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

List of Figures

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 implementration was hard

1.6 Program Structure

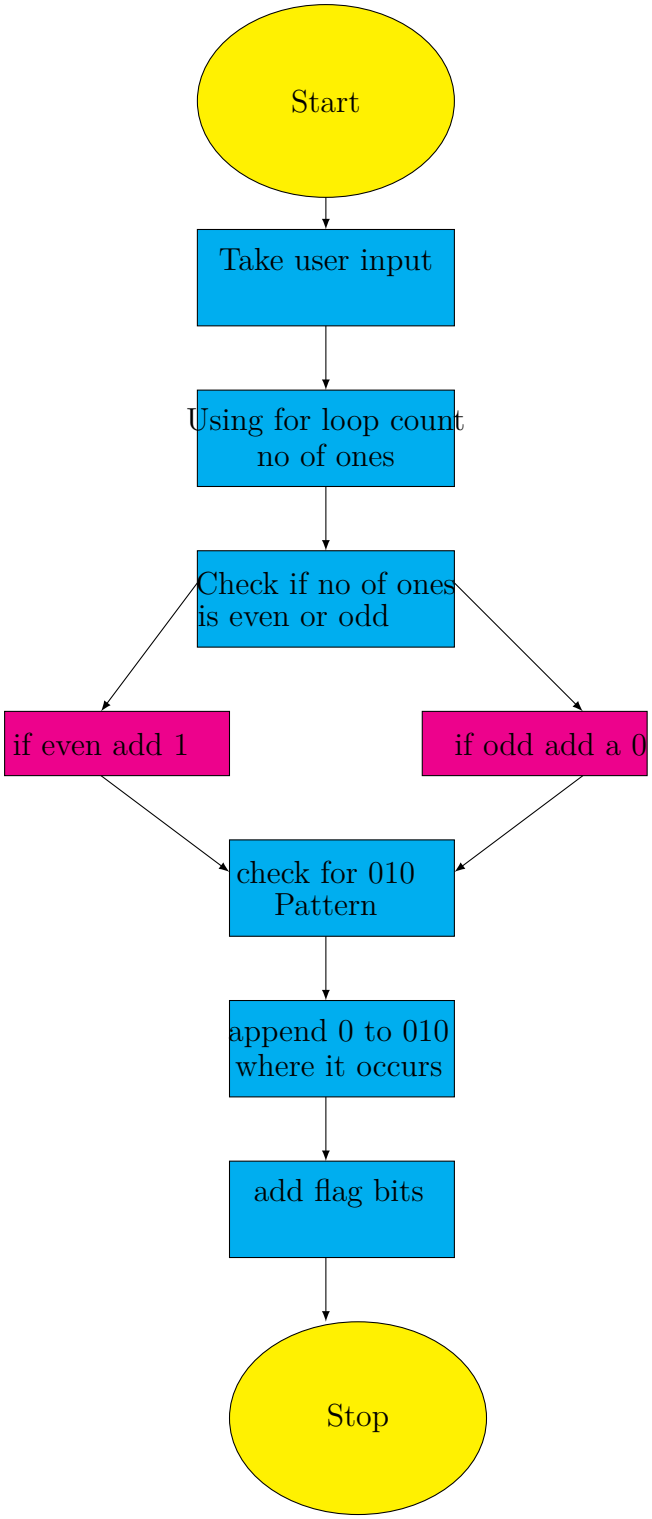


Figure 1: Flow chart-1

1.7 Screenshots

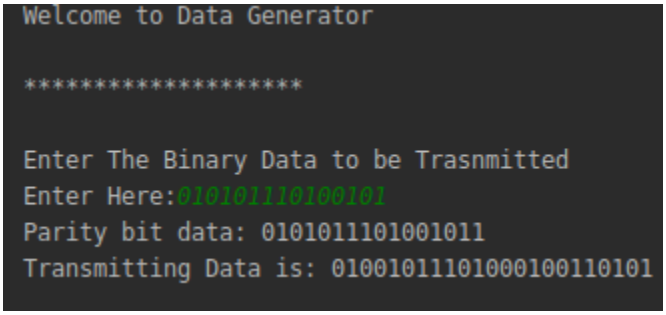


Figure 2: Screenshot1

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

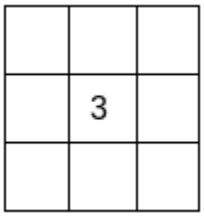
- $1 <= Position <= 9$
- $1 <= Number <= 9$

Terminal

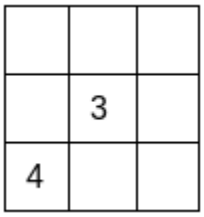
- Print 'Welcome to the Game!'.
- Print whether it is Player 1's or Player 2's chance.
- Get the position and number to be entered from the user.
- Show tic tac toe with data.
- Continue till the game gets draw or some player wins and show the result.
- Ask the user whether to continue for the next game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

2.2 Sample Output:

Welcome to the Game!
Player 1's chance
Enter the position and number to be entered: 5,3



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



... 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

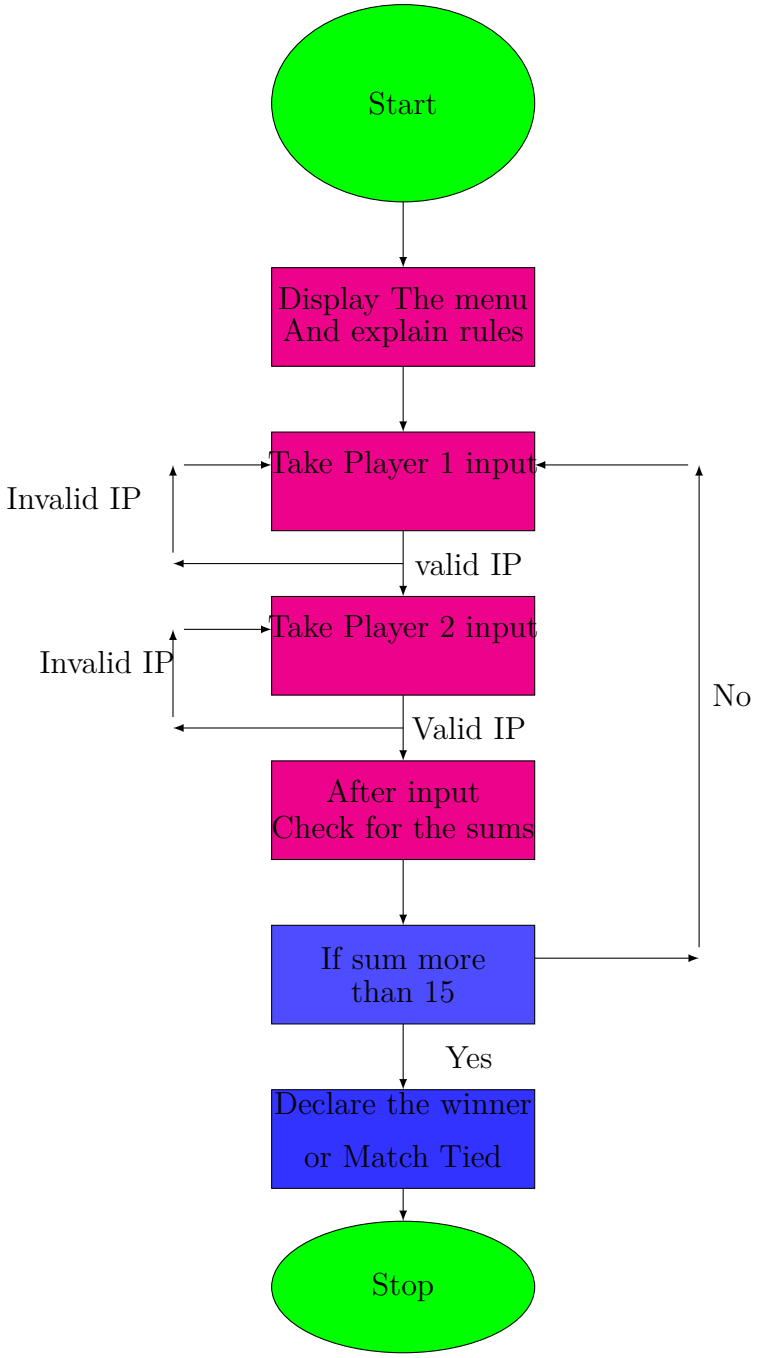


Figure 3: Flowchart-2

2.7 Screenshots

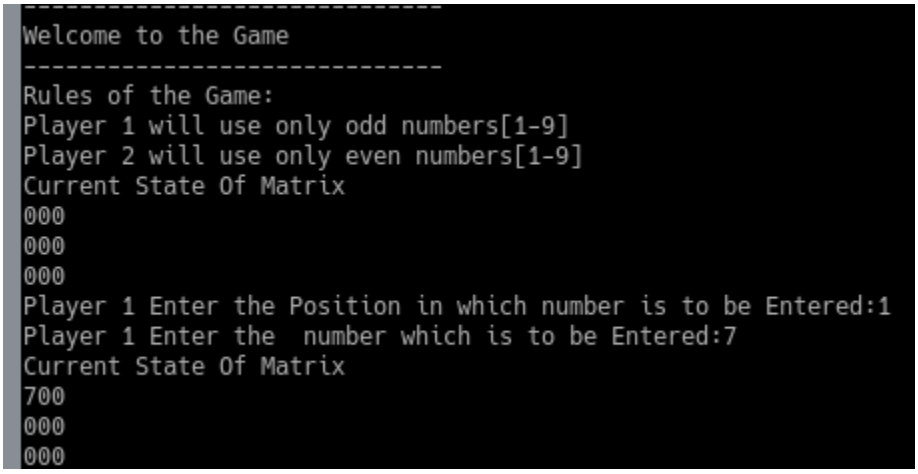


Figure 4: Screenshot2

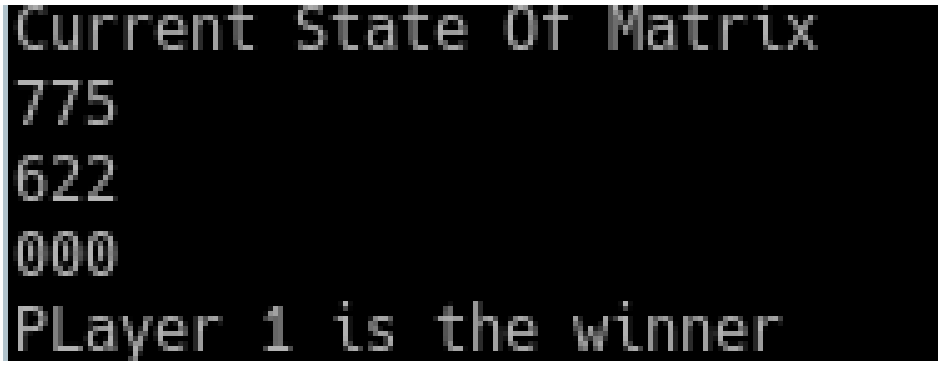


Figure 5: Screenshot3

3 Git

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

4 Appendix

4.1 Code for ps1

```
#######Problem Statement-1###########
print("Welcome to Data Generator\n")
a='*'
print(a*20)
print("\nEnter The Binary Data to be Trasnmitted")
i_data=input("Enter Here:")      # Taking Input from the user
count=0
for items in i_data:             # Counting Number of one's
if items=='1':
count+=1

if count%2==0:                   # Checking If number of ones is even or odd
i_data1=i_data+'1'

else:
i_data1=i_data+'0'

print("Parity bit data:",i_data1)  # Printing The parity bit data
o_data=[]
j=0
k=0
o_data1=""
for items in i_data1:                      #bit Stuffing Logic Starts Here
o_data1=o_data1+items




o_data3=o_data1[j:]

if '010' in o_data3:

o_data1=o_data1+'0'

j=k
j+=2
k+=1

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

```
#####Problem Statement-2########
def represent(list1):
k=1
str1="".join(lst)
str2=str1[1:4]
str3=str1[4:7]
str4=str1[7:]
print(str2)
print(str3)
print(str4)



a='-'
print(a*30)                                          #Showing the Welcome
    Message and Rules
print("Welcome to the Game")
print(a*30)
print("Rules of the Game:")
print("Player 1 will use only odd numbers[1-9]")
print("Player 2 will use only even numbers[1-9]")
lsts=[0]*20

lst=['0']*10                                              # Defining an
    empty list
i=0
while i<10:
lst.append("")
i+=1                                                 #Inserting 9 empty
    elements in the list
i=1
k=0
d=[0]*200
index1=0
f=0
while i<11:
print("Current State Of Matrix")
represent(lst)
while True:                                          # Taking the input from
    Player 1 and checking the boundary conditions
pn1=int(input("Player 1 Enter the Position in which number is to be Entered:")
    )
if pn1>0 and pn1<10:
break
else:
print("Incorrect Input")
while True:
pp1=int(input("Player 1 Enter the  number which is to be Entered:"))
```

```
if pp1>0 and pp1<10 and pp1%2!=0:
break
else:
print("Incorrect Input")
lst[pn1] = str(pp1)
print("Current State Of Matrix")
represent(lst)
while True:                                          # Taking the input from
    player 2 and checking the boundary conditions
pn2=int(input("Player 2 Enter the Position in which number is to be Entered:")
    )
if pn2>0 and pn2<10:
break
else:
print("Incorrect Input")
while True:
pp2=int(input("Player 2 Enter the  number which is to be Entered:"))
if pp2>0 and pp2<10 and pp2%2==0:
break
else:
print("Incorrect Input")

lst[pn2]=str(pp2)
print("Current State Of Matrix")
represent(lst)
lsts[pn1]=pp1
lsts[pn2]=pp2
#print(lsts)
#-------------------------------------------------------------------
#Checking the sum
for count in range(1,4):
d[index1]+=lsts[count]
index1+=1
#print("index1",index1)
for count in range(4,7):
d[index1]+=lsts[count]
#print("hi")
index1+=1
for count in range(7,10):
d[index1]+= lsts[count]
#print("hi")
index1+=1
for count in range(1, 8, 3):
d[index1]+= lsts[count]
#print("hi")
index1+=1
for count in range(2, 9, 3):
#print(count)
d[index1]+= lsts[count]
#print("hi")
```

```
index1+=1
for count in range(3, 10, 3):
d[index1]+= lsts[count]
index1+=1
for count in range(1, 10, 4):
d[index1]+= lsts[count]
#print("hi")
index1+=1
for count in range(3, 8, 2):
d[index1]+= lsts[count]
#print("hi")
index1+=1
for items in d:
if items>14 and items%2==0:
f=1
break
elif items>14 and items%2!=0:
f=2
break
if f==1 or f==2:
break
i+=2
#print("i is",i)
#k+=1
# Checking the winner
if f==1:
print("Player 2 is the winner")
elif f==2:
print("PLayer 1 is the winner")
elif f==0:
print("Match Tied")
```

References

[1] Geeks For Geeks. *Sring 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*. PythonGuruhttps://thepythonguru.com/python-builtin-functions/range/.

[4] Stack Overflow. *Bit Stuffing*. https://stackoverflow.com/questions/44115364/byte-stuffing-unstuffing-in-python.