Assignment-7

ELP - 718 Telecommunication Software Laboratory

Ekta Singh

2019JTM2086

2019-2021

A report presented for the assignment on

Python and Github

Bharti School Of

Telecommunication Technology and Management

IIT Delhi

India

Sep 18, 2019

Contents

List of Figures

1 Problem Statement-1

1.1 Problem Statement

1. **Parity Check**
2. 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 1âĂŹs in the bit string is even and has the value 0 otherwise, i.e., Odd Parity Check.
3. **Bit-Oriented Framing**
4. 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 Input-Output

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

1.3 Assumptions

1. We have taken a variable binary which store the entered binary numberÂă by the user
2. Newbinary variable store the binary data after inserting 0 wherever 010 was present
3. Many more variables were initialized for comparing counting etc

1.4 Algorithm and Implementation

1. Start
2. First take the binary data as a input

3. Now check whether the input data is validÂăor not and display the message accordinglyÂă
4. Now using count ,count the number of 1Âă
5. If numberÂă of 1 is even than display 1 for even parityÂă
6. If number of 1 is odd than displsy 0 as odd parity
7. Now check the binary data and insert a 0 wherever 010 is present i.e 0100
8. The string 0101 is used as the bit string or flag to indicate the end of the frame.
9. Now ,display the output
10. end

1.5 Diffculties/issues faced

1. It was not as much problematic but some where identation problem cause problem while runningÂă the code in Gedit

1.6 Program Structure

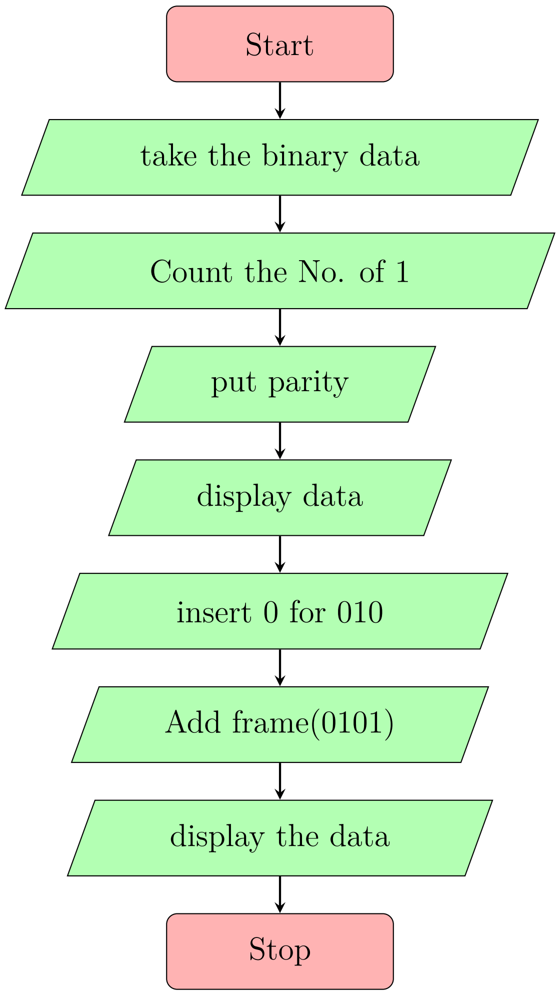
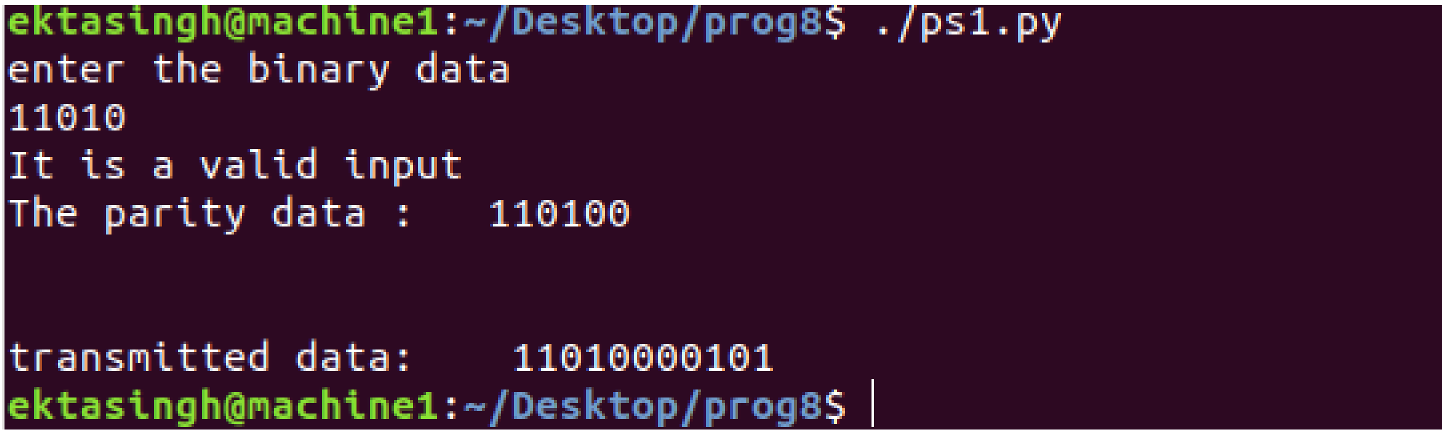


Figure 1: Problem 1 Flowchart

1.7 Screenshots



```
ektasingh@machine1:~/Desktop/prog8$ ./ps1.py
enter the binary data
11010
It is a valid input
The parity data :   110100


transmitted data:    11010000101
ektasingh@machine1:~/Desktop/prog8$ 
```

Figure 2: Screenshot1

2 Problem Statement-2

2.1 Problem Statement

3X3 Numeric Tic-Tac-Toe (Use numbers 1 to 9 instead of Xs and Os) 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

2.2 Assumptions

- Block was used where player can show their move
- Blocklog to store the occupied positions
- Player variable was taken toh provide the turn of a player
- Many functions was defined for the player move , turn , even ,odd etc

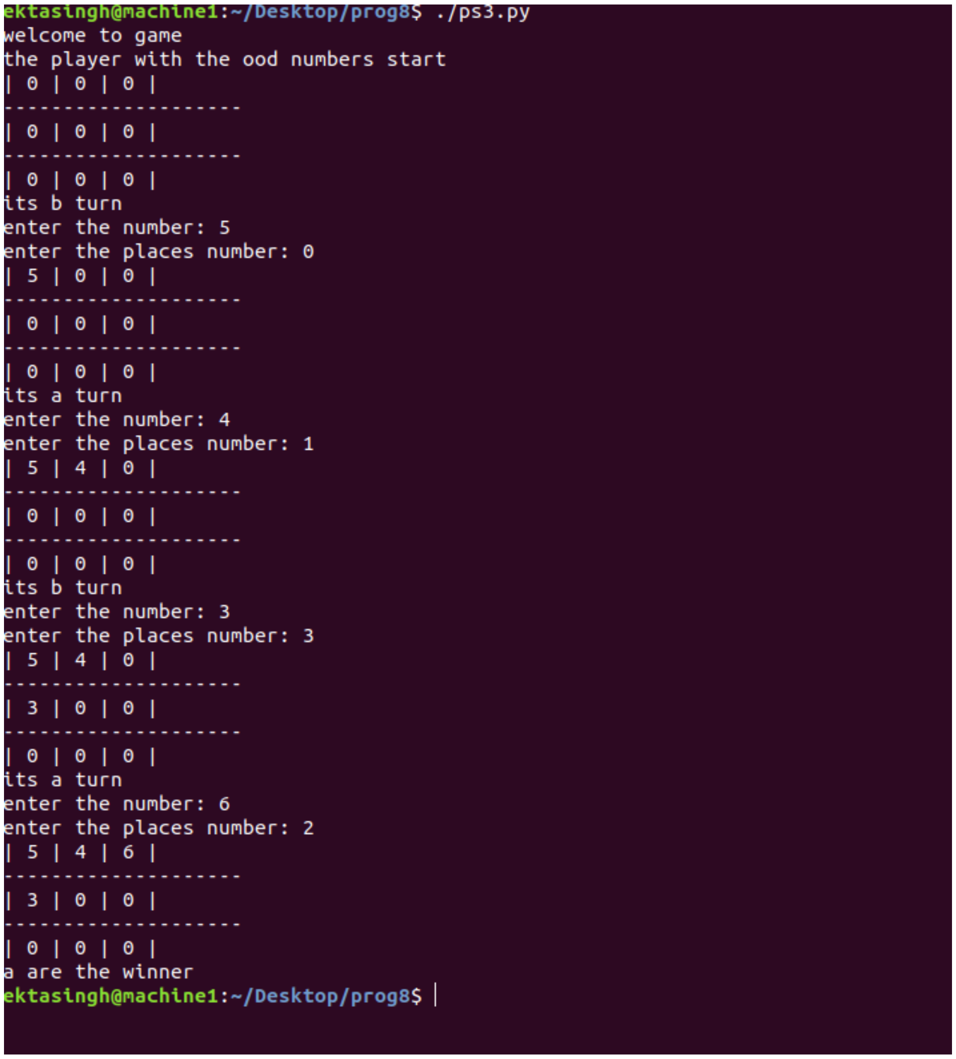
2.3 Algorithm and Implementation

1. Start
2. Firstly, i used two block i.e block of 3x3 order and block log of 3x3 order
3. Block take the input given by ever player while blocklog make every position one showing as it is occupied
4. Check whether input is in a range of 1 to 9 else show invalid
5. Now check all the condition staight or diagonally if in any case sum becomes 15 then break the game and display the last player as a winner
6. EndÂă

2.4 Diffculties/issues faced

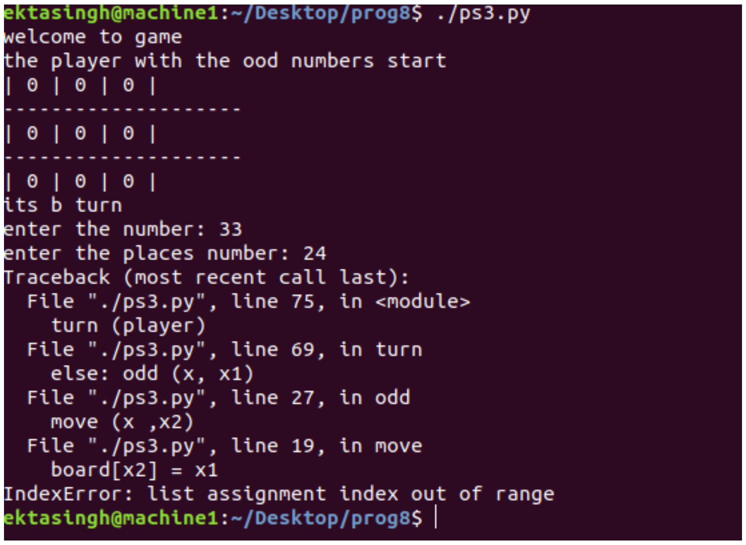
- I faced problem in switching the player and taking the data in oocupied position which was later on resolved
- Secondly, same identation was a big problem in Gedit

2.5 Screenshots



```
ektasingh@machine1:~/Desktop/prog8$ ./ps3.py
welcome to game
the player with the ood numbers start
| 0 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
its b turn
enter the number: 5
enter the places number: 0
| 5 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
its a turn
enter the number: 4
enter the places number: 1
| 5 | 4 | 0 |
--------------------
| 0 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
its b turn
enter the number: 3
enter the places number: 3
| 5 | 4 | 0 |
--------------------
| 3 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
its a turn
enter the number: 6
enter the places number: 2
| 5 | 4 | 6 |
--------------------
| 3 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
a are the winner
ektasingh@machine1:~/Desktop/prog8$
```

Figure 3: Screenshot2



```
ektasingh@machine1:~/Desktop/prog8$ ./ps3.py
welcome to game
the player with the ood numbers start
| 0 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
--------------------
| 0 | 0 | 0 |
its b turn
enter the number: 33
enter the places number: 24
Traceback (most recent call last):
  File "./ps3.py", line 75, in <module>
    turn (player)
  File "./ps3.py", line 69, in turn
    else: odd (x, x1)
  File "./ps3.py", line 27, in odd
    move (x ,x2)
  File "./ps3.py", line 19, in move
    board[x2] = x1
IndexError: list assignment index out of range
ektasingh@machine1:~/Desktop/prog8$ 
```

Figure 4: Screenshot3

2.6 Program Structure

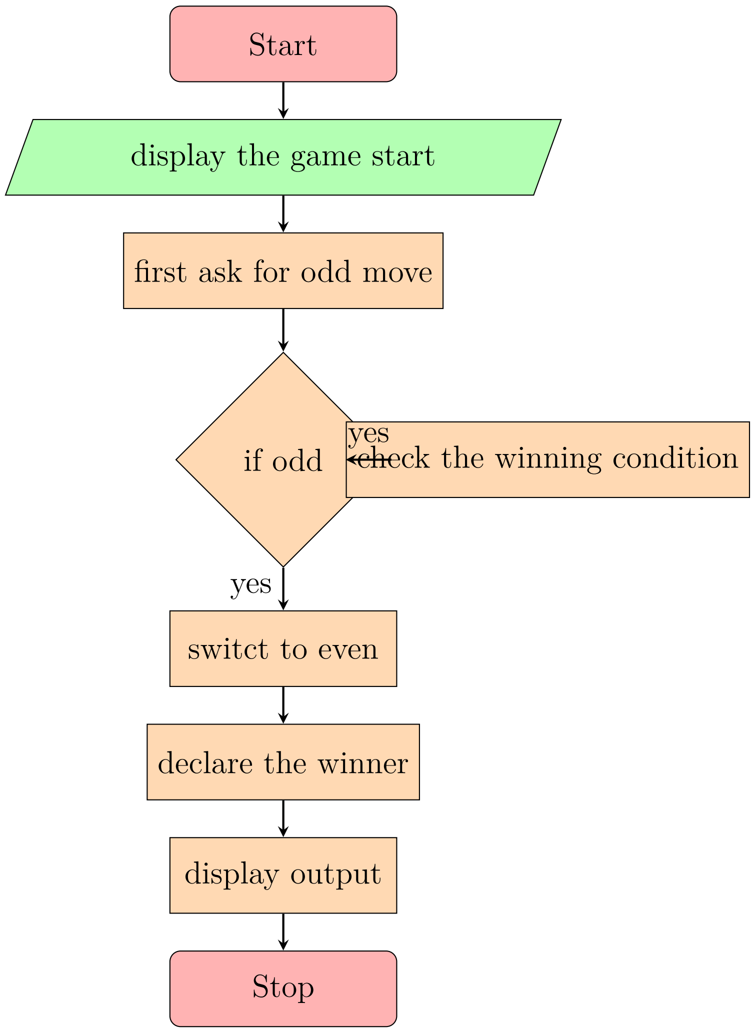


Figure 5: Problem 1 Flowchart

3 Appendix

3.1 Code for ps1

```
#!/usr/bin/python3

def check (input):
  p=set (input)
  data={'0','1'}
  if data==p or data=={'0'} or data=={'1'}:
   print("It is a valid input")


binary=input("enter the binary data\n")      #taking input
check (binary)
one='1'
count=binary.count(one)
#print("count is:",count)
if count%2==0:                                        #adddding parity bit
  binary=binary+'1'
  print("The parity data :  ",binary)      #data with parity
else:
  binary=binary+'0'
  print("The parity data :  ",binary)
print("\n")
newbinary=binary.replace("010","0100")
newbinary= newbinary+"0101"                          #for insertion of 0
print("transmitted data:   ",newbinary)
```

3.2 Code for ps2

```python
#!/usr/bin/python3
board = [0,0,0,
     0,0,0,
     0,0,0]
boardLog = [0, 0, 0,
        0, 0, 0,
        0, 0, 0]

player = 'b' #with this we'll know which player's turn it is

def tic_tac_toe ():
    print ('|' ,board[0],'|',board[1] ,'|', board[2],'|')
    print ('---------------------')
    print ('|' ,board[3],'|',board[4] ,'|', board[5],'|')
    print ('---------------------')
    print ('|' ,board[6],'|',board[7] ,'|', board[8],'|')

def move(x1,x2):
    board[x2] = x16
    boardLog[x2] = 1
    tic_tac_toe()

def odd (x, x2):
    while  (x%2==0):
        x = int(input ('enter an odd number'))
    #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)
    move (x ,x2)

def even (x ,x2) :
    while  (x%2!=0):
        x = int(input ('enter an even number'))
    #Same here
    move (x ,x2)

def winner():
    if (boardLog[0] + boardLog[1] + boardLog[2] == 3):       #checking the condition of winning
      if (board[0]+board [1]+board[2]==15):
          print ('{0} are the winner' .format(player))
          return True
    if (boardLog[0] + boardLog[3] + boardLog[6] == 3):
      if (board[0]+board [3]+board[6]==15):
          print ('{0} are the winner' .format(player))
          return True
    if (boardLog[1] + boardLog[4] + boardLog[7] == 3):
      if (board[1]+board [4]+board[7]==15):
          print ('{0} are the winner' .format(player))
```

```
           return True
    if (boardLog[3] + boardLog[4] + boardLog[5] == 3):
      if (board[3]+board [4]+board[5]==15):
          print ('{0} are the winner' .format(player))
          return True
    if (boardLog[2] + boardLog[5] + boardLog[8] == 3):
      if (board[2]+board [5]+board[8]==15):
          print ('{0} are the winner' .format(player))
          return True
    if (boardLog[6] + boardLog[7] + boardLog[8] == 3):
      if (board[6]+board [7]+board[8]==15):
           print ('{0} are the winner' .format(player))
           return True

    else: return False

def turn (s):
    print ('its '+ s +' turn')
    x = int (input ('enter the number: '))                #entering the no. of players
    x1 = int (input ('enter the places number: '))                  #entering postion
    if player == 'a':
        even (x, x1)
    else: odd (x, x1)

print( 'welcome to game')
print ('the player with the ood numbers start')
tic_tac_toe ()
while (True):
    turn (player)
    if winner(): break
    else:
        if player == 'a': player = 'b'              #switching the chances of players
        else: player = 'a'
```

References

[1] BitBucket. *Learn Git*. `https://www.atlassian.com/git/tutorials`.

[2] GeeksForGeeks. *List methods in Python*. `https://www.geeksforgeeks.org/list-methods-python/`.

[3] Python Software Foundation. *Python 3.7.4 documentation*. `https://docs.python.org/3/`.

[4] Telusko. *Python Programming Tutorial for Beginners*. `https://www.youtube.com/watch?v=QXeEoD0pB3E&list=PLsyeobzWxl7poL9JTVyndKe62ieoN-MZ3`.

[5] w3schools. *Python Dictionaries*. `https://www.w3schools.com/python/python_dictionaries.asp`.