

Assignment-8

ELP-718 Telecom Software Lab

Rohit Kumar Upadhyaya

2019JTM2676

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

1	Problem Statement-1	4
1.1	Problem Statement	4
1.2	Assumptions	4
1.3	Algorithm and Implementation	5
1.4	Inputs and Outputs	5
1.5	Program Structure	6
1.6	Screenshots	7
2	Problem Statement-2	8
2.1	Assumptions	8
2.2	Inputs and Outputs	9
2.3	Algorithm and Implementation	9
2.4	Program Structure	10
2.5	Screenshots	11
3	Appendix	12
3.1	Code for ps1	12
3.2	Code for ps2	14
3.3	GITHUB LINK	16

List of Figures

1	Problem 1 Flowchart	6
2	Output of Runs and Average	7
3	Problem 2 Flowchart	10
4	Output file read	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 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

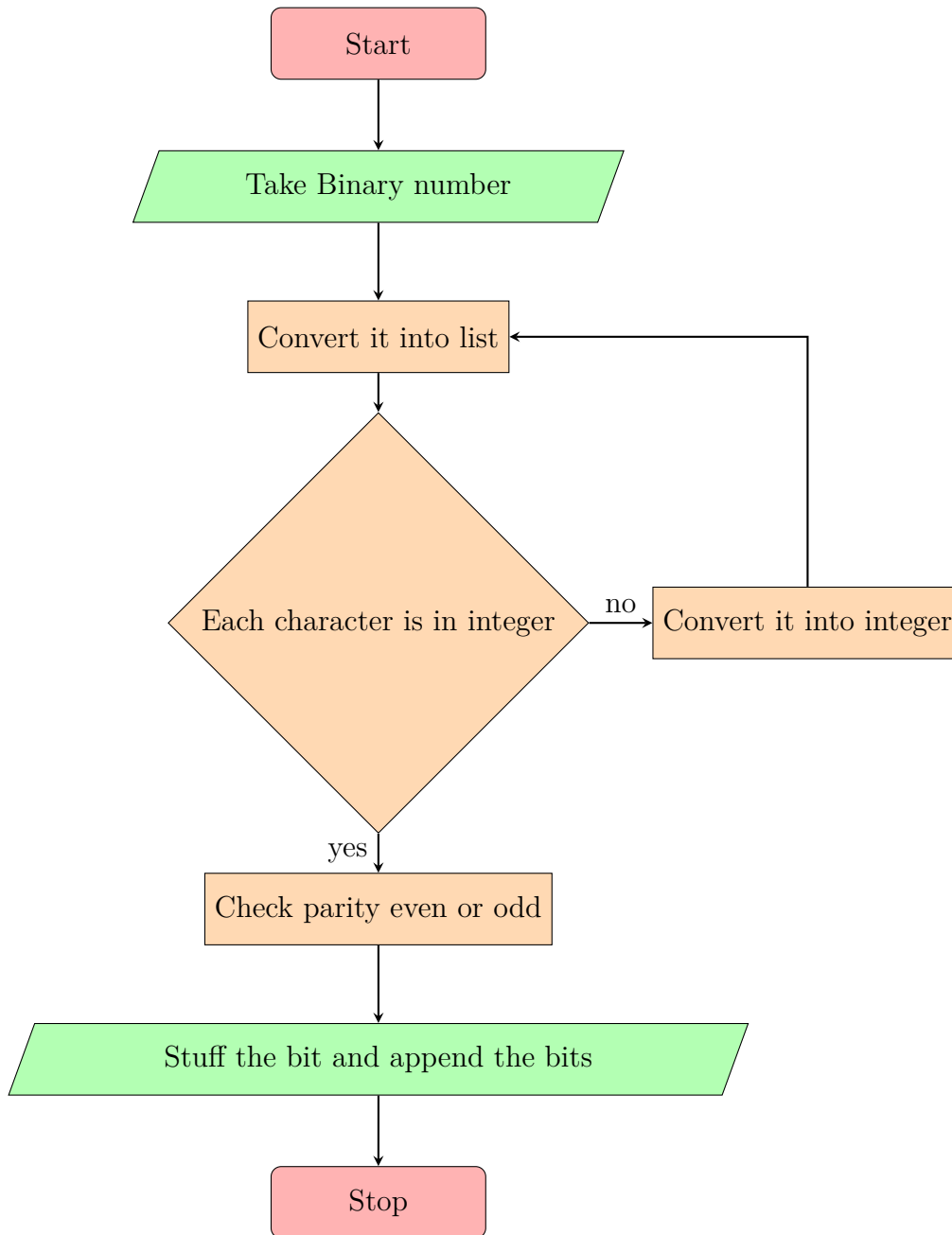
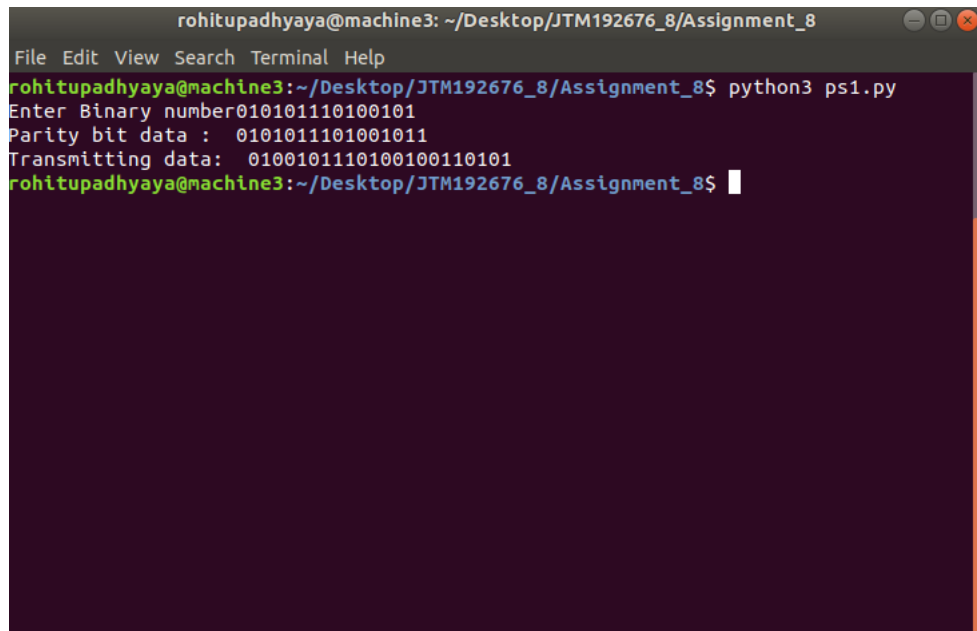


Figure 1: Problem 1 Flowchart

1.6 Screenshots



```
rohitupadhyaya@machine3: ~/Desktop/JTM192676_8/Assignment_8
File Edit View Search Terminal Help
rohitupadhyaya@machine3:~/Desktop/JTM192676_8/Assignment_8$ python3 ps1.py
Enter Binary number010101110100101
Parity bit data : 0101011101001011
Transmitting data: 0100101110100100110101
rohitupadhyaya@machine3:~/Desktop/JTM192676_8/Assignment_8$
```

Figure 2: Output Of Parity and Transmitting data

2 Problem Statement-2

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. $1 \leq Position \leq 9$
2. $1 \leq Number \leq 9$

Terminal:

1. Print Welcome to the Game!.
2. Print whether it is Player 1's or Player 2's 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

	3	

Player 2 chance

Continue till game ends

Enter the position and number to be entered: 7,4

	3	
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

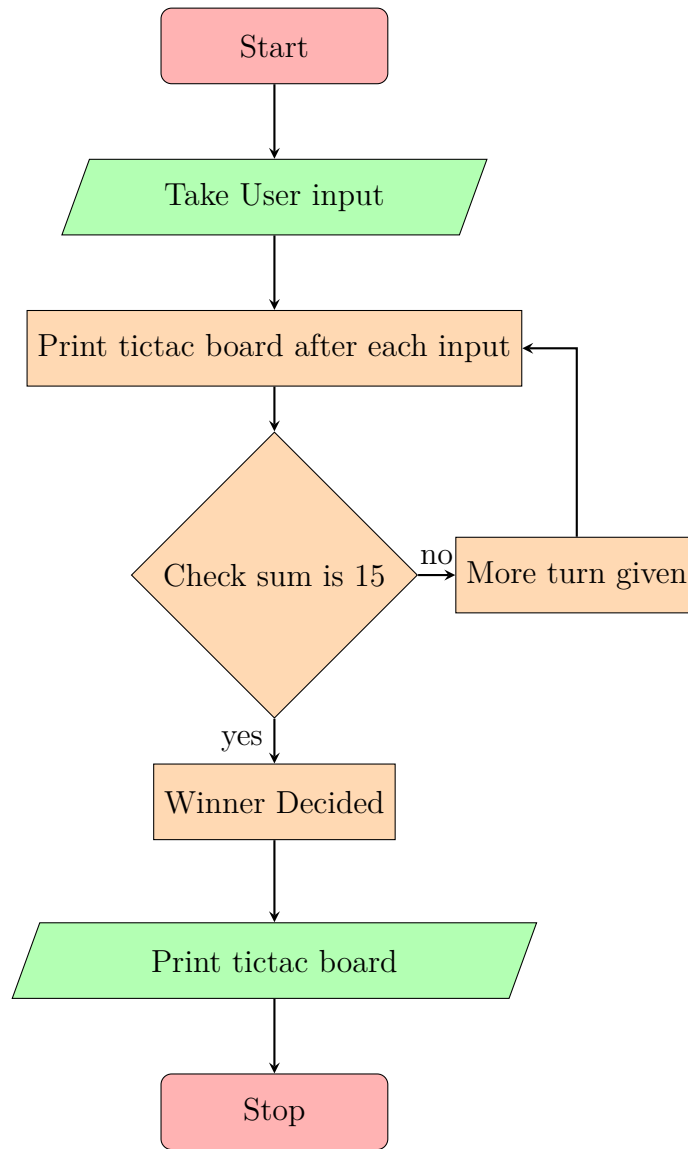


Figure 3: Problem 2 Flowchart

2.5 Screenshots

```
rohitupadhyaya@machine3:~/Desktop/JTM192676_8/Assignment_8$ python3 ps2.py
Welcome to the Game!
Player 1's chance
Enter the position and number to be entered:
3
7
| 0 | 0 | 7 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
Player 2's chance
Enter the position and number to be entered:
4
8
| 0 | 0 | 7 |
| 8 | 0 | 0 |
| 0 | 0 | 0 |
Player 1's chance
Enter the position and number to be entered:
█
```

Figure 4: Output

3 Appendix

3.1 Code for ps1

```
1
2
3
4 import math
5 import sys
6 #Converting binary to decimal
7 def btod(binary):
8     binary1 = binary
9     d, i, n = 0, 0, 0
10    while (binary != 0):
11        dec = binary % 10
12        d = d + dec * pow(2, i)
13        binary = binary // 10
14        i += 1
15    return d
16 #generating a parity bit
17 def Par(n):
18     p = 0
19     while n:
20         p = ~p
21         n = n & (n - 1)
22     return p
23 # Taking input
24 z = (input("Enter Binary number"))
25 b = int(z)
26 n = btod(b)
27
28 if Par(n)==1:
29     t=0
30 else:
31     t=1
32 h=str(t)
33 #Converting string into list
34 y = (list(z))
35 y.append(h)
36 k=""
37 g = k.join(y)
38 #Printing output
39 print ("Parity bit data : ",g)
40 l = len(g)
41 i = 0
42 #Bit stuffing
43 while (i < l):
44     if(y[i] == '0' and y[i+1] == '1' and y[i+2] == '0'):
45         y.insert(i+3,'0')
```

```
46 i=i+4
47 #Bit Appending
48 y.append('0')
49 y.append('1')
50 y.append('0')
51 y.append('1')
52 c = ""
53 q = c.join(y)
54 #Printing Output
55 print("Transmitting data: ",q)
```

3.2 Code for ps2

```
1
2
3 import sys
4 import math
5 game = [0,0,0,
6 0,0,0,
7 0,0,0]
8
9 #Printing box of game
10 def tic_tac_toe ():
11
12 print ( '| ' ,game[0] , '| ' ,game[1] , '| ' , game[2] , '| ' )
13
14
15
16 print ( '| ' ,game[3] , '| ' ,game[4] , '| ' , game[5] , '| ' )
17
18
19
20 print ( '| ' ,game[6] , '| ' ,game[7] , '| ' , game[8] , '| ' )
21
22 #Player1 input
23 def game1():
24 print("Player 1's chance")
25 print("Enter the position and number to be entered: ")
26
27 x1 = int(input())          #Taking position
28 x2 = int(input())          #Taking number
29
30 if x1 < 9:
31 game[x1-1] = x2
32
33 tic_tac_toe()
34 game2()
35 else:
36 print("invalid number")
37
38 #player2 input
39 def game2():
40 print("Player 2's chance")
41 print("Enter the position and number to be entered: ")
42
43 x1 = int(input()) # Taking position
44 x2 = int(input()) # Taking number
45
46 if x1 < 9:
47 game[x1 - 1] = x2
48
```

```
49 tic_tac_toe()
50 game1()
51 else:
52 print("invalid number")
53
54
55
56 print("Welcome to the Game!")
57
58 game1()
59
60
61
62
63
64
65 tic_tac_toe()
66
67 while (true):
68
69 turn(B)
70
71 p1(x1)
72 break
```

3.3 GITHUB LINK

https://github.com/RkJapper/Assignment_8

References

- [1] *Python Programming*. <https://stackoverflow.com/questions/48682011/tic-tac-toe-with-numbers-in-python>.
- [2] IOStream Computing LLC. *Learn Linux*. <https://www.learnlinux.tv/>.
- [3] H Kopka and PW Daly. A guide to $\{\backslash\text{LaTeX}\}$ -document. 1995.
- [4] Medium. *A Quick Primer to Version Control Using Git*. <https://towardsdatascience.com/a-quick-primer-to-version-control-using-git-3fbdbb123262>.