστα (πίνακας) που περιλαμβάνει όλα τα μηνύματα D που δημιουργούνται

λίστα T - > λίστα (πίνακας), όπου κάθε στοιχείο του είναι το μήνυμα D + FCS(Frame Checker Sequence)

μεταβλητή P - > Δυαδικός αριθμός που πρέπει να είναι διαίρεσιμη η ακολουθία T(Δίνεται απο τον χρήστη)

μεταβλητή BER - > Το bit error rate το οποίο δίνεται απο το χρήστη και αναδεικνύει τη πιθανότητα αλλοίωσης των μηνυμάτων T

μεταβλητή count_true - > Αριθμός μηνυμάτων που έχουν αλλοιωθεί λόγω του bit error rate

μεταβλητή count_spotted_by_crc - > Αριθμός μηνυμάτων που ανιχνεύονται εσφαλμένα απο το CRC

<u>NOTE: Να σημειωθεί οτί ο τύπος δεδομένων των μηνυμάτων (ακολουθίες D και T) είναι της κλάσης String.</u>

<u>2 - > ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ</u>

Στις 2 παρακάτω εικόνες φαίνεται ο αλγόριθμος υλοποίησης της αριθμητικής modulo 2, ο οποίος χρησιμοποιείται για τον υπολογισμό του FCS(Frame Check Sequence).

```python
# Performs Modulo-2 division
def modulo2(Divident, Divisor):
    # Number of bits to be XORed at a time.
    pick = len(Divisor)

    # Slicing the divident to appropriate
    # length for particular step
    temp = Divident[0: pick]

    while pick < len(Divident):
        if temp[0] == '1':
            # replace the divident by the result
            # of XOR and pull 1 bit down
            temp = xor(Divisor, temp) + Divident[pick]
        else:
            # If the leftmost bit of the dividend (or the
            # part used in each step) is 0, the step cannot
            # use the regular divisor; we need to use an
            # all-0s divisor
            temp = xor('0' * pick, temp) + Divident[pick]
        pick += 1

    # For the last n bits, we have to carry it out
    # normally as increased value of pick will cause
    # Index Out of Bounds
    if temp[0] == '1':
        temp = xor(Divisor, temp)
    else:
        temp = xor('0' * pick, temp)

    return temp
```

```
# function that performs XOR method
def xor(a, b):
    # initialize result
    result = []

    # Traverse all bits, if bits are
    # same, then XOR is 0, else 1
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')

    return ''.join(result)
```

Στη συνάρτηση modulo2, η οποία δέχεται 2 παραμέτρους (Divident-> το μήνυμα που πρόκειται να μεταδοθεί, δηλαδή η τιμή που διαιρείται

Divisor -> ο διαιρέτης, στη περίπτωση μας ειναι ο αριθμός P που δίνεται απο τον χρήστη)

χρησιμοποιείται η λίστα temp η οποία αναδεικνύει το υπόλοιπο της διαίρεσης μεταξύ divident και divisor και ενημερώνεται συνεχώς.

Επίσης, χρησιμοποιείται η μεταβλητή pick η οποία αναφέρεται σαν δείκτης έτσι ώστε να κατεβαίνει το κατάλληλο bit απο το μήνυμα κάθε φορά για να πραγματοποιηθούν οι πράξεις XOR.

Όσο η μεταβλητή pick δεν έχει ξεπεράσει τον πλήθος των bits του μηνύματος πραγματοποιούνται συνεχώς οι πράξεις XOR και ενημερώνεται το υπόλοιπο(temp). Όταν το πρώτο bit του υπολοίπου ειναι 1 τότε πραγματοποιοείται φυσιολογικά η πράξη XOR, κατεβαίνει ενα bit απο το μήνυμα και συνεχίζεται η ίδια διαδικασία. Σε περίπτωση που το πρώτο bit απο το υπόλοιπο είναι 0 πρέπει να μετακινηθεί κατα 1 bit από τη διαίρεση μέχρι το πρώτο bit του υπολοίπου να γινει 1.

Τέλος, η συνάρτηση επιστρέφει το υπόλοιπο της διαίρεσης (Divident με Divisor).

```python
# function that generates the altered message using the bit error rate
def MessageWithNoise(Message, bit_error):
    message_with_noise = copy.deepcopy(Message)
    flag = False
    for i in range(len(message_with_noise)):
        # choosing random number between 0 and 1 (float numbers only)
        random_number = rand.uniform(0, 1)
        # if the generated number is smaller than the bit error, we alter the i_th bit
        # of the message
        if random_number < bit_error:
            flag = True
            if message_with_noise[i] == '1':
                message_with_noise = message_with_noise[:i] + '0' + message_with_noise[i + 1:]
            else:
                message_with_noise = message_with_noise[:i] + '1' + message_with_noise[i + 1:]
    return message_with_noise, flag
```

Η παραπάνω εικόνα παρουσιάζει την συνάρτηση η οποία αλλοιώνει το μήνυμα που μεταδίδεται με βάση το Bit Error Rate.

Για κάθε bit του μηνύματος (Όσο τρέχει η επανάληψη) δίνεται

ένας τυχαίος πραγματικός αριθμός στο διάστημα (0,1). Σε περίπτωση που ο αριθμός είναι μικρότερος απο το bit error rate τοτε αλλάζει ένα bit της ακολουθίας(απο 0 δηλαδή γίνεται 1 και το αντίστροφο).

Τέλος, η συνάρτηση επιστρέφει το αλλοιωμένο μήνυμα(διαφορετικά επιστρέφεται το ίδιο μήνυμα) καθώς και τη μεταβλητή flag, η οποία είναι τύπου δεδομένων boolean και είναι True όταν έχει γίνει αλλοίωση του μηνύματος και False σε διαφορετική περίπτωση.

3 - > ΚΑΤΑΓΡΑΦΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

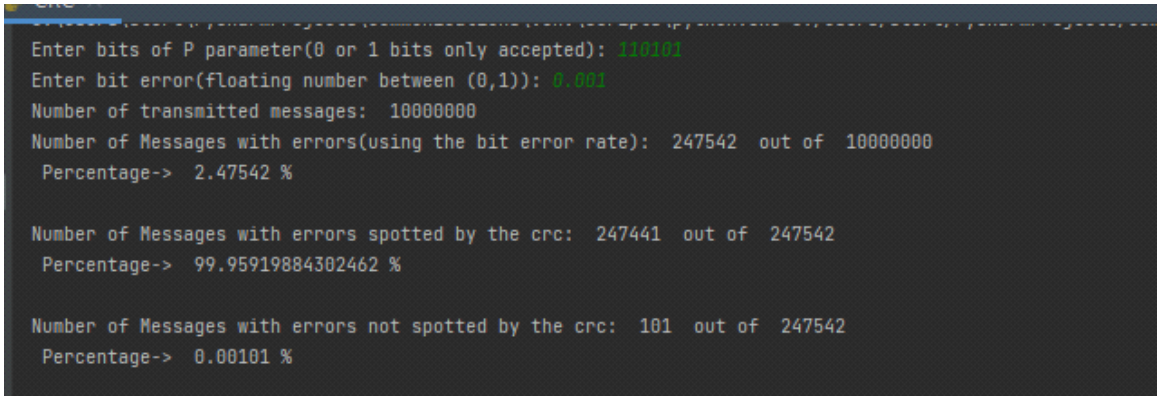
Για k = 20, P = 110101, BER = 0.001 καταγράφτηκαν τα εξής ποσοστά:

Για 10.000.000 μηνύματα έχουμε:

1)Ποσοστό μηνυμάτων που φθάνουν με σφάλμα στον αποδέκτη-> 2.47542 % (247542 μηνύματα απο τα 10.000.000)

2)Ποσοστό μηνυμάτων που ανιχνεύονται ως εσφαλμένα απο το CRC ->99.95919884302462 % (247441 μηνύματα απο τα 247542)

3)Ποσοστό μηνυμάτων που φθάνουν με σφάλμα στον αποδέκτη και δεν ανιχνεύονται απο το CRC - > 0.00101 % (101 απο τα 247542)



Φοιτητής 2ου έτους Πληροφορικής:Μουμτζής Στέργιος

ΑΕΜ:3620