Assignment 8

ELP - 718 Telecommunication Software Laboratory

Varun Gupta

2019BSY7505

2019-2021

A report presented for the assignment on

Python

Bharti School Of

Telecommunication Technology and Management

IIT Delhi

India

September 18, 2019

Contents

List of Figures

Problem Statement-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.

Assumptions

- The bit is provided in the single line
- Use of predefined functions is allowed

Program structure

Algorithm and Implementation

1. Input the bit stream from the user
2. Check the no of 1 in the string
3. if no of 1 is odd, add 1 to the string
4. if no of 1 is even add 0 to string
5. search for substring 010
6. replace the substring 010 with o1oo
7. append 0101 bit at the end of the frame
8. display the result

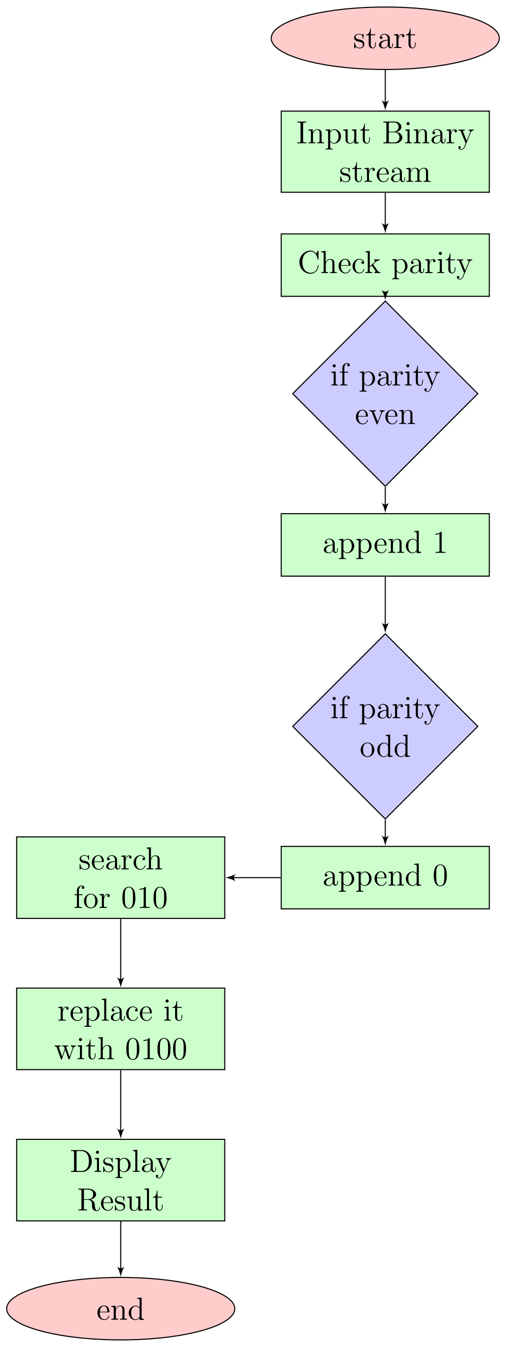


Figure 1: Flow chart 1

Input 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

Test Cases

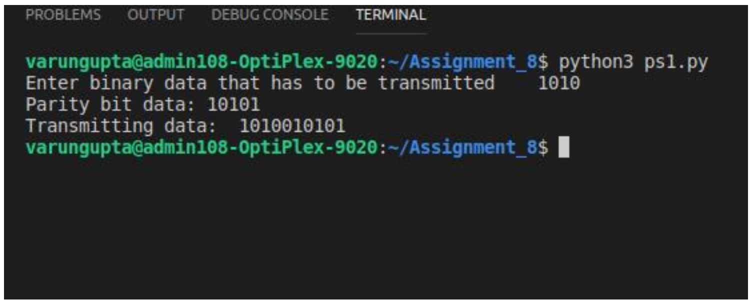
Sample Input

010101110100101

Sample Output

Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101

Screenshot



```
varungupta@admin108-OptiPlex-9020:~/Assignment_8$ python3 ps1.py
Enter binary data that has to be transmitted    1010
Parity bit data: 10101
Transmitting data:  1010010101
varungupta@admin108-OptiPlex-9020:~/Assignment_8$ 
```

Figure 2: Screenshot1

Problem Statement-2

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

Assumptions

1. value of position lies betweeen 0 to 8
2. value of number lies between 1 to 9
3. the text of number is not displayed on tthe terminal for user.

Program Structure

Algorithm and implementation

- Welcome message
- Randomly picks a player
- Player 1 chosses odd no always
- Input data one by one from the players
- checking if data is within the constrain
- If not in constrain, the show error message
- List the point taken by player 1 and Player 2
- Try to find three collinear points out of any of this list
- If found collinesr, then evaluate the sum of the values at theat points
- If the sum is greater than 15

- Declare the player as winner
- Do it till all thge 9 values are inputted
- If none of the player is winner
- Declare draw
- End

Input Output Format

Input Format

- 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.
- 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.

Output Format

Player 1 won

Test cases

Sample Input

```
Welcome to the game
It is player 2 's chance
Enter the position2
Enter the number to be entered45
Enter valid no between 0 to 9
It is player 2 's chance
```

Sample Output

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

Screenshot

Appendix

Appendix-A : Code for ps1

```
data = input("Enter binary data that has to be transmitted\t")     #Inputting binary string
count=0
for i in data:
    if i == '1':
        count+=1   # if even parity
if (count%2) == 0:
   data = data + '1'    #add 1 at the end
else:
    data = data + '0'   #add 0 for odd
print('Parity bit data:', data)
data = data.replace('010', '0100')
data = data + '0101'
print("Transmitting data: ",data)
```

Appendix-B : Code for ps2

```python
import random
from itertools import combinations

def collinear(x1,y1,x2,y2,x3,y3):
    a = x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2)
    if (a == 0):
        return 1
    else:
        return 0
def evaluate(str):
    collpnt = []
    comb = combinations(list(range(0,len(str))), 3)
    for i in list(comb):
        j= i[0]
        x1 = str[j] % 3
        y1 = str[j] // 3
        k = i[1]
        x2 = str[k] % 3
        y2 = str[k] // 3
        l = i[2]
        x3 = str[l] % 3
        y3 = str[l] // 3
        if collinear(x1,y1,x2,y2,x3,y3):
            collpnt.append(i)
    return collpnt



data = [0,0,0,0,0,0,0,0,0]
pos1 = []
pos2 = []
a = []
b = []
print("Welcome to the game")
y = random.randint(1,2)
o  = y
chan = 0
while 1:
    print("It is player",o,"'s chance")
    pos = int(input("Enter the position "))
    if pos > 8 or pos < 0:
        print('Enter valid Position between 0 to 8')
        continue
    no = int(input("Enter the number to be entered "))
    if no > 9 or no < 0:
        print('Enter valid no between 0 to 9')
        continue
    if (no % 2) == 0 and (chan % 2) == 0:
        print('Enter any odd no between 0 to 9')
        continue
    if (no % 2) != 0 and (chan % 2) != 0:
        print('Enter any even no between 0 to 9')
        continue
    data[pos] = no
    print(data[0:3])
    print(data[3:6])
    print(data[6:9])
    if (chan % 2) == 0:
        pos1.append(pos)
    if (chan % 2) != 0:
        pos2.append(pos)
    if len(pos1)>=3:
        a = evaluate(pos1)
    if len(pos2)>=3:
        b = evaluate(pos2)
    if(len(pos1)>=3):
        for k in a:
            if data[pos1[k[0]]] +  data[pos1[k[1]]] + data[pos1[k[2]]] >= 15:
                print("Player",y,"wins")
                exit()
    if(len(pos2)>=3):
        for l in b:
            if data[pos2[l[0]]] +  data[pos2[l[1]]] + data[pos2[l[2]]] >= 15:
                print("Player",y,"doesnt wins")
                exit()
    if chan == 8:
        print("Match Draw")
        exit()
    chan+=1
    if o == 1:
        o = 2
    else:
        o = 1
    print(pos1, pos2)
```

References

[1] https://www.geeksforgeeks.org/

[2] www.overleaf.com

[3] https://stackoverflow.com/

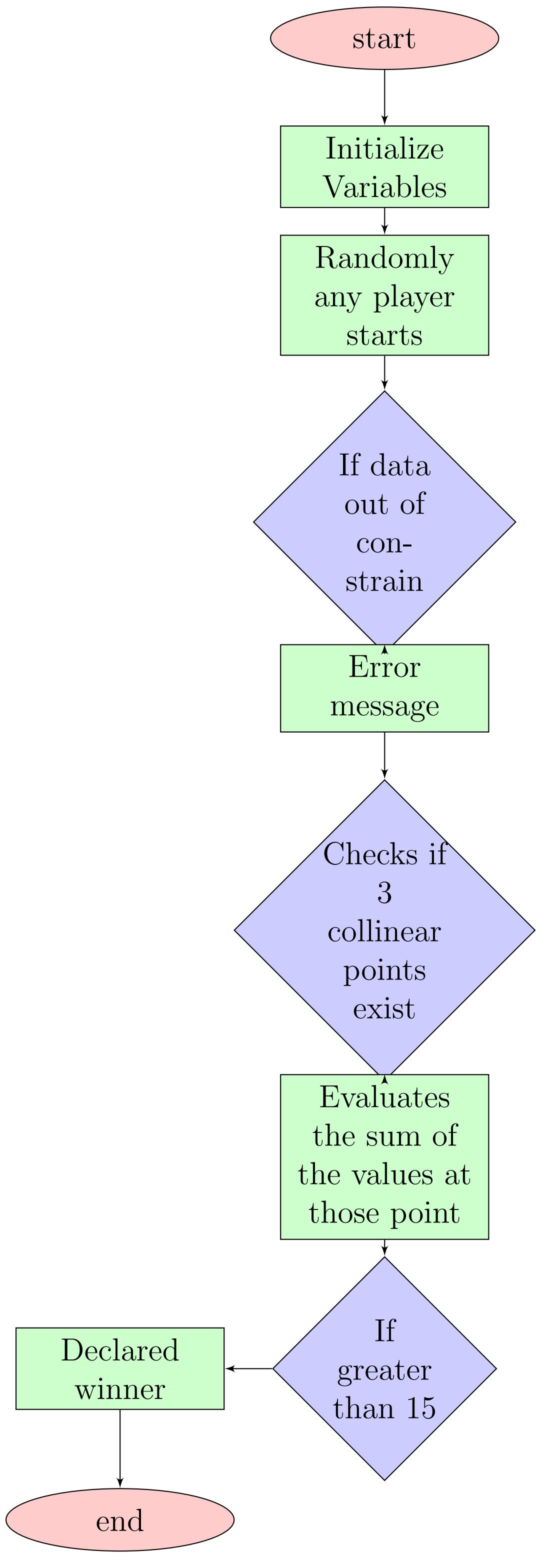
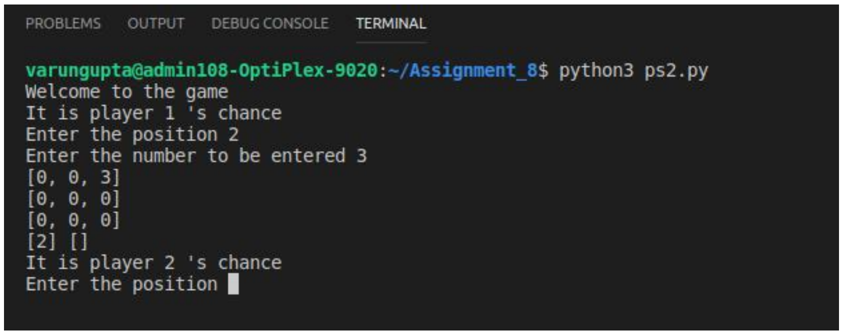

start
Initialize Variables
Randomly any player starts
If data out of con- strain
Error message
Checks if 3 collinear points exist
Evaluates the sum of the values at those point
If greater than 15
Declared winner
end

Figure 3: Flow chart 2



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
varungupta@admin108-OptiPlex-9020:~/Assignment_8$ python3 ps2.py
Welcome to the game
It is player 1 's chance
Enter the position 2
Enter the number to be entered 3
[0, 0, 3]
[0, 0, 0]
[0, 0, 0]
[2] []
It is player 2 's chance
Enter the position 
```

Figure 4: Screenshot2