Assignment-8
ELP-718 Telecom Software Lab

Ayush Gupta
2019JTM2673
2019-20

11 sept,2019

A report presented for the assignment on
Python and Github.

Bharti School Of
Telecommunication Technology and Management
IIT DELHI

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 1s 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.

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

1.2 Assumptions

1. Using linux environment through Ubuntu.
2. Using Latex Libraries and Packages.
3. Using Pycharm Ide and gnu make for python code compilation and execution.

1.3 Algorithm and Implementation

1. Create ps1.py
2. Take input from user in binary form.
3. Convert each character of vstring as list element.
4. Check for parity even and odd.
5. Append the parity bit.
6. Check for 010 and stuff the bit and append 0101 at the last.

1.4 Inputs and Outputs

.

Sample input
010101110100101

Sample Output :
Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101

1.5 Program Structure

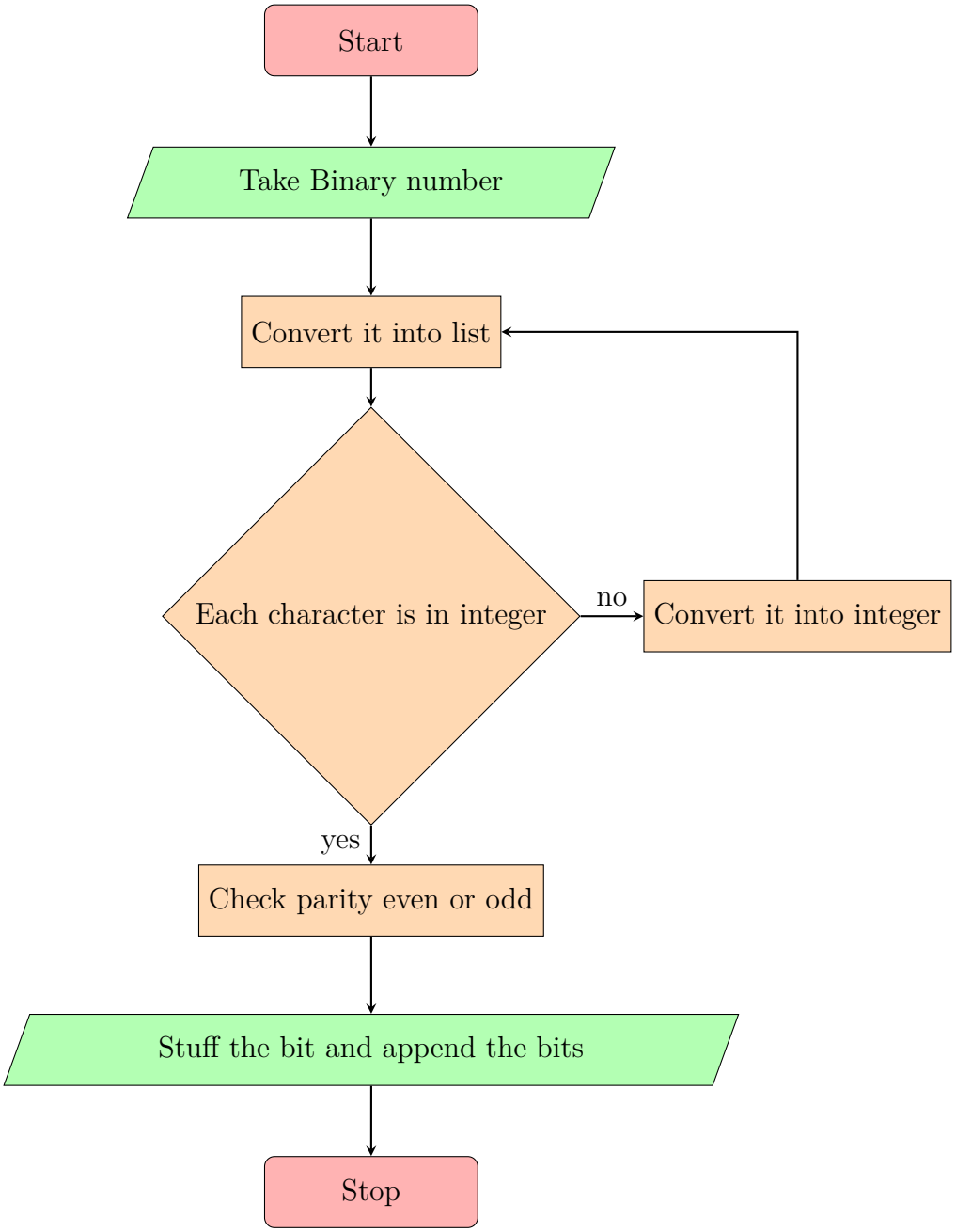


Figure 1: Problem 1 Flowchart

1.6 Screenshots

Figure 2: Output Of Parity and Transmitting data

2 Problem Statement-2

3X3 Numeric Tic-Tac-Toe (Use numbers 1 to 9 instead of XâĂŹs and OâĂŹs) 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 âĂŞ Line can be horizontal, vertical or diagonal

Constraints:

1. $1 <= Position <= 9$
2. $1 <= Number <= 9$

Terminal:

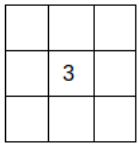
1. Print Welcome to the Game!.
2. Print whether it is Player 1âĂŹs or Player 2âĂŹs chance.
3. Get the position and number to be entered from the user.
4. Show tic tac toe with data.
5. Continue till the game gets draw or some player wins and show the result.
6. Ask the user whether to continue for the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

2.1 Assumptions

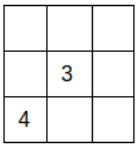
1. Using linux environment through Ubuntu.
2. Using Latex Libraries and Packages.
3. Using Pycharm Ide and gnu make for python code compilation and execution.

2.2 Inputs and Outputs

. **Sample Output format:**
Welcome to the Game!
Player 1 chance
Enter the position and number to be entered: 5,3



Player 2 chance
Continue till game ends
Enter the position and number to be entered: 7,4



2.3 Algorithm and Implementation

1. Create ps2.py
2. Create tictac board print function.
3. Take each player position and number.
4. Check if sum is 15 of any row column or
5. If sum is 15 winner decided.

2.4 Program Structure

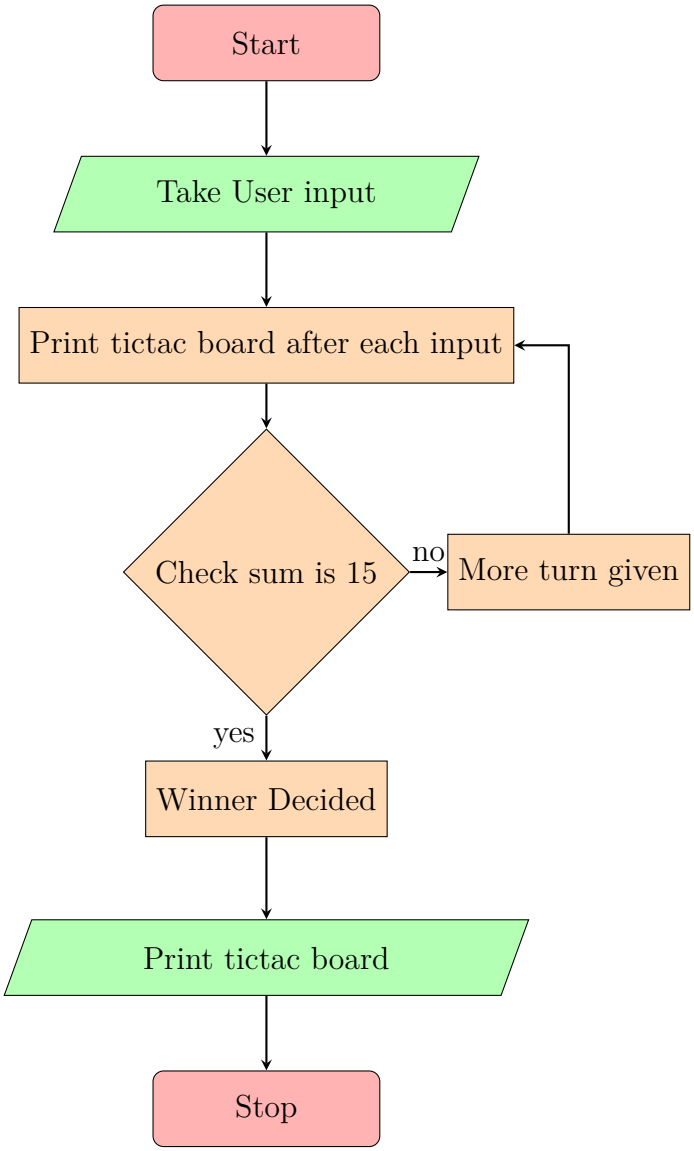


Figure 3: Problem 2 Flowchart

2.5 Screenshots

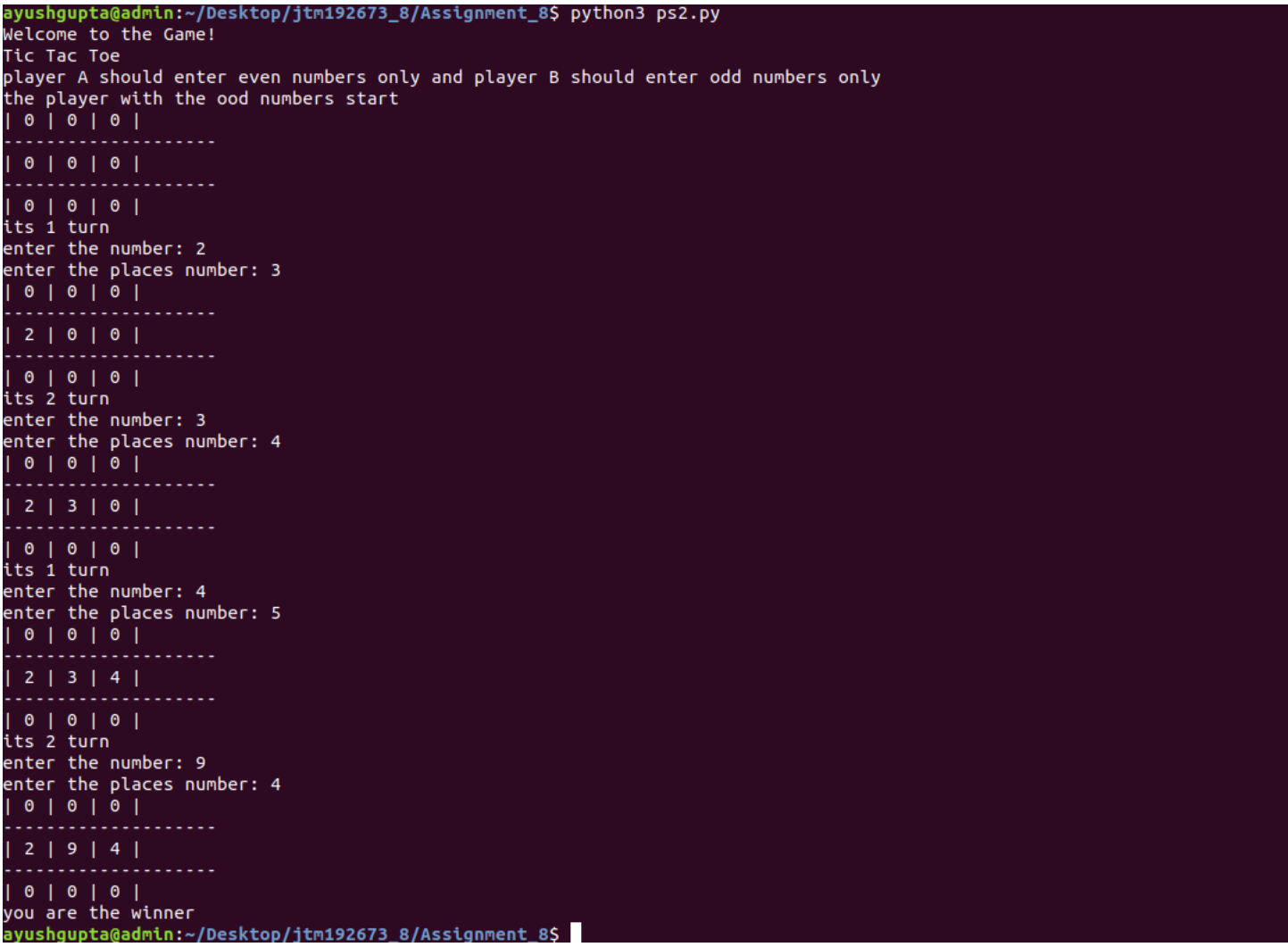


Figure 4: Output

3 Appendix

3.1 Code for ps1

```
1
2
3
4
5 #.....Problem 1.....
6 print("Enter binary bit dat that has to be transmitted: ",end=' ')
7 s=input()
8 count=0
9 for i in s:        #...loop to count number of 1 in string...
10 #print(i)
11 if i=='1':
12 count=count+1
13 #..condition for even parity check....
14 if count%2==0:
15 print("This is even parity" )
16 p=s[:len(s)]+'1'
17 print("Parity bit data:", p)  #...for even parity....
18 #print (p)
19 #print (type(p))
20 q= list(p)
21 #print(q)
22 for k in range(0, len(q)):
23 q[k] = int(q[k])
24
25 #print (q)
26
27 #for r in q:
28 #   print(r, end="")
29
30 #print (type(q))
31 #print("Parity bit data:", p)  #...for even parity....
32 #'010' in p
33 #td=[ch for ch in q]
34 #td = int(td)
35 #print(td)
36 #...code for transmitted data...
37 for j in range(len(q)):
38 if q[j]==0 and q[j+1]==1 and q[j+2]==0:
39
40 q.insert(j+3, 0)
41 print(q)
42 q.append(0)
43 q.append(1)
44 q.append(0)
45 q.append(1)
```

```
#print (q)
#s= " "
#r= s.join(q)
#print(r)
print("Transmitted Data:")
for l in q:
#print("Transmitting Data:", l)
print(l, end='')



#.....for odd parity data...
else:
print("This is Odd parity" )
p=s[:len(s)]+'0'
print("Parity bit data:", p)  #...for even parity....
#print (p)
#print (type(p))
q= list(p)
#print(q)
for k in range(0, len(q)):
q[k] = int(q[k])

print (q)

#for r in q:
#   print(r, end="")

#print (type(q))
#print("Parity bit data:", p)  #...for even parity....
#'010' in p
#td=[ch for ch in q]
#td = int(td)
#print(td)
#....code for tansmitted data...
for j in range(len(q)):
if q[j]==0 and q[j+1]==1 and q[j+2]==0:

q.insert(j+3, 0)
print(q)
q.append(0)
q.append(1)
q.append(0)
q.append(1)
print (q)
#s= " "
#r= s.join(q)
#print(r)
print("Transmitted Data:")
```

```
96 for l in q:
97 #print("Transmitting Data:", l)
98 print(l, end='')
```

3.2 Code for ps2

```
1
2
3
4 #.......Problem2.........
5 print("Welcome to the Game!")
6 board = [0, 0, 0,
7 0, 0, 0,
8 0, 0, 0]
9 player = '1' #with this we'll know which player's turn it is
10
11 def tic_tac_toe ():
12 print ('|' ,board[0],'|',board[1] ,'|', board[2],'|')
13 print ('-----------------------')
14 print ('|' ,board[3],'|',board[4] ,'|', board[5],'|')
15 print ('-----------------------')
16 print ('|' ,board[6],'|',board[7] ,'|', board[8],'|')
17
18 def move(x1,x2):
19 board[x2] = x1
20 tic_tac_toe()
21
22 def odd (x, x2):
23 while  (x%2==0):
24 x = int(input ('enter an odd number'))
25 #Nothing here because if we get out of the while is because it's a valid
      number (we're not checking numbers out of range or anything)
26 move (x ,x2)
27
28 def even (x ,x2) :
29 while  (x%2!=0):
30 x = int(input ('enter an even number'))
31 #Same here
32 move (x ,x2)
33
34 def winner ():
35 if (board[0]+board [1]+board[2]==15 or
36 board[0]+board [3]+board[6]==15 or
37 board[1]+board [4]+board[7]==15 or
38 board[3]+board [4]+board[5]==15 or
39 board[2]+board [5]+board[8]==15 or
40 board[6]+board [7]+board[8]==15):
41 print ('you are the winner')
42 return True #To know if we need to stop the game
43 else: return False
44
45 def turn(s):
46 print ('its '+ s +' turn')
47 x = int (input ('enter the number: '))
```

```
x1 = int (input ('enter the places number: '))
if player == '1':
even(x, x1)
else: odd(x, x1)
https://github.com/RkJapper/Assignment\_8
print('Tic Tac Toe')
print ('player A should enter even numbers only'+' and player B should enter
    odd numbers only')
print ('the player with the ood numbers start')
tic_tac_toe ()
while (True):
turn(player)
if winner(): break
else:
if player == '1': player = '2'
else: player = '1'
```

3.3 GITHUB LINK

https://github.com/2019JTM2673/Assignment_8.git

References