Assignment-8

ELP-718 Telecom Software Laboratory

Nimish

2019JTM2675

2019-2021

A report for the assignment on Basic Python.

Bharti School Of

Telecommunication Technology And Management

IIT Delhi

India

Contents

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 1âs 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.

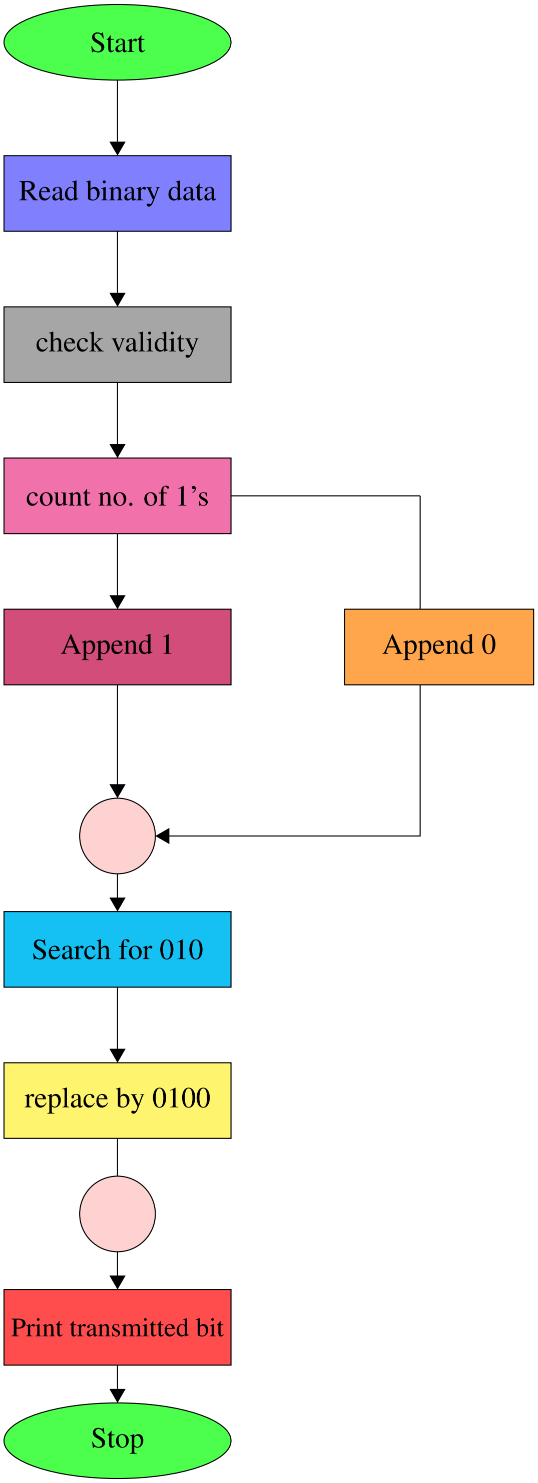
1.2 Assumptions

- here we have assumed that the order of storing of information about login credentials and marks are in the same order.

1.3 Algorithm and Implementation

- 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

1.4 Flowchart



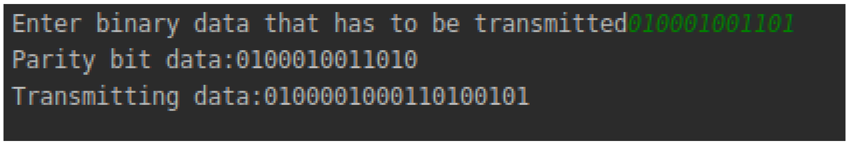
1.5 Input and Output Format

here we entered 5 students and two instructo details along with the password and category.

after that wee netered the marks scored by student in 5 subjects.

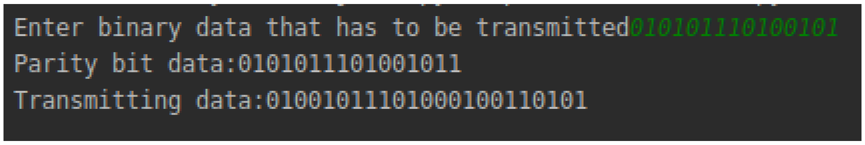
then we asked for login id from the user and when it matches successfully we ask for password.

if the password also matches correctly then we provide them with 7 different options to chose from and whatever they chose we display according to that.



```
Enter binary data that has to be transmitted010001001101
Parity bit data:0100010011010
Transmitting data:0100001000110100101
```

Figure 1: when bit0 is appended



```
Enter binary data that has to be transmitted010101110100101
Parity bit data:0101011101001011
Transmitting data:01001011101000100110101
```

Figure 2: when bit1 is appended

2 Problem Statement-2

2.1 Problem Statement

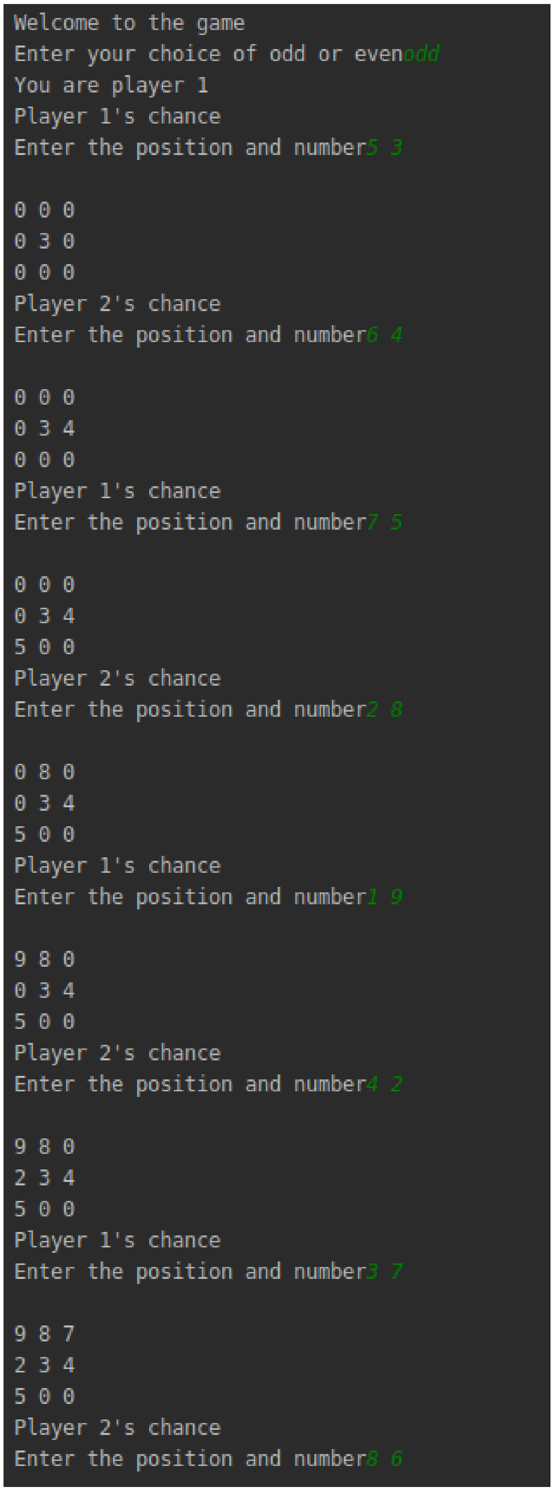
- 3X3 Numeric Tic-Tac-Toe (Use numbers 1 to 9 instead of X and O) One player plays with the odd numbers (1, 3, 5, 7, 9) and the other player plays with the even numbers (2,4,6,8). All numbers can be used only once. The player who puts down 15 points in a line wins (sum of 3 numbers). Always Player with 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 â 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 or Player 2 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.

2.2 Algorithm and Implementation

- Enter the choice of player from the user.
- Print the chosen player.
- Enter position and number to be entered at that position from the user.
- Print the board of game after entering data from user.
- Find sum of rows, columns and diagonals.
- If sum is equal to 15,declare that player as winner.
- Print winner and ask user to continue game or exit

2.3 Input and Output Format

when we compile the program it gives th frequency of each character in alphabets .



```
Welcome to the game
Enter your choice of odd or evenodd
You are player 1
Player 1's chance
Enter the position and number5 3

0 0 0
0 3 0
0 0 0
Player 2's chance
Enter the position and number6 4

0 0 0
0 3 4
0 0 0
Player 1's chance
Enter the position and number7 5

0 0 0
0 3 4
5 0 0
Player 2's chance
Enter the position and number2 8

0 8 0
0 3 4
5 0 0
Player 1's chance
Enter the position and number1 9

9 8 0
0 3 4
5 0 0
Player 2's chance
Enter the position and number4 2

9 8 0
2 3 4
5 0 0
Player 1's chance
Enter the position and number3 7

9 8 7
2 3 4
5 0 0
Player 2's chance
Enter the position and number8 6
```

Figure 3: Each entry by a player

3 Appendix

3.1 Appendix-A : code for ps1

ps1.py

```
# getting input data
data = input(”Enter␣binary␣data␣that␣has␣to␣be␣transmitted”)
count = 0
for i in range(0, len(data)):   # starting of loop till the end of data
    if data[i] in ’01’:
        if data[i] in ’1’:
            count += 1
    else:
        print(”Please␣enter␣a␣valid␣binary␣data”)   # checking of boundary condition
        break
if (count % 2 == 0):    #checking for even parity bit
    print(”Parity␣bit␣data:{0}”.format(data + ”1”))
    data2 = data.replace(”010”, ”0100”) #searching for 010 and replacing with 0100
    print(”Transmitting␣data:{0}”.format(data2 + ”10101”))  #final transmitted data
else:
    print(”Parity␣bit␣data:{0}”.format(data + ”0”))
    data2=data.replace(”010”,”0100”)
    print(”Transmitting␣data:{0}”.format(data2+”00101”))
```

3.2 Appendix-B : code for ps2

ps2.py

```
def game(player):   #function definition
    j=0
    lst = [0, 0, 0, 0, 0, 0, 0, 0, 0]   #defining a list
    while j<9:
        x,y=input("Enter␣the␣position␣and␣number").split()  #reading multiple inputs
        if int(x)==(1 or 3 or 5 or 7 or 9): #updating the value of list whenver the players enter a
            lst[int(x)-1]=int(y)
        else:
            lst[int(x) - 1] = int(y)
        i=0
        while i<9:
            if (i%3==0):
                print()
            print(lst[i],end="␣")
            i+=1
        print()
        if player in "Player␣1's␣chance":   #defining the chances of the players
            player="Player␣2's␣chance"
            print(player)
        else:
            player="Player␣1's␣chance"
            print(player)
        j += 1
    c_sum = [0, 0]  #finding the sum of columnn elements
    i = 0
    k = 0
    while i < 3:
        while k < 9:
            c_sum[i] = c_sum[i] + lst[k]
            k += 3
    print("column␣elemnts␣sum␣is:") #printing the sum of column elements
    i = 0
    while i < 3:
        print(c_sum[i])
    d_sum = [0, 0]  #finding the diagonal elements sum
    i = 0
    k = 0
    while i < 2:
        while k < 9:
            d_sum[i] = d_sum[i] + lst[k]
            k += 4
        i += 1
        print("Diagonal␣elemnts␣sum␣is:")   #printing of diagonal elements sum
        i = 0
    while i < 2:
        print(d_sum[i])
    r_sum = [0, 0, 0]   #finding row elemnts sum
    i = 0
    k = 0
    while i < 3:
        while k < 3:
            r_sum[i] = r_sum[i] + lst[k]
            k += 1
        i += 1
        print("Row␣elemnts␣sum␣is:")    #printing row elements sum
        i = 0
    while i < 3:
        print(r_sum[i])


print(”Welcome␣to␣the␣game”)
choice=input(”Enter␣your␣choice␣of␣odd␣or␣even”)     #asking the player to chose
if choice==”odd”:
    print(”You␣are␣player␣1”)
else:
    print(”You␣are␣player␣2”)
player=”Player␣1’s␣chance”
print(player)
game(player)     #function calling
```

References

[1] https://docs.python.org/3/

[2] https://www.geeksforgeeks.org/python-programming-language/

[3] String comparision
https://git-scm.com/do

[4] String concept
https://www.atlassian.com/git/