Assignment-8

ELP - 718 Telecom Software Laboratory

Santosh Kumar

2019JTM2088

2019-2021

A report presented for the assignment on

Python Basics 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

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 1's 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: 0100101110100010011010 1

1.2 Algorithm and Implementation

part 1

1. First get the binary number.
2. Convert the binary to decimal.

Part 2:

1. second check the deimal having the parity or not.

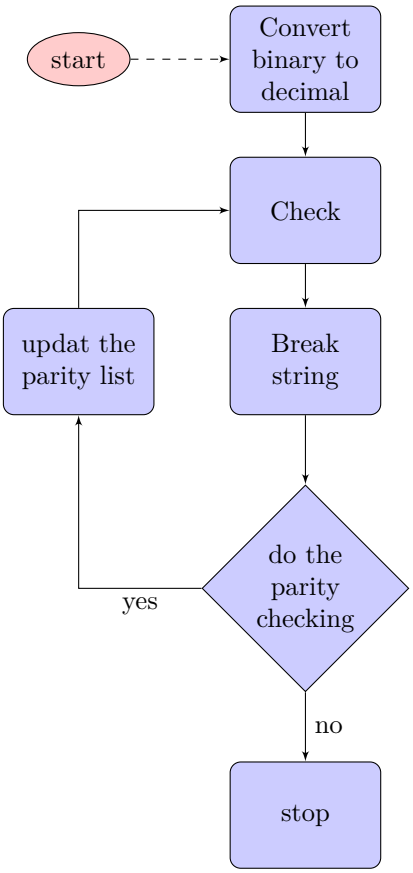
Part 3:

1. if Parity is not then add 1 to make a parity using list.

Part 4:

1. Do the replacing of string with specific pattern given.

1.3 Program Structure



1.4 Screenshots

```
santoshkumar@machine7:~/Desktop/Assignment_8$ ./ps1.py
Enter binary bit data that has to be transmitted.
1001010101
odd parity
1001010101
santoshkumar@machine7:~/Desktop/Assignment_8$
```

Figure 1: Output of ps1

```
antoshkumar@machine7:~/Desktop/Assignment_8$ git add .
antoshkumar@machine7:~/Desktop/Assignment_8$ git status
n branch master
our branch is ahead of 'origin/master' by 1 commit.
 (use "git push" to publish your local commits)

hanges to be committed:
 (use "git reset HEAD <file>..." to unstage)

       modified:   .ps1.py.swp
       new file:   .ps2.py.swp
       modified:   ps1.py

antoshkumar@machine7:~/Desktop/Assignment_8$ git commit -m "All are commited"
master 66d9222] All are commited
3 files changed, 19 insertions(+), 6 deletions(-)
create mode 100644 .ps2.py.swp
antoshkumar@machine7:~/Desktop/Assignment_8$ git status
n branch master
our branch is ahead of 'origin/master' by 2 commits.
 (use "git push" to publish your local commits)

othing to commit, working tree clean
antoshkumar@machine7:~/Desktop/Assignment_8$ git push -u origin master
sername for 'https://github.com': 2019JTM2088
assword for 'https://2019JTM2088@github.com':
ounting objects: 8, done.
elta compression using up to 8 threads.
ompressing objects: 100% (8/8), done.
riting objects: 100% (8/8), 1.89 KiB | 969.00 KiB/s, done.
otal 8 (delta 3), reused 0 (delta 0)
emote: Resolving deltas: 100% (3/3), completed with 3 local objects.
o https://github.com/2019JTM2088/Assignment_8.git
  69e9c63..66d9222  master -> master
ranch 'master' set up to track remote branch 'master' from 'origin'.
```

Figure 2: Git commit of ps1

1.5 Difficulty faced

1.

2 Problem Statement-2

2.1 Problem Statement

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:**
• $1 <= Position <= 9$
• $1 <= Number <= 9$
Terminal:
• Print 'Welcome to the Game!'.
• Print whether it is Player 1's or Player 2's chance.
• Get the position and number to be entered from the user.
• Show tic tac toe with data.
• 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.

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

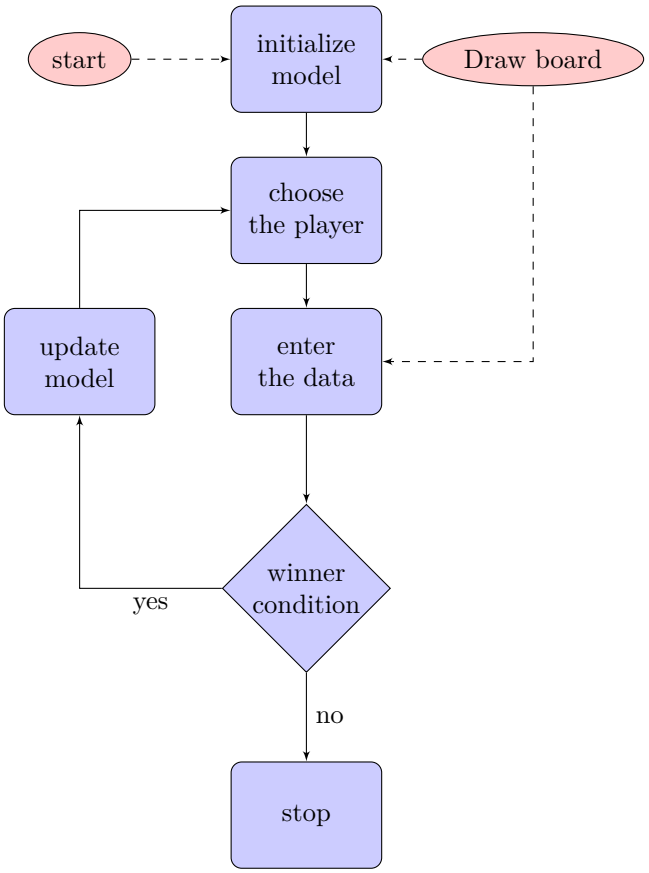
Player 2's chance Enter the position and number to be entered: 7,4

Continue till game ends Note – Must use at least one User Defined Function.

2.2 Algorithm and Implementation

Step 1
Draw the Board
Step 2
Second Choose the player.
Step 3
Enter the data and do check for another option
Step 4
show the board for next entery.
Step 5
After completion of data filling check the coditon for winner.

2.3 Program Structure



2.4 Screenshots

```
antoshkumar@machine7:~/Desktop/Assignment_8$ git add .
antoshkumar@machine7:~/Desktop/Assignment_8$ git commit -m "Second program board drawing"
master 9867a12] Second program board drawing
2 files changed, 20 insertions(+), 15 deletions(-)
mode change 100644 => 100755 ps2.py
antoshkumar@machine7:~/Desktop/Assignment_8$ git status
n branch master
our branch is ahead of 'origin/master' by 1 commit.
 (use "git push" to publish your local commits)

othing to commit, working tree clean
antoshkumar@machine7:~/Desktop/Assignment_8$ git push -u origin master
sername for 'https://github.com': 2019JTM2088
assword for 'https://2019JTM2088@github.com':
ounting objects: 4, done.
elta compression using up to 8 threads.
ompressing objects: 100% (4/4), done.
riting objects: 100% (4/4), 889 bytes | 889.00 KiB/s, done.
otal 4 (delta 2), reused 0 (delta 0)
emote: Resolving deltas: 100% (2/2), completed with 2 local objects.
o https://github.com/2019JTM2088/Assignment_8.git
  66d9222..9867a12  master -> master
ranch 'master' set up to track remote branch 'master' from 'origin'.
```

Figure 3: Git commit of ps2

```
santoshkumar@machine7:~/Desktop/Assignment_8$ chmod +x ps2.py
santoshkumar@machine7:~/Desktop/Assignment_8$ ./ps2.py
'Welcome to the Game!'
 _____  _____  _____
|     ||     ||     |
|     ||     ||     |
 _____  _____  _____
|     ||     ||     |
|     ||     ||     |
 _____  _____  _____
|     ||     ||     |
|     ||     ||     |
 _____  _____  _____

santoshkumar@machine7:~/Desktop/Assignment_8$
```

Figure 4: Output of ps2

2.5 Difficulty faced

1.

A Appendix

A.1 Code for ps1

```
#! /usr/bin/python3
# Python3 program to convert
# binary to decimal

# Function to convert
# binary to decimal
def binaryToDecimal(n):
num = n;
dec_value = 0;

# Initializing base
# value to 1, i.e 2 ^ 0
base = 1;

temp = num;
while(temp):
last_digit = temp % 10;
temp = int(temp / 10);

dec_value += last_digit * base;
base = base * 2;
return dec_value;

# Driver Code
print("Enter binary bit data that has to be transmitted.")

num =int(input(),2)
binaryToDecimal(num)




def getParity( num ):
parity = 0
while num:
parity = ~parity
num = num & (num - 1)
return parity

# Driver program to test getParity()
#num = 7
print (
( "odd parity" if getParity(num) else "even parity"))




# Function to convert Decimal number
```

```
# to Binary number

def decimalToBinary(num):
return bin(num).replace("0b","")

# Driver code
if __name__ == '__main__':
print(decimalToBinary(num))
```

A.2 Code for ps2

```
#! /usr/bin/python3

print("'Welcome to the Game!'")
# draw a board:
def draw():
    board = ''

    for i in range(-1,6):

        if i%2==0:
            board += '|      ' * 4
            board += '\n|      |      |      |'

        else:
            board += ' _____ ' * 3

        board += '\n'
    print (board)

draw()
List1=[1,3,5,7,9]
List2=[2,4,6,8]

# Check for empty places on board
def possibilities(board):
    l = []

    for i in range(len(board)):
        for j in range(len(board)):

            if board[i][j] == 0:
                l.append((i, j))
    return(l)

while not winner :
    draw()

    if playerOneTurn :
        print( "Player 1:")
    else :
        print( "Player 2:")
```

A.3 GIt link

@manualref1, title = "Python Crash Course - Cheat Sheets", note = "`https://github.com/2019JTM2088/Assignment_8.git`",

References

[1] *Python - Overview*. https://www.tutorialspoint.com/python/python_overview.htm.

[2] *Python Crash Course - Cheat Sheets*. https://ehmatthes.github.io/pcc/cheatsheets/README.html.

[3] *Python Programming Language*. https://www.geeksforgeeks.org/python-programming-language/.

[4] *Python Tutorial*. https://www.javatpoint.com/python-lists.