Batch:A2

Roll Number: 16010421063 **Experiment Number:6**

Name:Arya Nair

Title of the Experiment:Error detection and correction

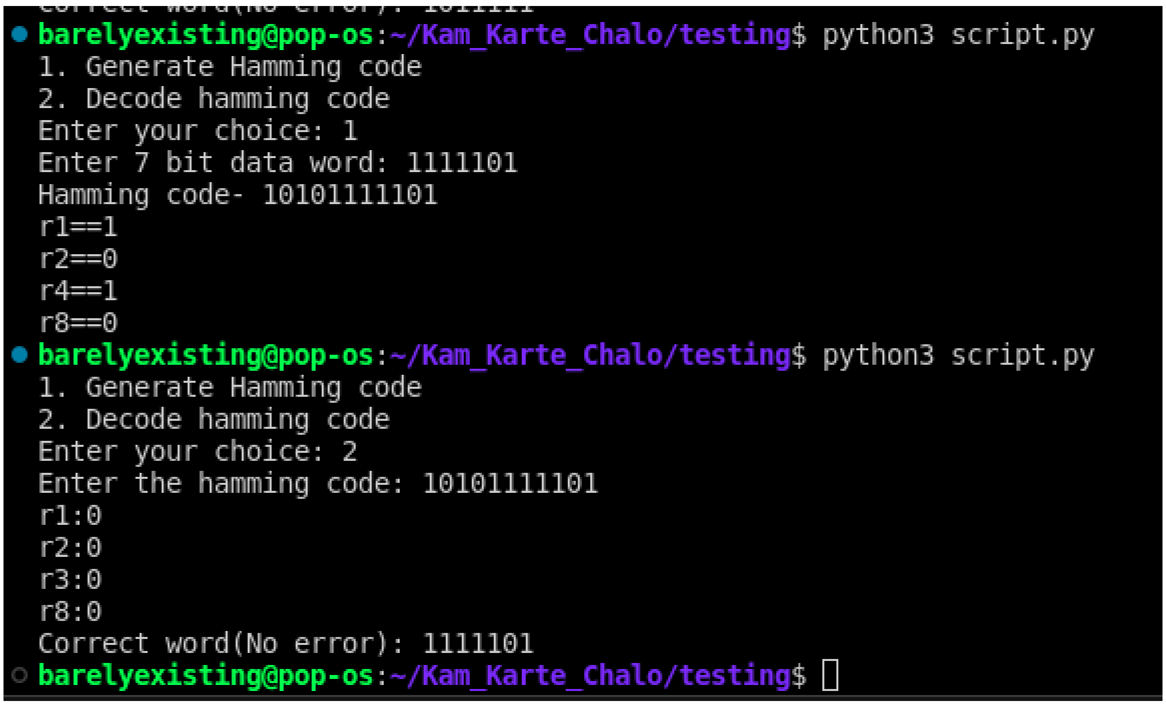
Program:

```
def Transmitter():
    DW=input("Enter 7 bit data word: ")
    DW=DW[::-1]

    if(len(DW)<7 or len(DW)>7):
        print("Invalid data word")
        return

    for i in DW:
        if i=='0' or i=='1':
            pass
        else:
            print("Invalid bit value")
            return



    codeword=''

    r1=int(DW[6])^int(DW[5])^int(DW[3])^int(DW[2])^int(DW[0])
    r2=int(DW[6])^int(DW[4])^int(DW[3])^int(DW[1])^int(DW[0])
    r4=int(DW[5])^int(DW[4])^int(DW[3])
    r8=int(DW[2])^int(DW[1])^int(DW[0])


    codeword=DW[:3]+str(r8)+DW[3:6]+str(r4)+DW[6]+str(r2)+str(r1)

    print(f"Hamming code- {codeword}")

    print(f"r1=={r1}")

    print(f"r2=={r2}")

    print(f"r4=={r4}")

    print(f"r8=={r8}")





def Receiver():

    code=input('Enter the hamming code: ')

    reversed_code=code[::-1]


    if len(reversed_code)!=11:

        print("Invalid Code")

        return


    for i in reversed_code:

        if i=='0' or i=='1':

            pass

        else:

            print("Invalid bit value")

            return
```

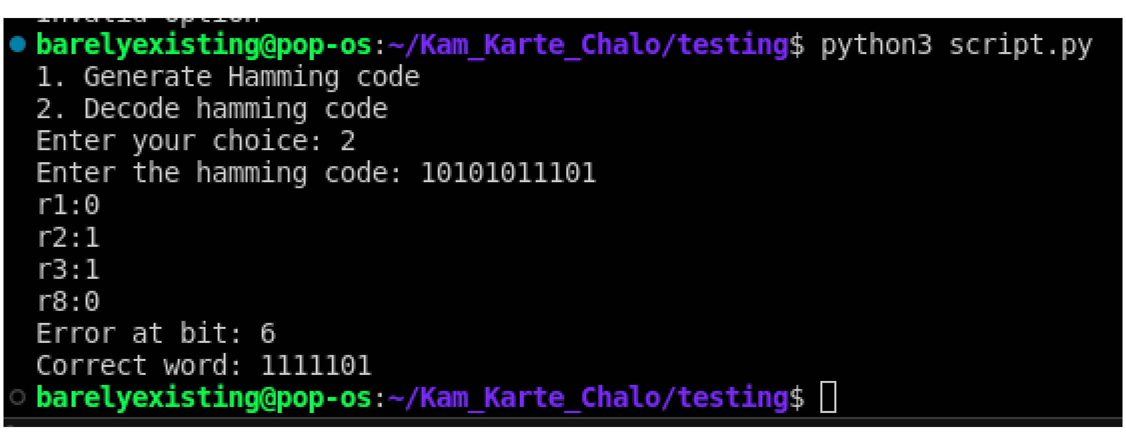
```
r1=str(int(reversed_code[2])^int(reversed_code[4])^int(reversed_code[6]
)^int(reversed_code[8])^int(reversed_code[10])^int(reversed_code[0]))

r2=str(int(reversed_code[2])^int(reversed_code[5])^int(reversed_code[6]
)^int(reversed_code[9])^int(reversed_code[10])^int(reversed_code[1]))

r4=str(int(reversed_code[4])^int(reversed_code[5])^int(reversed_code[6]
)^int(reversed_code[3]))

r8=str(int(reversed_code[8])^int(reversed_code[9])^int(reversed_code[10
])^int(reversed_code[7]))


    print(f"r1:{r1}\nr2:{r2}\nr3:{r4}\nr8:{r8}")


    if(r1+r2+r4+r8=="0000"):


        data=reversed_code[2]+reversed_code[4:7]+reversed_code[8:]


        print(f"Correct word(No error): {data}")
    else:


        error_bin=r8+r4+r2+r1
        error_dec=int(error_bin,2)


        print(f"Error at bit: {error_dec}")
        error_dec-=1
```

```
       if reversed_code[error_dec]=='1':

reversed_code=reversed_code[:error_dec]+'0'+reversed_code[error_dec+1:]

       else :

reversed_code=reversed_code[:error_dec]+'1'+reversed_code[error_dec+1:]


       data=reversed_code[2]+reversed_code[4:7]+reversed_code[8:]


       print(f"Correct word: {data}")



if __name__=='__main__':

   print("1. Generate Hamming code\n2. Decode hamming code")

   option=input("Enter your choice: ")

   if option=='1':

       Transmitter()

   elif option=='2':

       Receiver()

   elif option=='3':

       print("Exiting ...")

   else:

       print("Invalid Option")
```

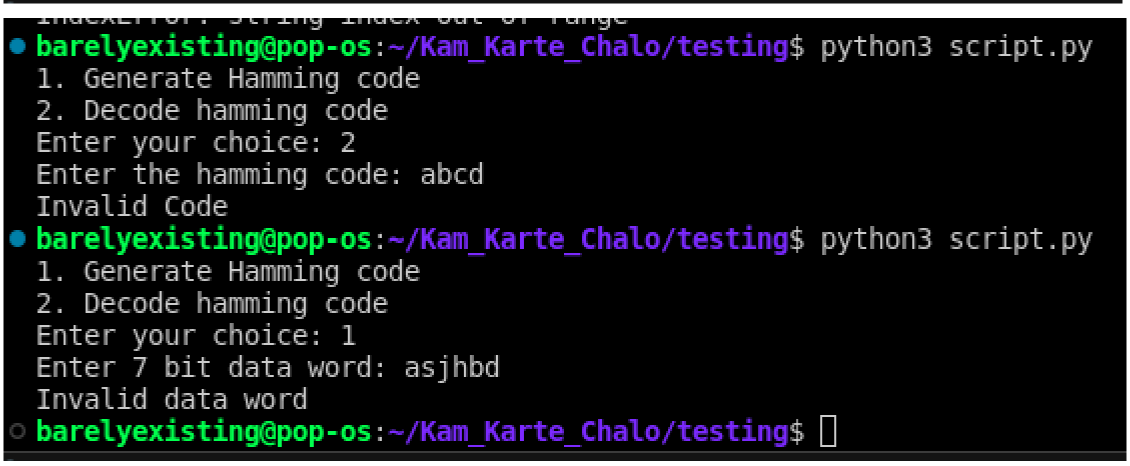
Output:



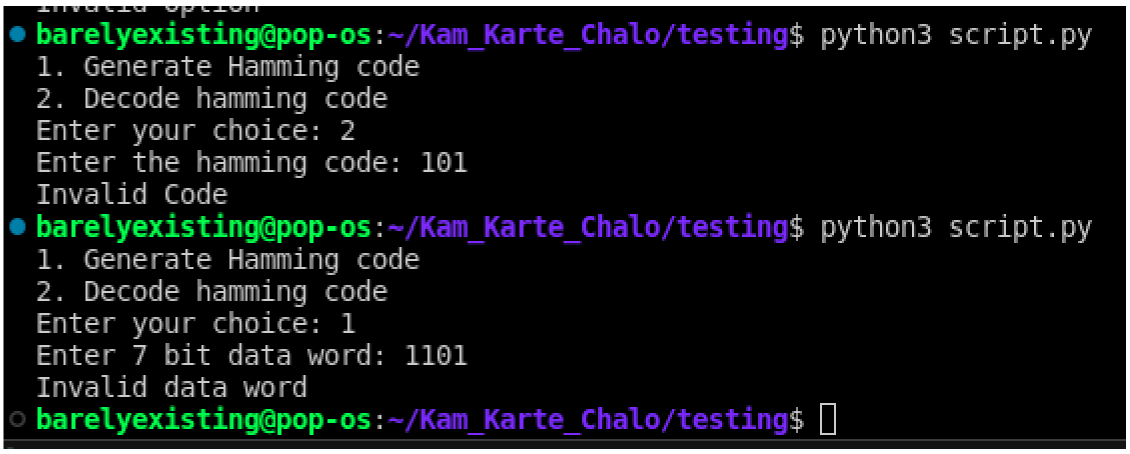
```
Correct word(No error): 1011111
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 1
Enter 7 bit data word: 1111101
Hamming code- 10101111101
r1==1
r2==0
r4==1
r8==0
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 2
Enter the hamming code: 10101111101
r1:0
r2:0
r3:0
r8:0
Correct word(No error): 1111101
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ 
```



```
Invalid option
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 2
Enter the hamming code: 10101011101
r1:0
r2:1
r3:1
r8:0
Error at bit: 6
Correct word: 1111101
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ 
```



```
IndexError: string index out of range
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 2
Enter the hamming code: abcd
Invalid Code
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 1
Enter 7 bit data word: asjhbd
Invalid data word
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ 
```



```
Invalid option
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 2
Enter the hamming code: 101
Invalid Code
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ python3 script.py
1. Generate Hamming code
2. Decode hamming code
Enter your choice: 1
Enter 7 bit data word: 1101
Invalid data word
barelyexisting@pop-os:~/Kam_Karte_Chalo/testing$ 
```

Post Lab Question- Answers (If Any):

1. What are the different methods used for error detection
   Ans: Detection:
   1. 2d parity check
   2. Checksum
   3. Cyclic redundancy Check
   4. Simple parity
   Correction:
   1. Hamming code

2. Which layer of the OSI model usually does the function of error detection? Ans
   Ans: Data link layer does the error detection

3. If the data unit is 111111 and the divisor is 1010,wht is the dividend at the Transmitter?
   Ans dividend 1.01

1. What is Hamming distance ? What is minimum Hamming distance?
   Ans Hamming distance is a metric for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the number of bit positions in which the two bits are different.The minimum Hamming distance is used to define some essential notions in coding theory, such as error detecting and error correcting codes. … In other words, a code is k-errors correcting if, and only if, the minimum Hamming distance between any two of its codewords is at least 2k+1.. What

is the minimal Hamming distance between any two correct codewords? Explanation: Since we use a generalized version of Hamming(7, 4) code, the minimal hamming distance is 3. It cannot correct burst errors.

CO4: Execute their knowledge of computer communication principles, including error detection and correction, multiplexing, flow control and error control

---

Conclusion: We understood the implementation of receiver and transmitter of hamming code

---