ELP780

Software Lab

Aghil Sabu

2018EET2865

A report presented for the assignment on
Assignment 8 - Python and Github

Electrical Engineering
IIT Delhi
India
September 18, 2019

Contents

1 Problem Satement 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.

1.2 Input Format

Enter binary bit data that has to be transmitted.

1.3 Output Format

Print binary bit data with parity bit.
Print the modified string that is to be transmitted

1.4 Sample Input

010101110100101

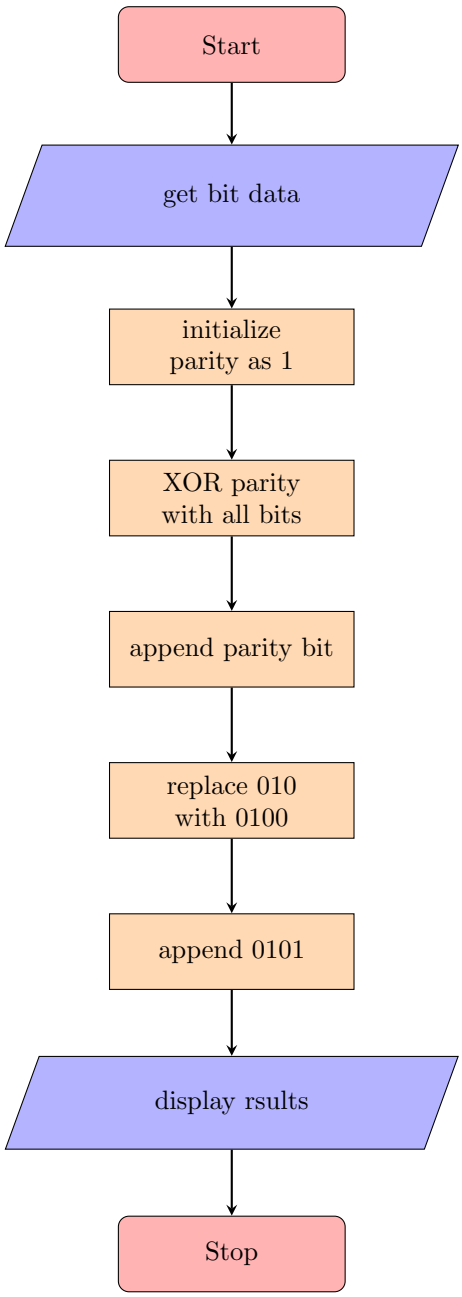
1.5 Sample Output

Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101

1.6 PS1 algorithm

- Read bit data from user
- Initialize parity as 1
- XOR parity with all bits to get parity value
- append parity bit
- replace pattern 010 with 0100
- append 0101 at the end
- display rsults in the required format

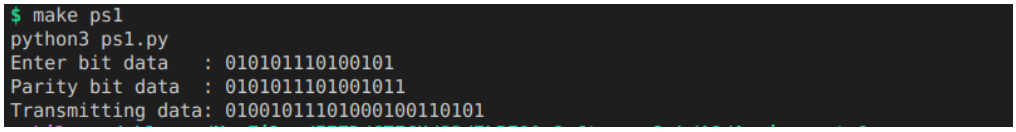
1.7 PS1 Flow Chart



1.8 PS1 Solution-Code

```
# reading bit data from user
data=input("Enter bit data   : ")
# initialising parity to 1 as initial parity value
parity=1
# finding parity of data stream
for i in data:
    parity=parity^int(i)
# adding parity bit to data
data+=str(parity)
print("Parity bit data  :",data)
# replacing 010 in data with 0100 to get data to be transmitted
trans=data.replace("010","0100")
# appending 0101 to the data to be transmitted
trans+="0101"
print("Transmitting data:",trans)
```

1.9 PS1 Output Screenshots



2 Problem Satement 2

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:

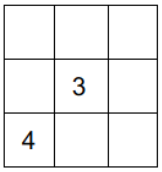
- Print 'Welcome to the Game!'.
- Print whether it is Player 1's or Player 2's chance.
- Get the position and number to be entered from the user.
- Show tic tac toe with data.
- Continue till the game gets draw or some player wins and show the result.
- Ask the user whether to continue for the next game or exit.

2.1 Sample Output:

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

	3	

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



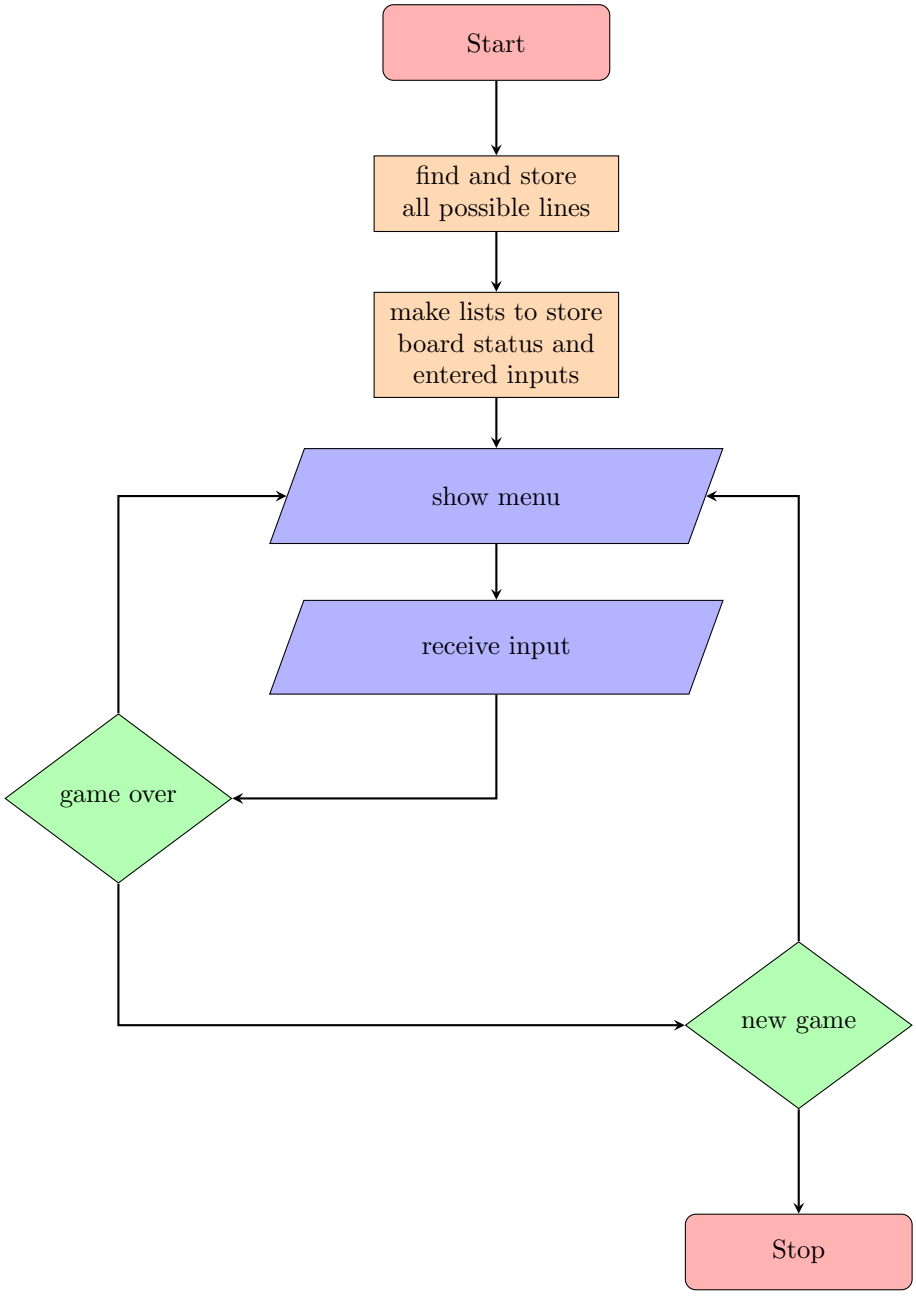
... Continue till game ends

Note – Must use at least one User Defined Function.

2.2 PS2 algorithm

- Save all possible lines to a 2D list lines
- make two 1D lists to store the board and the inputs already entered
- show the menu for the apropriate player
- receive players input
- check whether input is valid
- check whether game is over or is a draw
- if game is over, check if players wnat to play a new game

2.3 PS2 Flow Chart



2.4 PS2 Solution-Code

2.5 PS2 Output

2.5.1 input file

2.5.2 output file

3 makefile

3.1 makefile code

```
all: ps1 ps2
ps1:
python3 ps1.py
ps2:
python3 ps2.py

# clean:
#  rm *.out
```

3.2 makefile output

4 GIT

4.1 GIT Commit Screenshots

```
commit 7e5b94319220cb653e124c3478cdc291ac656c86
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 11:15:20 2019 +0530

    test program created for yaac

commit ff6181ee1be0b8eaaff9eede5cca037673e46e2c
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:26:45 2019 +0530

    added comments to PS1

commit cd567740072a48c352555b58aa2ecc315911356a
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:20:28 2019 +0530

    Ps1 basic structure created

commit 4a68b208805773de1eb656dff5b0c7bccafb44b7
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:51:29 2019 +0530

    ran a test program

commit beb6d2a5f7b16c9c6dec389db5bdae450a39257d
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:27:31 2019 +0530

    Initial Commit
```


```
commit 7e5b94319220cb653e124c3478cdc291ac656c86
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 11:15:20 2019 +0530

    test program created for yaac

commit ff6181ee1be0b8eaaff9eede5cca037673e46e2c
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:26:45 2019 +0530

    added comments to PS1

commit cd567740072a48c352555b58aa2ecc315911356a
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 10:20:28 2019 +0530

    Ps1 basic structure created

commit 4a68b208805773de1eb656dff5b0c7bccafb44b7
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:51:29 2019 +0530

    ran a test program

commit beb6d2a5f7b16c9c6dec389db5bdae450a39257d
Author: Aghil Sabu <aghilsabu@gmail.com>
Date:   Wed Sep 11 09:27:31 2019 +0530

    Initial Commit
```