

Assignment 8

ELP - 718 Telecommunication Software Laboratory

Srinivas Anjay Kumar

2019JTM2169

2019-2021

A report presented for the assignment on
Python and Github



**Bharti School Of
Telecommunication Technology and Management
IIT Delhi
India**

September 18, 2019

Contents

1	Problem Statement-1	4
1.1	Problem Statement	4
1.2	Assumptions	4
1.3	Algorithm and Implementation	4
1.4	Input-Output Format	5
1.5	Difficulties Faced	5
1.6	Program Structure	6
1.7	Screenshots	7
2	Problem Statement-2	8
2.1	Problem Statement	8
2.2	Sample Output:	8
2.3	Assumptions	9
2.4	Algorithm and Implementation	9
2.5	Difficulties Faced	9
2.6	Program Structure	10
2.7	Screenshots	11
3	Git	12
4	Appendix	13
4.1	Code for ps1	13
4.2	Code for ps2	14

List of Figures

1	Screenshot1	7
2	Flowchart-2	10
3	Screenshot2	11
4	Screenshot3	11

1 Problem Statement-1

1.1 Problem Statement

Parity Check

The simplest way of error detection is to append a single bit, called a parity check, to a string of data bits. This parity check bit has the value 1 if the number of 1's in the bit string is even and has the value 0 otherwise, i.e., Odd Parity Check.

Bit-Oriented Framing

Data Link Layer needs to pack bits into frames so that each frame is distinguishable from another. Frames can be fixed or variable size. In variable size framing, we define the end of the frame using a bit-oriented approach. It uses a special string of bits, called a flag for both idle fills and to indicate the beginning and the ending of frames.

The bit stuffing rule is to insert a 0 after each appearance of 010 in the original data. The string 0101 is used as the bit string or flag to indicate the end of the frame.

1.2 Assumptions

The only Assumption is:

- We know python 3 syntax

1.3 Algorithm and Implementation

The Algorithm is :

1. Take User input as a string
2. Now apply a for loop in that string.
3. Now checkfor each character is it '1' or is it '0'
4. if character is '1' add the counter by 1
5. Now outside the loop check whether counter is even or odd number
6. If counter is even add a '1' at the end otherwise add a '0'
7. Now this parity data has to be bit stuffed
8. Check for 010 in that string
9. If 010 occurs add a '0' at the end of 010
10. Similarly check for 010 pattern in the whole string.
11. finally append 0101 string to bit stuffed result

1.4 Input-Output Format

Input Format

Enter binary bit data that has to be transmitted.

Output Format

Print binary bit data with parity bit.

Print the modified string that is to be transmitted

Sample Input

010101110100101

Sample Output

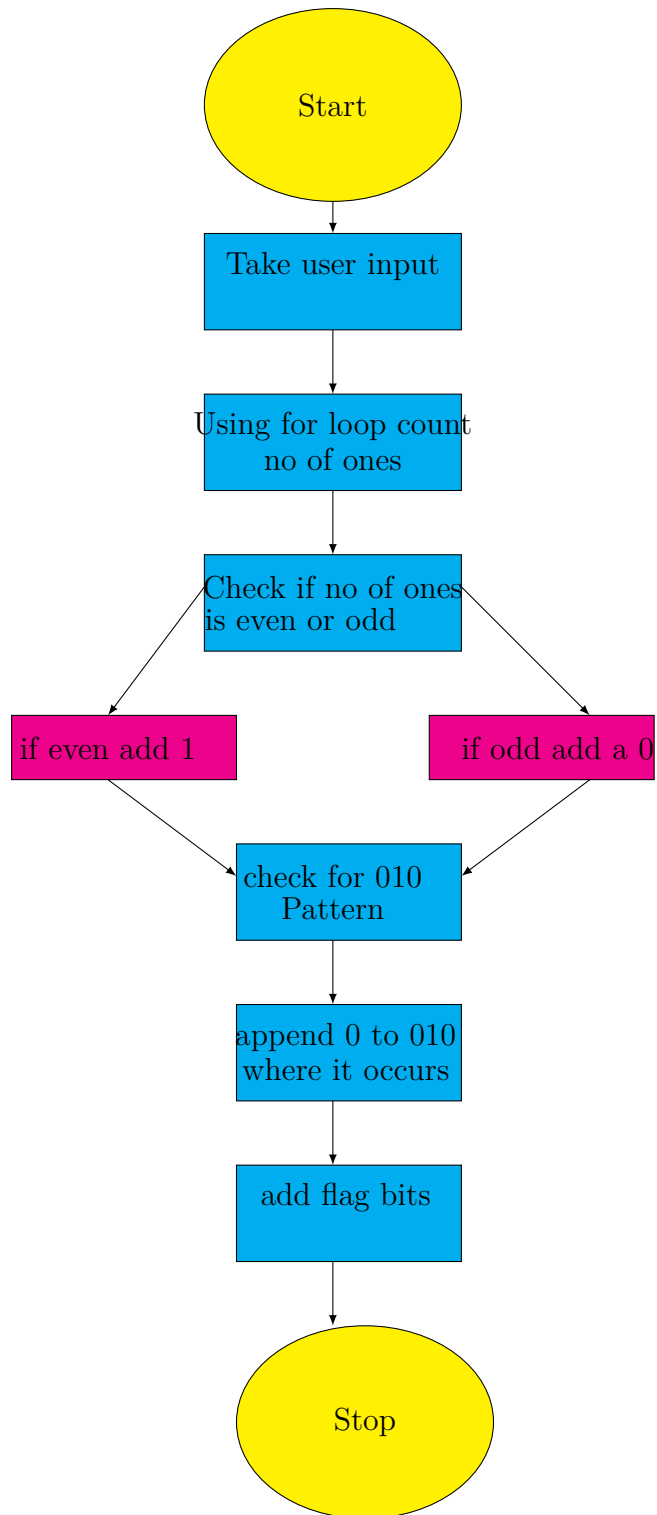
Parity bit data : 0101011101001011

Transmitting data: 01001011101000100110101

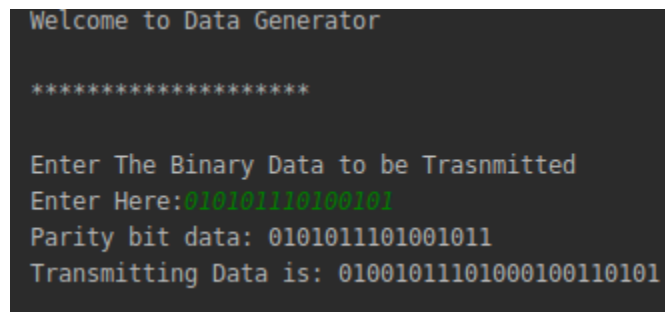
1.5 Difficulties Faced

1. New to Python so syntax was a problem
2. Bit stuffing logic implementation was hard

1.6 Program Structure



1.7 Screenshots



```
Welcome to Data Generator

*****

Enter The Binary Data to be Trasnmitted
Enter Here:010101110100101
Parity bit data: 0101011101001011
Transmitting Data is: 01001011101000100110101
```

Figure 1: Screenshot1

2 Problem Statement-2

2.1 Problem Statement

3X3 Numeric Tic-Tac-Toe (Use numbers 1 to 9 instead of X's and O's)

One player plays with the odd numbers (1, 3, 5, 7, 9) and the other player plays with the even numbers (2,4,6,8). All numbers can be used only once. The player who puts down 15 points in a line wins (sum of 3 numbers). Always Player with odd numbers starts the game. Once a line contains two numbers whose sum is 15 or greater, there is no way to complete that line, although filling in the remaining cells might be necessary to complete a different line.

Note - Line can be horizontal, vertical or diagonal

Constraints

- $1 \leq Position \leq 9$
- $1 \leq Number \leq 9$

Terminal

- Print 'Welcome to the Game!'.
- Print whether it is Player 1's or Player 2's chance.
- Get the position and number to be entered from the user.
- Show tic tac toe with data.
- Continue till the game gets draw or some player wins and show the result.
- Ask the user whether to continue for the next game or exit.

2.2 Sample Output:

Welcome to the Game!

Player 1's chance

Enter the position and number to be entered: 5,3

	3	

Player 2's chance
Enter the position and number to be entered: 7,4

	3	
4		

... Continue till game ends

2.3 Assumptions

The Assumptions are:

- Each Player plays one by one
- They know the rules of the game

2.4 Algorithm and Implementation

The Algorithm is :

1. Display the welcome menu
2. Explain the rules
3. Enter for player1 in the constraints if input of of bound again ask for input
4. Similarly for Player 2
5. Check the sum of matrix Horizontly, Vertically and Diagonally
6. If any players sum which is any fashion defined above is greater than or equal to 15.
7. That player is the winner
8. If no winner than print match tied

2.5 Difficulties Faced

1. Matrix Implementation in python
2. Time Constraints
3. Out of bounds error in List
4. Logic Implementation

2.6 Program Structure

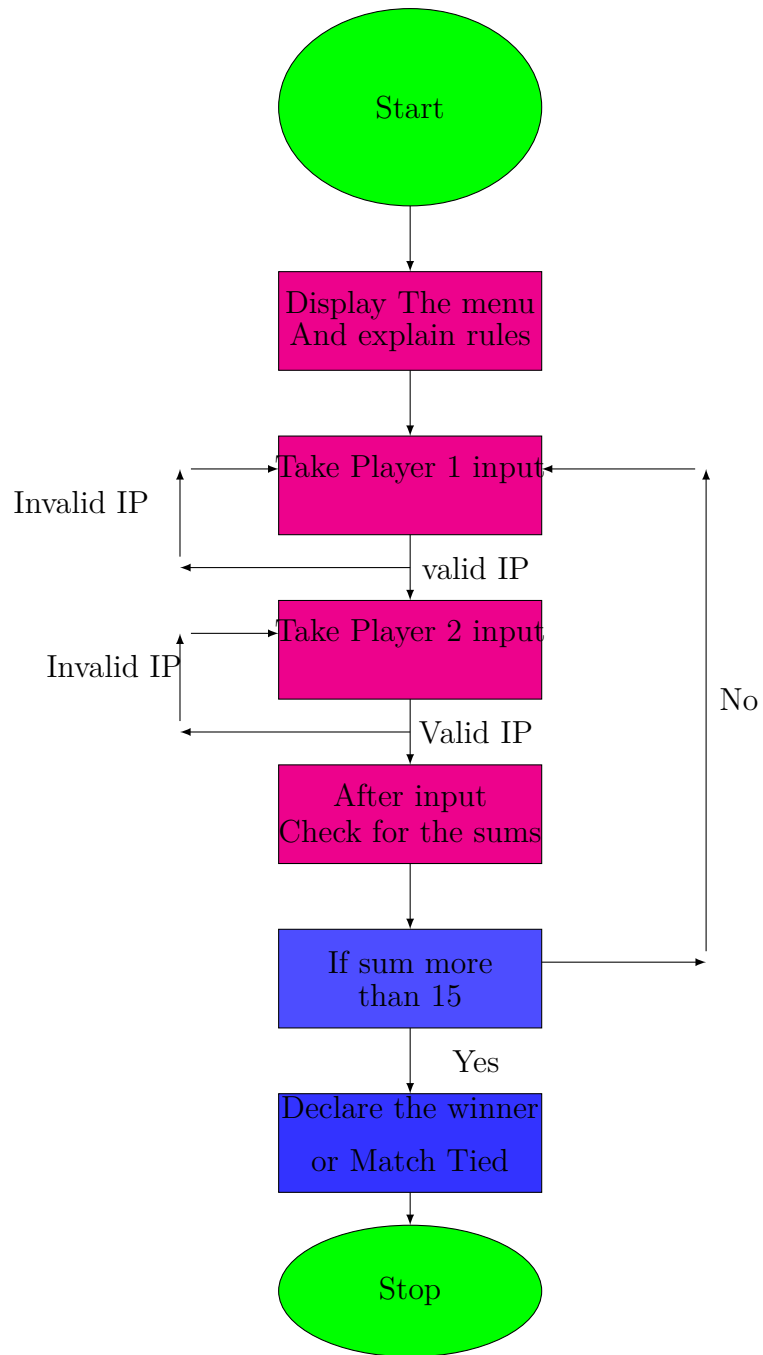


Figure 2: Flowchart-2

2.7 Screenshots

```
-----  
Welcome to the Game  
-----  
Rules of the Game:  
Player 1 will use only odd numbers[1-9]  
Player 2 will use only even numbers[1-9]  
Current State Of Matrix  
000  
000  
000  
Player 1 Enter the Position in which number is to be Entered:1  
Player 1 Enter the number which is to be Entered:7  
Current State Of Matrix  
700  
000  
000
```

Figure 3: Screenshot2

```
Current State Of Matrix  
775  
622  
000  
PLayer 1 is the winner
```

Figure 4: Screenshot3

3 Git

```
commit 451d3b1c4472c6642c6e21ca16bd14a3e23c01 (HEAD -> master, origin/master)
Author: Anjay <kumaranjay8@gmail.com>
Date: Wed Sep 18 15:42:48 2019 +0530

    Report

commit 539295067ea07ab88645112abe5260b48c2719e5
Author: Anjay <kumaranjay8@gmail.com>
Date: Wed Sep 18 14:18:25 2019 +0530

    2nd almost completed

commit dcac2a7c923db18badf3ce0a17dbfd5e0c2afe2a
Author: Anjay <kumaranjay8@gmail.com>
Date: Wed Sep 18 11:08:59 2019 +0530

    ps2 initial Progress

commit 0afdd64bafd9391e1b32f2907f3ddd98ba7c9af0
Author: Anjay <kumaranjay8@gmail.com>
Date: Wed Sep 18 10:28:04 2019 +0530

    ps1 completed

commit 0cc284edc449b4b77bc9f4d38ff39a584f57ead2
Author: Anjay <kumaranjay8@gmail.com>
Date: Wed Sep 18 09:42:34 2019 +0530

    Half Hour
```

Git Url is :https://github.com/2019JTM2169/Assignment_8/commits/master

4 Appendix

4.1 Code for ps1

```
1 #####Problem Statement-1#####
2 print("Welcome to Data Generator\n")
3 a='*'
4 print(a*20)
5 print("\nEnter The Binary Data to be Trasnmitted")
6 i_data=input("Enter Here:")      # Taking Input from the user
7 count=0
8 for items in i_data:              # Counting Number of one's
9     if items=='1':
10         count+=1
11
12 if count%2==0:                    # Checking If number of ones is even or odd
13     i_data1=i_data+'1'
14
15 else:
16     i_data1=i_data+'0'
17
18 print("Parity bit data:",i_data1) # Printing The parity bit data
19 o_data=[]
20 j=0
21 k=0
22 o_data1=""
23 for items in i_data1:              #bit Stuffing Logic Starts Here
24     o_data1=o_data1+items
25
26
27
28
29 o_data3=o_data1[j:]
30
31 if '010' in o_data3:
32
33     o_data1=o_data1+'0'
34
35 j=k
36 j+=2
37 k+=1
38
39 k+=1
40
41 o_datafinal=o_data1+'0101'        #Appending flag for end of the frame
42 print("Transmitting Data is:",o_datafinal)    #printing The final data
```

4.2 Code for ps2

```
1 #####Problem Statement-2#####
2 def represent(list1):
3     k=1
4     str1="".join(lst)
5     str2=str1[1:4]
6     str3=str1[4:7]
7     str4=str1[7:]
8     print(str2)
9     print(str3)
10    print(str4)
11
12
13
14    a='_'
15    print(a*30)                                #Showing the Welcome
        Message and Rules
16    print("Welcome to the Game")
17    print(a*30)
18    print("Rules of the Game:")
19    print("Player 1 will use only odd numbers[1-9]")
20    print("Player 2 will use only even numbers[1-9]")
21    lsts=[0]*20
22
23    lst=['0']*10                                # Defining an
        empty list
24    i=0
25    while i<10:
26        lst.append("")
27        i+=1                                    #Inserting 9 empty
        elements in the list
28    i=1
29    k=0
30    d=[0]*200
31    index1=0
32    f=0
33    while i<11:
34        print("Current State Of Matrix")
35        represent(lst)
36    while True:                                # Taking the input from
        Player 1 and checking the boundary conditions
37        pn1=int(input("Player 1 Enter the Position in which number is to be Entered:"))
38        if pn1>0 and pn1<10:
39            break
40        else:
41            print("Incorrect Input")
42        while True:
43            pp1=int(input("Player 1 Enter the number which is to be Entered:"))
```

```

44 if pp1>0 and pp1<10 and pp1%2!=0:
45     break
46 else:
47     print("Incorrect Input")
48     lst[pn1] = str(pp1)
49     print("Current State Of Matrix")
50     represent(lst)
51     while True:                                     # Taking the input from
        player 2 and checking the boundary conditions
52     pn2=int(input("Player 2 Enter the Position in which number is to be Entered:")
        )
53     if pn2>0 and pn2<10:
54         break
55     else:
56         print("Incorrect Input")
57     while True:
58     pp2=int(input("Player 2 Enter the number which is to be Entered:"))
59     if pp2>0 and pp2<10 and pp2%2==0:
60         break
61     else:
62         print("Incorrect Input")
63
64     lst[pn2]=str(pp2)
65     print("Current State Of Matrix")
66     represent(lst)
67     lsts[pn1]=pp1
68     lsts[pn2]=pp2
69     #print(lsts)
70     #-----
71     #Checking the sum
72     for count in range(1,4):
73         d[index1]+=lsts[count]
74         index1+=1
75     #print("index1",index1)
76     for count in range(4,7):
77         d[index1]+=lsts[count]
78     #print("hi")
79     index1+=1
80     for count in range(7,10):
81         d[index1]+= lsts[count]
82     #print("hi")
83     index1+=1
84     for count in range(1, 8, 3):
85         d[index1]+= lsts[count]
86     #print("hi")
87     index1+=1
88     for count in range(2, 9, 3):
89         #print(count)
90         d[index1]+= lsts[count]
91         #print("hi")

```

```

92 index1+=1
93 for count in range(3, 10, 3):
94     d[index1]+= lsts[count]
95     index1+=1
96 for count in range(1, 10, 4):
97     d[index1]+= lsts[count]
98     #print("hi")
99     index1+=1
100 for count in range(3, 8, 2):
101     d[index1]+= lsts[count]
102     #print("hi")
103     index1+=1
104 for items in d:
105     if items>14 and items%2==0:
106         f=1
107         break
108     elif items>14 and items%2!=0:
109         f=2
110         break
111     if f==1 or f==2:
112         break
113     i+=2
114     #print("i is ",i)
115     #k+=1
116     # Checking the winner
117     if f==1:
118         print("Player 2 is the winner")
119     elif f==2:
120         print("PLayer 1 is the winner")
121     elif f==0:
122         print("Match Tied")

```


References

- [1] Geeks For Geeks. *String Formatting*. <https://www.geeksforgeeks.org/python-strings/>.
- [2] Geeks For Geeks. *Tic Tac Toe*. <https://www.geeksforgeeks.org/python-implementation-automatic-tic-tac-toe-game-using-random-number/>.
- [3] Python Guru. *Range Function*. PythonGuru<https://thepythonguru.com/python-builtin-functions/range/>.
- [4] Stack Overflow. *Bit Stuffing*. <https://stackoverflow.com/questions/44115364/byte-stuffing-unstuffing-in-python>.