Assignment-8
ELP- 718 Telecom Software Laboratory

Utkarsh badal
2019JTM2679
2019-20

A report presented for the assignment on
*PYTHON BASICS**

Bharti School
of
Telecommunication Technology and Management
IIT DELHI, Delhi
India
September 18, 2019

Contents

List of Figures

1 Problem Statement-1

1.1 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.

1.2 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.

1.3 Assumptions

- input is taken as binary string

1.4 Algorithm and Implementation

- Take input bit stream from user as string.
- Typecasted it into integers.
- Converted into list.
- Checked for odd and even parity.
- If odd parity then added 0 in the last.
- If even parity then added 1 in the last .
- Stuffing of bit 0 after 010
- Added 0101 at the end of the list to show that bit stream is ended.

1.5 Flow Chart

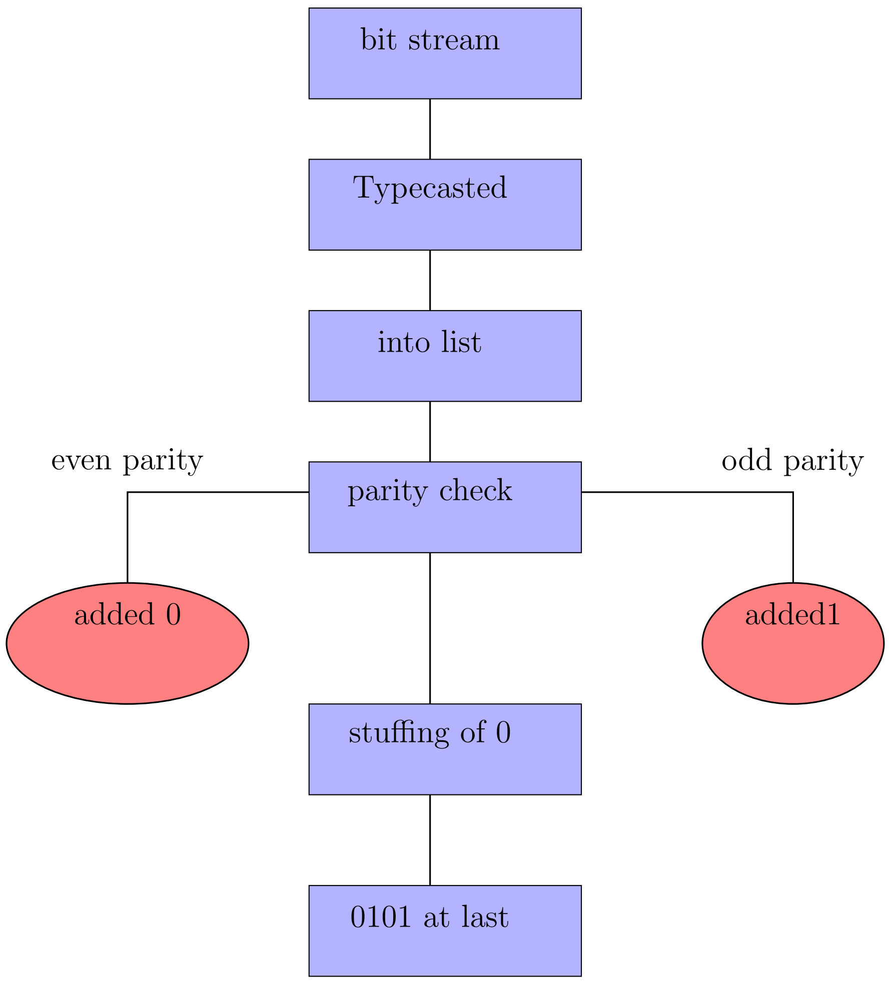


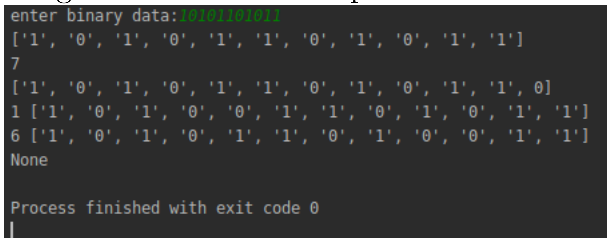
Figure 1: Flow Chart for Figure 1

1.6 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

Figure 2: Terminal Output of Problem 1



```
enter binary data:10101101011
['1', '0', '1', '0', '1', '1', '0', '1', '0', '1', '1']
7
['1', '0', '1', '0', '1', '1', '0', '1', '0', '1', '1', 0]
1 ['1', '0', '1', '0', '0', '1', '1', '0', '1', '0', '1', '1']
6 ['1', '0', '1', '0', '1', '1', '0', '1', '0', '0', '1', '1']
None

Process finished with exit code 0
```

1.7 Screenshots

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 Constraints

- position should be in between (0,9)
- Number should be in between (1,9)

2.3 Terminal

- 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.

2.4 Assumptions

- i am assuming here that player will not enter number in the filled place again.
- player will take numbers from it's list only.

2.5 Algorithm and Implementation

- took lists list1 and list2.
- which ever player plays first giving list1 to him.
- taking position and number from players
- writing in the tic tac toe table and running loop again.

2.6 Flow Chart

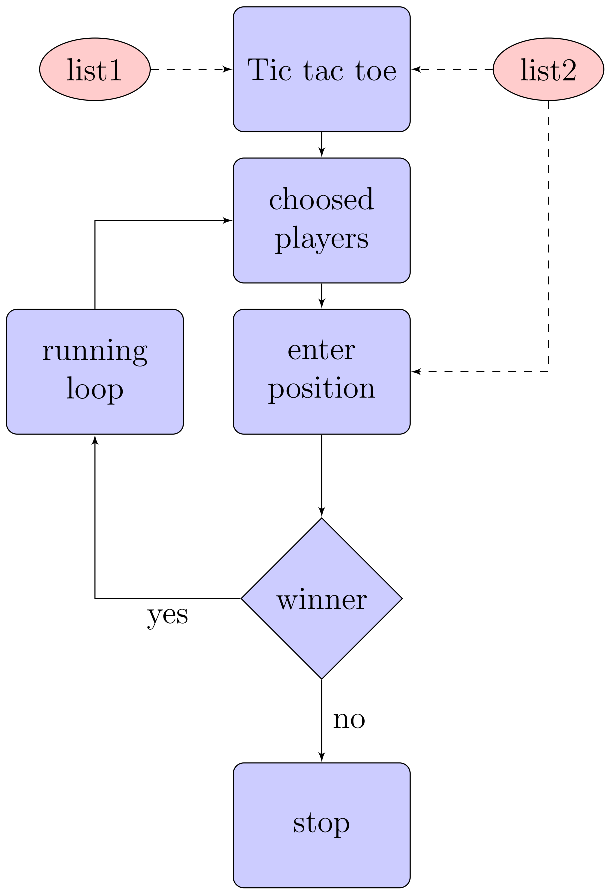


Figure 3: Flow Chart of Problem 2

2.7 Input and Output Format

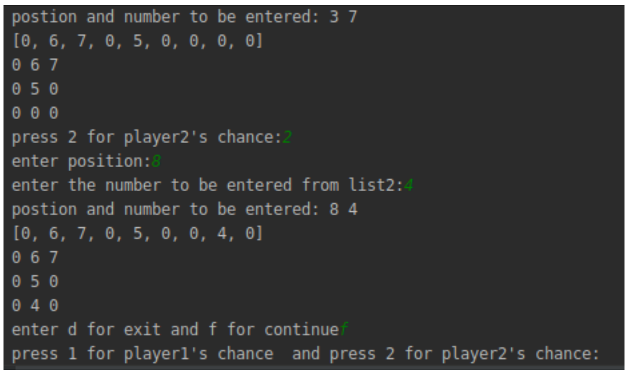
Input Format:

- list1
- list2

Output Format:

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

2.8 Screenshots



```
postion and number to be entered: 3 7
[0, 6, 7, 0, 5, 0, 0, 0, 0]
0 6 7
0 5 0
0 0 0
press 2 for player2's chance:2
enter position:8
enter the number to be entered from list2:4
postion and number to be entered: 8 4
[0, 6, 7, 0, 5, 0, 0, 4, 0]
0 6 7
0 5 0
0 4 0
enter d for exit and f for continuef
press 1 for player1's chance  and press 2 for player2's chance:
```

3 Appendix

3.1 Code for ps1

```
def printIndex(str, s):
    flag = False;
    for i in range(len(str)):
        if (str[i:i + len(s)] == s):              #taking index of list
            print(i, end=" ");
            flag = True;
            x = list(str)
            x.insert((i + 3), "0")
            print(x)


bin1 = input("enter binary data:")  # Driver code taking input binary data as strin
a = list(bin1)                     #converting string into list
print(a)
```

```
b = a.count("1")                  #counting for number of ones
print(b)
if (b % 2 == 0):                  #checking for odd and even parity
    a.append(1)

else:
    a.append(0)
print(a)
bin2 = "010"
printIndex(bin1, bin2)               #stuffing 0 after each 010
p = a.append("0101")
print(p)
```

3.2 Code for ps2

```
print("welcome to the game")                         #tic tac toe game
list1 = [1, 3, 5, 7, 9]
list2 = [2, 4, 6, 8]
board = [0, 0, 0, 0, 0, 0, 0, 0, 0]
print(board[0], board[1], board[2])                   #showing blank tic tac toe t
print(board[2], board[3], board[4])
print(board[6], board[7], board[8])
while [True]:
    n = int(input("press 1 for player1's chance  and press 2 for player2's chance:")
    if (n == 1):
        print("player1's chgance")
        p1 = list1

        a = int(input("enter the position:"))                                  #taki
        b = int(input("enter the number to be entered from list1:"))
        print("postion and number to be entered: %d %d " % (a, b))

        board[a - 1] = b
        print(board)
        print(board[0], board[1], board[2])
        print(board[3], board[4], board[5])
        print(board[6], board[7], board[8])                                     #chec

        i = board[0] + board[1] + board[2]
        j = board[3] + board[4] + board[5]
        k = board[6] + board[7] + board[8]
        m = board[0] + board[3] + board[8]
        n = board[2] + board[3] + board[6]
    if(i==15):
        print ("player1 is winner")
```

```
elif(j==15):
    print ("player1 is winner")
elif(k==15):
    print ("player1 is winner")
elif(m==15):
    print ("player1 is winner")
elif(n==15):
    print ("player1 is winner")
p = int(input("press 2 for player2's chance:"))                    #option for p
if (p == 2):
    p2 = list2

    x = int(input("enter position:"))
    y = int(input("enter the number to be entered from list2:"))       #taking
    print("postion and number to be entered: %d %d " % (x, y))
    board[x - 1] = y
    print(board)
    print(board[0], board[1], board[2])
    print(board[3], board[4], board[5])
    print(board[6], board[7], board[8])

    i = board[0] + board[1] + board[2]
    j = board[3] + board[4] + board[5]
    k = board[6] + board[7] + board[8]
    m = board[0] + board[3] + board[8]
    n = board[2] + board[3] + board[6]
if (i == 15):
    print("player1 is winner")
elif (j == 15):                                                         #checking
    print("player1 is winner")
elif (k == 15):
    print("player1 is winner")
elif (m == 15):
    print("player1 is winner")
elif (n == 15):
    print("player1 is winner")


s= input("enter d for exit and f for continue")                        #options
if(s=="f"):
    continue
else:
    break
```

References

[1] github. *git*. https://github.com/.

[2] hacker rank. *python*. https://www.hackerrank.com/challenges/python-tuples/problem.

[3] .thecrazyprogrammer. *python*. https://www.thecrazyprogrammer.com/2017/04/python-program-add-two-numbers.html.

[4] toutorials point. *python*. https://www.tutorialspoint.com/python/index.htm.

[5] .tutorialspoint. *git toutorials*. https://www.tutorialspoint.com/git/index.htm.

[6] w3schools. *python*. https://www.w3schools.com/python/python_arrays.asp.

[7] youtube. *python*. https://www.youtube.com/watch?v=Blzp9iuhZqo.