

## Assignment 8

**ELP-718 Telecommunication Software Laboratory**

**Shilpi Mishra**

**2019JTM2677**

**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

<b>1</b>	<b>Problem Statement-1</b>	<b>3</b>
1.1	Problem Statement . . . . .	3
1.2	Assumptions . . . . .	3
1.3	Algorithm and Implementation . . . . .	3
1.4	Program Structure . . . . .	4
1.5	Screenshots . . . . .	5
<b>2</b>	<b>Problem Statement-2</b>	<b>6</b>
2.1	Problem Statement . . . . .	6
2.2	Input Format . . . . .	6
2.3	Output Format . . . . .	6
2.4	Algorithm and Implementation . . . . .	7
2.5	Program Structure . . . . .	8
2.6	Screenshots . . . . .	9
<b>3</b>	<b>Appendix</b>	<b>10</b>
3.1	Code for ps1 . . . . .	10
3.2	Code for ps2 . . . . .	12

## List of Figures

1	Screenshot1 . . . . .	5
2	Screenshot5 . . . . .	9

# 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

- **Sample Input**

010101110100101

- **Sample Output**

Parity bit data : 0101011101001011 Transmitting data: 01001011101000100110101

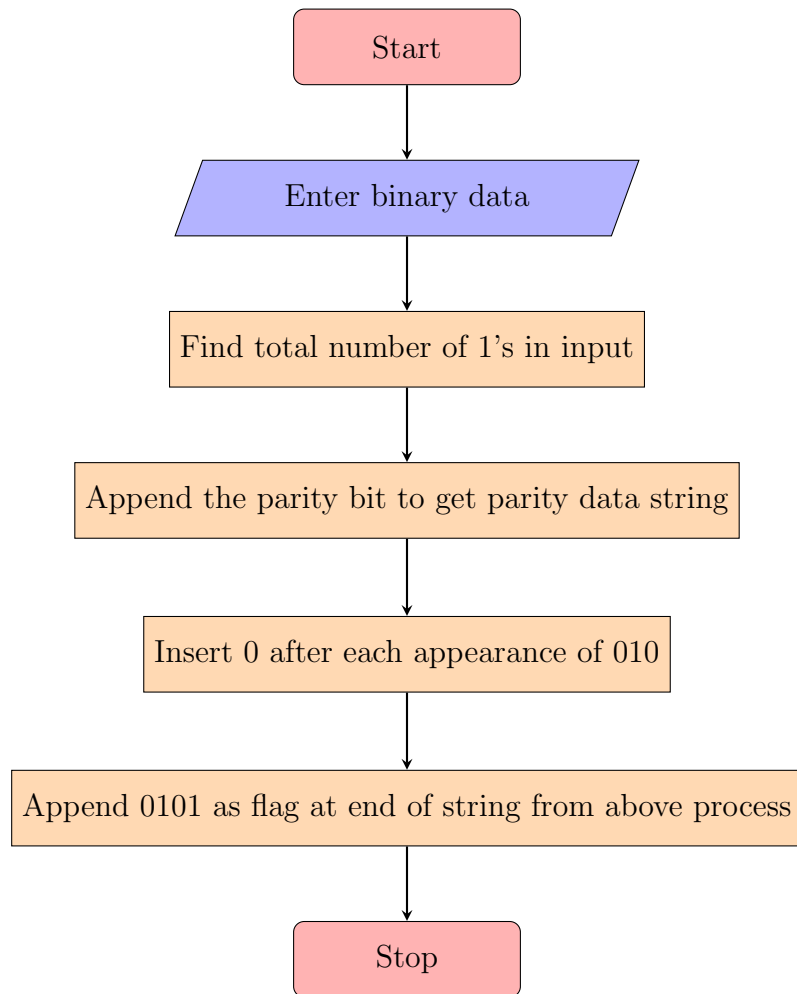
## 1.2 Assumptions

The data to be entered is binary data.

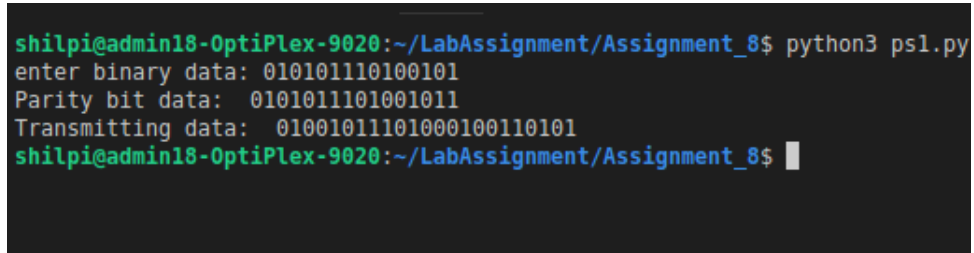
## 1.3 Algorithm and Implementation

- Take binary data as input string.
- Determine parity bit of the input data by counting numbers of 1 in data.
- Append parity bit to get parity data after odd parity check
- Insert a 0 after each appearance of 010 in the parity string .
- Append sub string 0101 as flag to indicate the end of the frame.

## 1.4 Program Structure



## 1.5 Screenshots

A terminal window with a dark background and light green text. The prompt is 'shilpi@admin18-OptiPlex-9020:~/LabAssignment/Assignment\_8\$'. The user enters 'python3 ps1.py'. The script outputs 'enter binary data: 010101110100101', 'Parity bit data: 0101011101001011', and 'Transmitting data: 01001011101000100110101'. The prompt returns to 'shilpi@admin18-OptiPlex-9020:~/LabAssignment/Assignment\_8\$' with a cursor.

```
shilpi@admin18-OptiPlex-9020:~/LabAssignment/Assignment_8$ python3 ps1.py
enter binary data: 010101110100101
Parity bit data: 0101011101001011
Transmitting data: 01001011101000100110101
shilpi@admin18-OptiPlex-9020:~/LabAssignment/Assignment_8$
```

Figure 1: 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 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 1s or Player 2s chance.  
Get the position and number to be entered from the user.  
Player 2s chance Enter the position and number to be entered: 7,4 .  
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 Input Format

Enter position and number for the game

### 2.3 Output Format

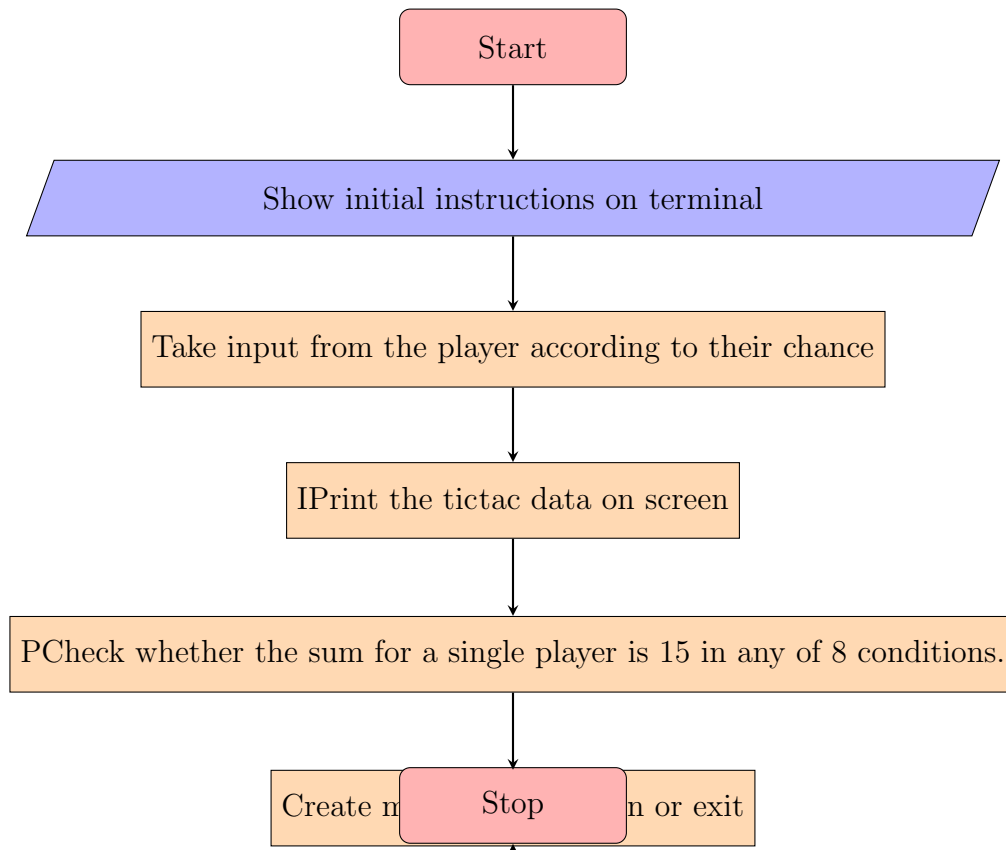
- Welcome to the Game!
- Player 1s chance
- Enter the position and number to be entered: 5,3
- Show tic tac data
- Player 2s chance
- Enter the position and number to be entered: 7,4

## 2.4 Algorithm and Implementation

- Show initial instructions on terminal.
- Take input from the player according to their chance
- Print the tictac data on screen
- Check whether the sum for a single player is 15 in any of 8 conditions.
- Create menu to play again or exit



## 2.5 Program Structure



## 2.6 Screenshots

```
shilpi@admin18-OptiPlex-9020:~/LabAssignment/Assignment_8$ python3 ps2.py
Welcome to the Game!
Player 1's chance
Enter the position and number to be entered:
5,3
5 3
[' ', ' ', ' ']
[' ', 3, ' ']
[' ', ' ', ' ']
Player 2's chance
Enter the position and number to be entered:
4,6
4 6
[' ', ' ', ' ']
[6, ' ', ' ']
[' ', ' ', ' ']
Player 1's chance
Enter the position and number to be entered:
```

Figure 2: Screenshot5

## 3 Appendix

### 3.1 Code for ps1

```
# program for Problem statement-1
import math
def parity(input_lis):      #function to get parity bit
    count=0
    par_list= input_lis
    for bit in input_lis:   #counting no of 1's
        if(bit=='1'):
            count+=1
    parity_bit='0'
    if count%2==0:
        parity_bit='1'
    par_list.append(parity_bit)
    return par_list

def orient_bit(par_data): #Function for bit stuffing
    trans_data=[]
    flag=0
    for x in range(0,len(par_data)):
        trans_data.append(par_data[x])
        flag=flag+1
        if flag >=3: #
            if par_data[x]=='0' and par_data[x-1]=='1' and par_data[x-2]=='0':
                trans_data.append('0')
                flag=0

    return trans_data

def main():
    input_data = input("enter binary data: ")
    input_list = list(input_data)
    parity_bit_data = parity(input_list)
    print("Parity bit data: ",''.join(parity_bit_data))
    transmit_data = orient_bit(parity_bit_data)
    transmit_data.append('0101')
```

```
    print("Transmitting data: ",''.join(transmit_data))

if __name__ == "__main__":
    main()
```

## 3.2 Code for ps2

```
# program for Problem statement-2

import math

def main():
    print("Welcome to the Game!")

    flag=1
    player=""
    while (True):

        if flag:
            player="Player1"
            print("Player 1's chance")
            flag=0

        else:
            player="Player2"
            print("Player 2's chance")
            flag=1

    print("Enter the position and number to be entered: ")
    in_string = [int(x) for x in input().split(',')]
    position=in_string[0]
    number=in_string[1]
    print(position, number)
    if position not in range(1,10) or number not in range(1,10):
        print("Please enter valid number and position")
        exit(1)
    tic_tac = ['_'] * 9
    tic_tac[position-1]= number
    for i in range(3):
        print(tic_tac[i*3:(i+1)*3])

    #if player == "Player1"

if __name__ == "__main__":
    main()
```

## References

- [1] Geeksforgeeks. *du command*. <https://www.geeksforgeeks.org/du-command-linux/>.
- [2] geeksforgeeks. *Lex Program to count number of words*. [https://www.tutorialspoint.com/python3/python\\_dictionary.htm](https://www.tutorialspoint.com/python3/python_dictionary.htm).
- [3] github. *git repo*. [https://github.com/2019JTM2677/Assignment\\_8](https://github.com/2019JTM2677/Assignment_8).