Assignment-8

ELP - 718 Telecom Software Laboratory

Suraj Parihar

2019JTM2678

2019-2021

A report presented for the assignment on

Python programming

Bharti School Of

Telecommunication Technology and Management

IIT Delhi

India

September 18, 2019

Contents

1 Problem Statement-1 3
1.1 Problem Statement 3
1.2 Assumptions 3
1.3 Program Structure 4
1.4 Algorithm and Implementation 5
1.5 Input and Output format 5
1.6 Test cases 5
1.7 Difficulties/Issues faced 5
1.8 Screenshots 6

2 Problem Statement-2 7
2.1 Problem Statement 7
2.2 Assumptions 7
2.3 Program Structure 8
2.4 Algorithm and Implementation 9
2.5 Input and Output format 9
2.6 Test cases 9
2.7 Difficulties/Issues faced 9
2.8 Screenshots 10

3 Appendix 11
3.1 Appendix-A : Code for ps1 11
3.2 Appendix-B : Code for ps2 13

List of Figures

1 Problem 1 Flowchart 4
2 Screenshot1 6
3 Problem 1 Flowchart 8
4 Output2 10
5 Output2 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.

1.2 Assumptions

- the program will not work until user inputs some binary data

1.3 Program Structure

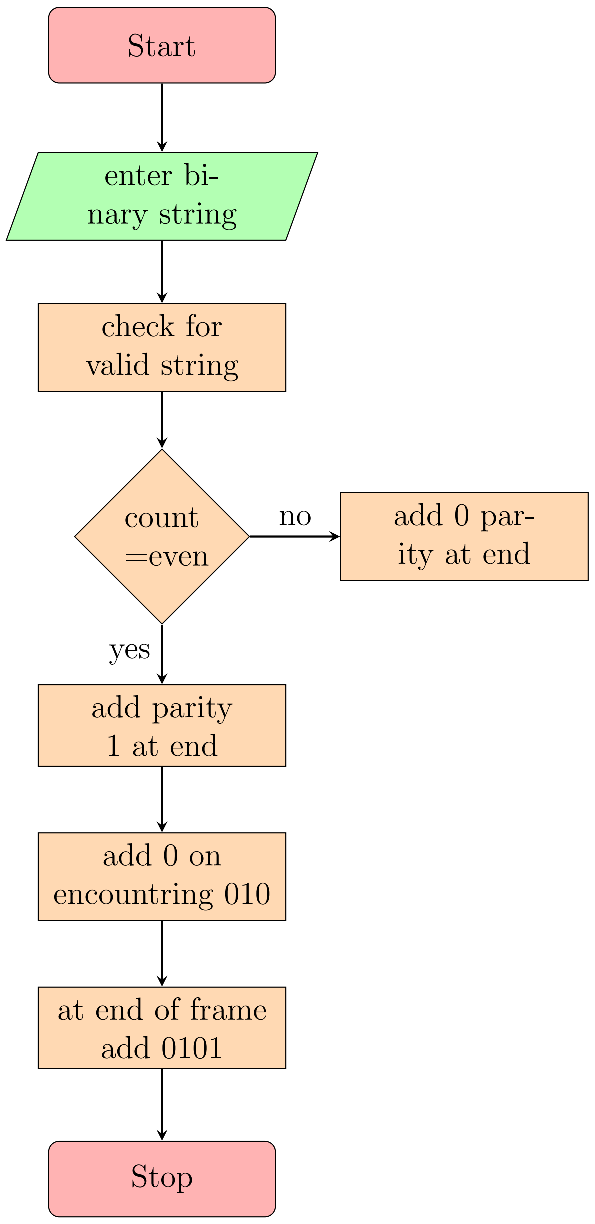


Figure 1: Problem 1 Flowchart

1.4 Algorithm and Implementation

- start
- Enter binary string by user.
- Check for valid binary string
- If string contains 0 and 1's only., then it is valid, otherwise invalid.
- Count no. of 1's in the string.
- If no of 1's is even, append a 1 at the end of the binary string otherwise append 0.
- Find a substring 010 in the string.
- If found, replace it by 0100.
- Print parity corrected string and transmitted bit
- Stop

1.5 Input and 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

1.6 Test cases

Sample Input 010101110100101

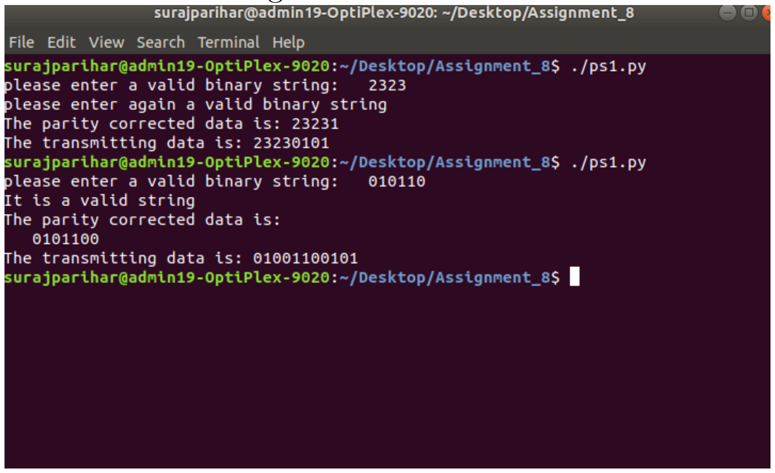
Sample Output
Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101

1.7 Difficulties/Issues faced

- Error code compiling.
- Replacinf the 010 by 0100

1.8 Screenshots

Figure 2: Screenshot1



```
surajparihar@admin19-OptiPlex-9020:~/Desktop/Assignment_8$ ./ps1.py
please enter a valid binary string:   2323
please enter again a valid binary string
The parity corrected data is: 23231
The transmitting data is: 23230101
surajparihar@admin19-OptiPlex-9020:~/Desktop/Assignment_8$ ./ps1.py
please enter a valid binary string:   010110
It is a valid string
The parity corrected data is:
   0101100
The transmitting data is: 01001100101
surajparihar@admin19-OptiPlex-9020:~/Desktop/Assignment_8$ 
```

surajparihar@admin19-OptiPlex-9020: ~/Desktop/Assignment_8
File Edit View Search Terminal Help

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

2.2 Assumptions

- The matrix position starts with index 1

2.3 Program Structure

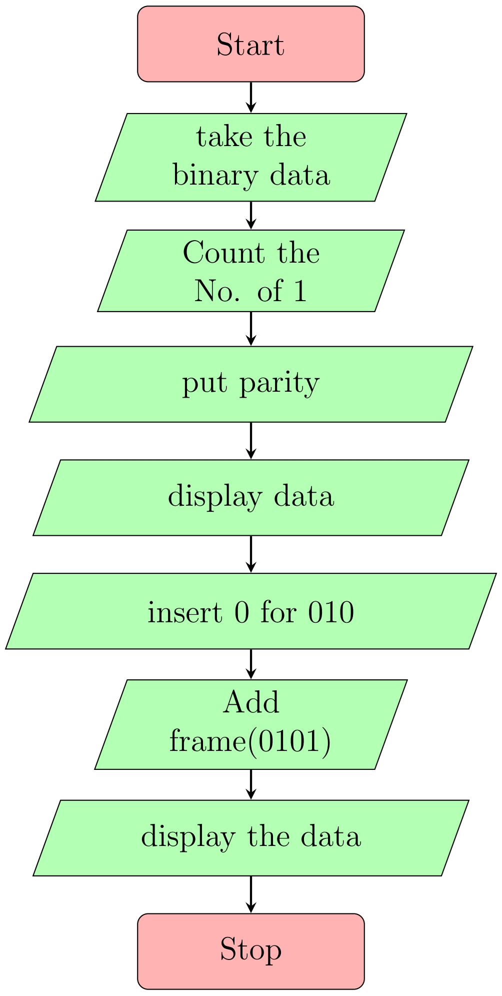


Figure 3: Problem 1 Flowchart

2.4 Algorithm and Implementation

1. Start
2. Enter the choice of player from the user.
3. Print the chosen player.
4. Enter position and number to be entered at that position from the user.
5. Print the board of game after entering data from user.
6. Find sum of rows, columns and diagonals.
7. If sum is equal to 15,declare that player as winner.
8. Print winner and ask user to continue game or exit
9. End

2.5 Input and Output format

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

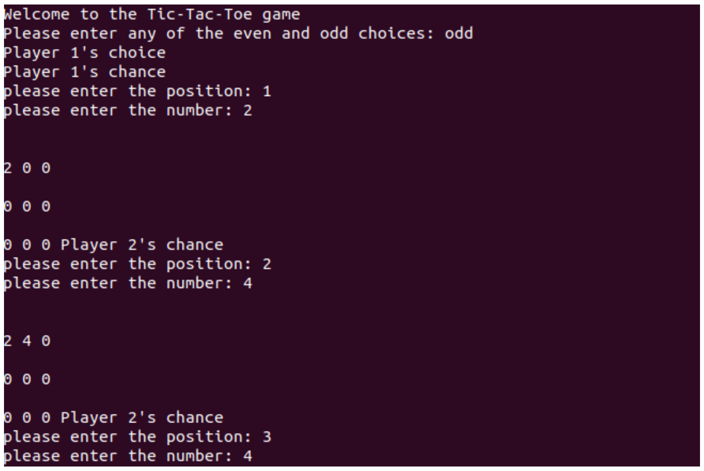
winner is player 1
want to continue or exit

2.6 Test cases

2.7 Difficulties/Issues faced

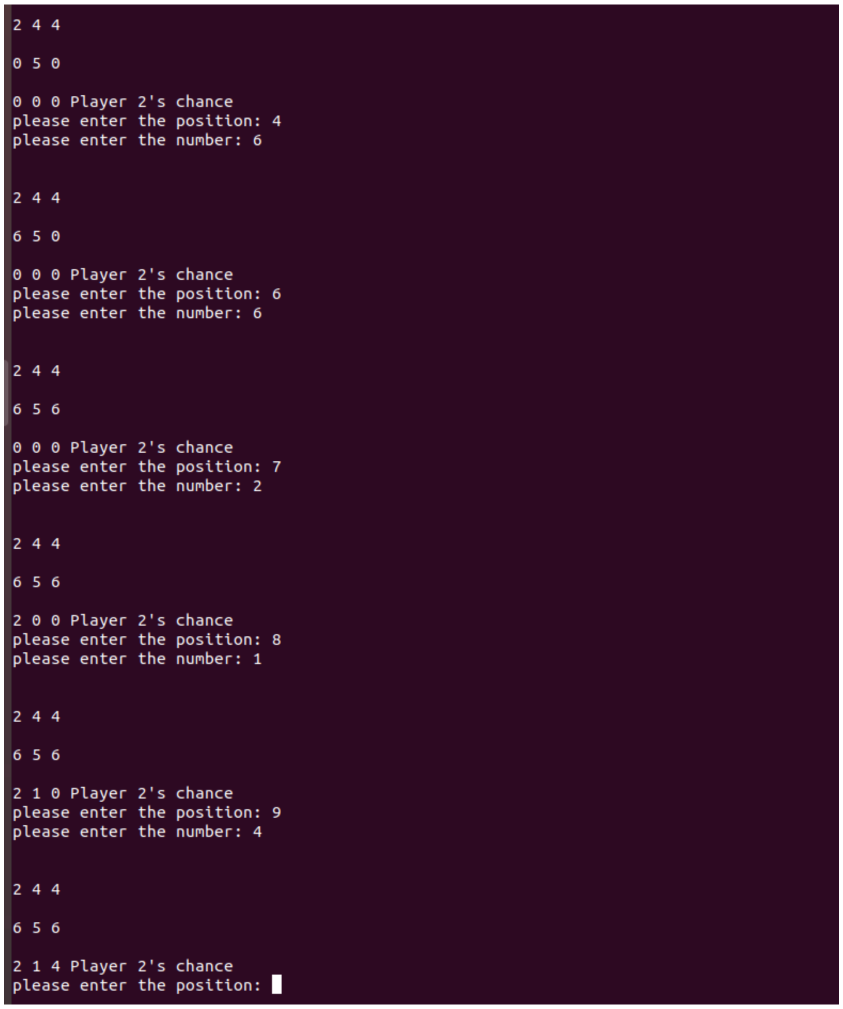
- Error code compiling.
- Calculating sum of all rows,columns and diagonals.

2.8 Screenshots



```
Welcome to the Tic-Tac-Toe game
Please enter any of the even and odd choices: odd
Player 1's choice
Player 1's chance
please enter the position: 1
please enter the number: 2


2 0 0

0 0 0

0 0 0 Player 2's chance
please enter the position: 2
please enter the number: 4


2 4 0

0 0 0

0 0 0 Player 2's chance
please enter the position: 3
please enter the number: 4
```

Figure 4: Output2



```
2 4 4

0 5 0

0 0 0 Player 2's chance
please enter the position: 4
please enter the number: 6


2 4 4

6 5 0

0 0 0 Player 2's chance
please enter the position: 6
please enter the number: 6


2 4 4

6 5 6

0 0 0 Player 2's chance
please enter the position: 7
please enter the number: 2


2 4 4

6 5 6

2 0 0 Player 2's chance
please enter the position: 8
please enter the number: 1


2 4 4

6 5 6

2 1 0 Player 2's chance
please enter the position: 9
please enter the number: 4


2 4 4

6 5 6

2 1 4 Player 2's chance
please enter the position: 
```

Figure 5: Output2

3 Appendix

3.1 Appendix-A : Code for ps1

```
#!/usr/bin/python3
#program for bit stuffing and parity check

def check(string):                    #function to check for a valid string
p=set(string)
s={'0','1'}
if s==p or p=={'0'} or p=={'1'}:
print("It is a valid string")
else:
print("please enter again a valid binary string")
```

```
if __name__ == "__main__" :
string=input("please enter a valid binary string:   ")
check(string)                     #calling check function


substring='1'
count=string.count(substring)
#print("count is:",count)
if count%2==0:                                        #to add parity bit at the end of
string2=string+'1'
print("The parity corrected data is:",string2)      #print parity added data
else:
string2=string+'0'
print("The parity corrected data is:\n  ",string2)


string3=string.replace("010","0100")                      #replacing the '010' substring
string4=string3+"0101"
print("The transmitting data is:",string4)
```

3.2 Appendix-B : Code for ps2

```
#!/usr/bin/python3
#program for a 3*3 tic-tac-toe
def entry(player):
count=0
j=0
lst=[0,0,0,0,0,0,0,0,0]                               #function to add values to the ls
for j in range(0,10):
posi=input("please enter the position: ")
num=input("please enter the number: ")
if int(posi)==(1 or 3 or 5 or 7 or 9):
lst[int(posi)-1] = int(num)
else:
lst[int(posi) - 1] = int(num)
for i in range(0,9):
if(i%3==0):
print("\n")
print(lst[i],end=" ")
i=i+1

if player in "player 1's chance":
player="Player 1's chace"
print(player)
else:
player="Player 2's chance"
print(player)

def winner():
if (lst[1]+lst [2]+lst[3]==15):
print ('{0} are the winner' .format(player))
return True

if (lst[4]+lst [5]+lst[6]==15):
print ('{0} are the winner' .format(player))
return True

if (lst[2]+lst [5]+lst[8]==15):
print ('{0} are the winner' .format(player))
return True

if (lst[3]+lst [7]+lst[5]==15):
print ('{0} are the winner' .format(player))
return True
```

```
if (lst[1]+lst [5]+lst[9]==15):
print ('{0} are the winner' .format(player))
return True

if (lst[7]+lst [8]+lst[9]==15):
print ('{0} are the winner' .format(player))
return True

else: return False

print("Welcome to the Tic-Tac-Toe game")                                  #taking choic
choice=input("Please enter any of the even and odd choices: ")
if choice=="odd":
print("Player 1's choice")
elif choice=="even":
print("Players 2's choice")
else:
print("Invalid choice")
exit(0)
player="Player 1's chance"
print(player)
entry(player)
while (True):
if winner(): break
```

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.