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

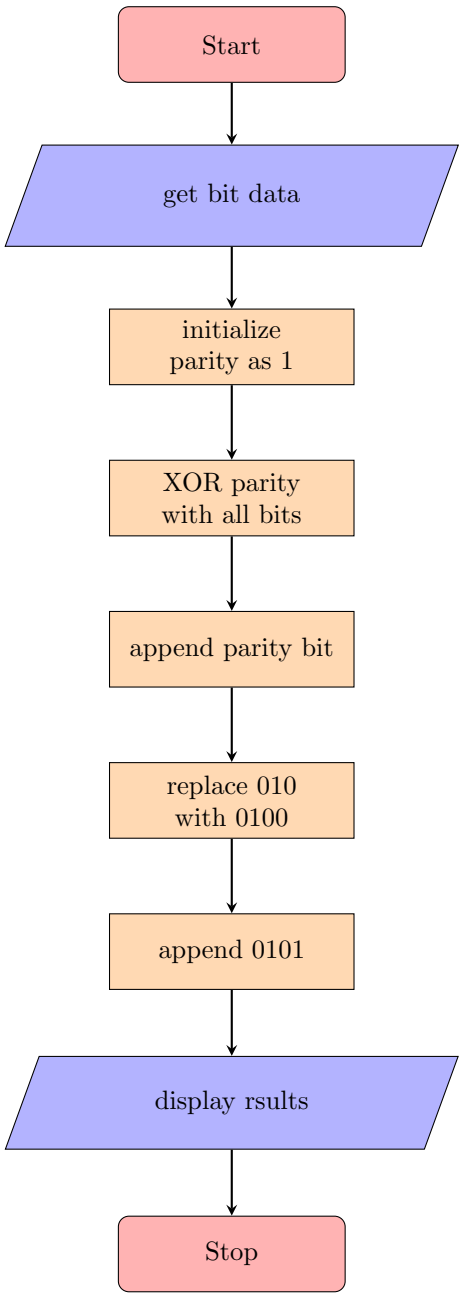
1.5 Sample Output

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

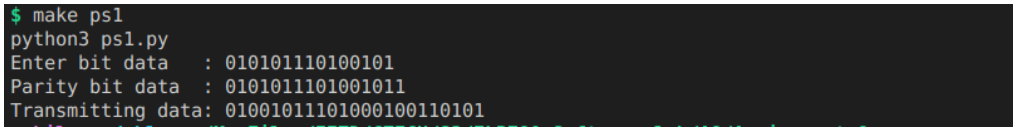
1.7 PS1 Flow Chart



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

2.1 Constraints

$1 <= Position <= 9$
$1 <= Number <= 9$

2.2 Terminal

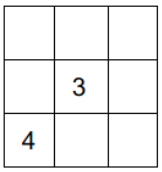
- Print 'Welcome to the Game!'.
- Print whether it is Player 1's or Player 2's chance.
- Get the position and number to be entered from the user.
- Show tic tac toe with data.
- Continue till the game gets draw or some player wins and show the result.
- Ask the user whether to continue for the next game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

2.3 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



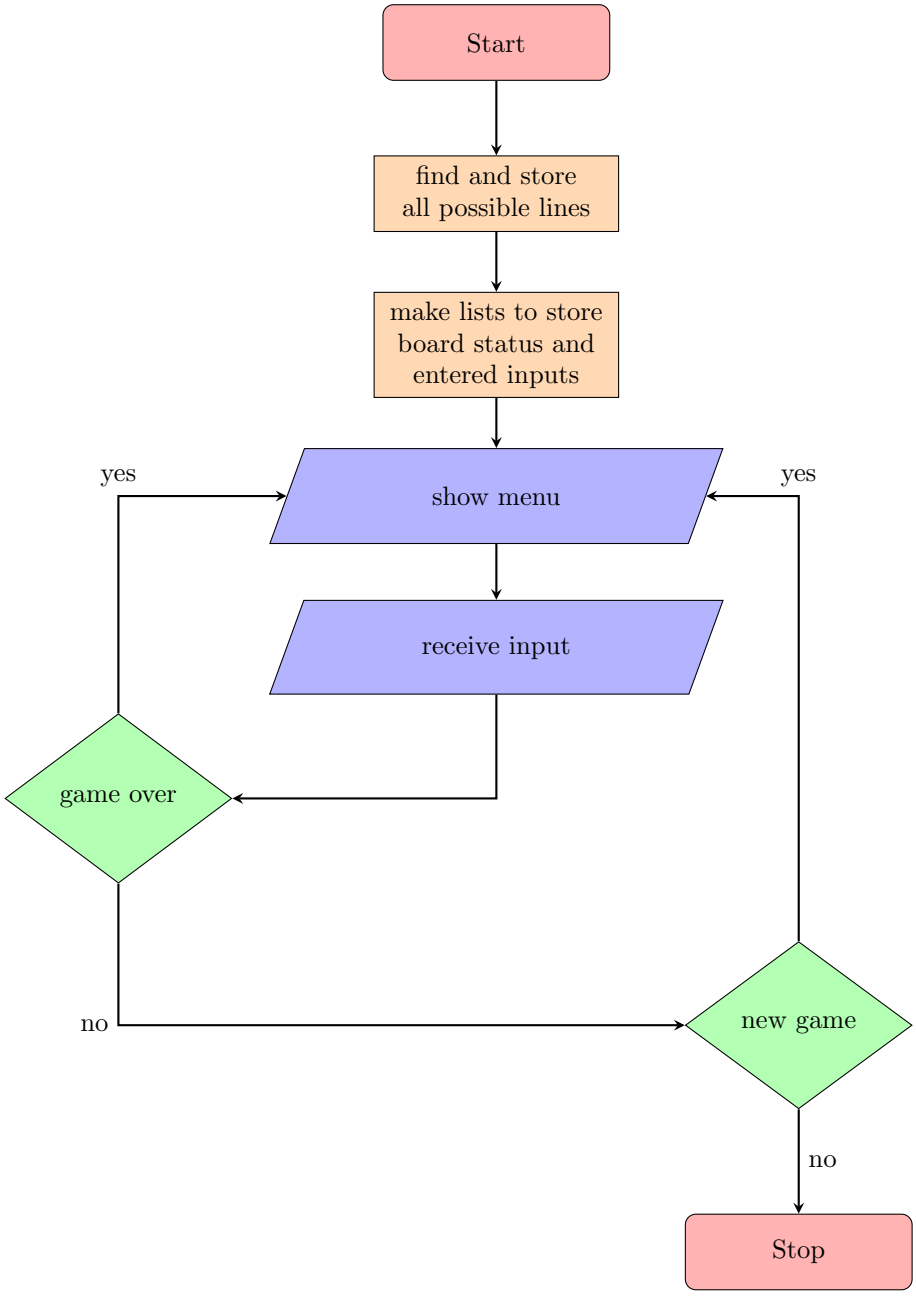
... Continue till game ends

Note – Must use at least one User Defined Function.

2.4 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

2.5 PS2 Flow Chart



2.6 PS2 Solution-Code

```
from os import system

size=3
winsum=15

# function to display all lines, if needed
def display(lines):
    for i in lines:
        for j in i:
            print(j,end=" ")
        print()

lines=[[]]
# adding all horizontal lines
for i in range(size):
    temp=[]
    for j in range(size):
        temp.append(i*size+j)
    lines.append(temp)
# adding all verticalal lines
for i in range(size):
    temp=[]
    for j in range(size):
        temp.append(i+size*j)
    lines.append(temp)
# adding diagonal lines
temp=[]
for i in range(size):
    temp.append(i+size*i)
lines.append(temp)
temp=[]
for i in range(size):
    temp.append(size*i+size-i-1)
lines.append(temp)
# display(lines)

# function to display board
def bdisplay(board):
    for i in range(size):
        print(end="\n\t")
        for j in range(size):
            if(board[i*size+j]==-1):
                print("_",end="\t")
            else:
```

```
                print(board[i*size+j],end="\t")
# board for playing the game
board=[]
for i in range(size*size):
    board.append(-1)
numbers=[]
for i in range(size*size):
    numbers.append(False)
# bdisplay(board)

# function to check whether game is over
def gameover(board,lines):
    for i in lines:
        sum=0
        for j in i:
            if(board[j]==-1):
                sum=0
                break
            else:
                sum+=board[j]
        if(sum==winsum):
            # for j in i:
            #     print(j,end=" ")
            # print()
            return True
    return False
# Starting the Game
player=0
while(1):
    # checking if a game is already over, then will show option to exit/continue
    if(player==size*size):
        system("clear")
        new=input("Do you want to start a new Game (y/n) : ")
        if(new=="y"):
            for i in range(size*size):
                board[i]=-1
                numbers[i]=False
            player=0
        elif(new=="n"):
            break
        else:
            continue
    # displaying game menus
    while(1):
        system("clear")
        print("\n\t Welcome to the Game!\n")
```

```
# Displaying the board
bdisplay(board)
print("\n\n\n\tplayer",(player%2)+1,"'s chance")
pos=input("Enter Position : ")
val=input("Enter value : ")
# checking whether inputs are valid
if(pos.isnumeric() and val.isnumeric()):
    pos=int(pos)
    val=int(val)
else:
    print("\n\tInvalid Input\n\nPress ENTER to continue")
    input()
    continue
if(pos<1 or val<1 or pos>size*size or val>size*size or board[pos-1]!=-1 or numbers[v
    print("\n\tInvalid Input\n\nPress ENTER to continue")
    input()
    continue
elif((player%2)==0 and val%2==0):
    print("\nERROR : Player 1 is only allowed to enter odd values")
    print("\nPress ENTER to continue")
    input()
    continue
elif((player%2)==1 and val%2!=0):
    print("\nERROR : Player 2 is only allowed to enter even values")
    print("\nPress ENTER to continue")
    input()
    continue
else:
    # updating the board with entered value
    board[pos-1]=val
    # checking if player has won the game
    if(gameover(board,lines)):
        print("\tPlayer",(player%2)+1,"wins the game")
        print("\nPress ENTER to continue")
        player=size*size
        input()
        break
    # checking if the board is completely filled
    elif(player==size*size-1):
        print("\tGame ends in a draw")
        print("\nPress ENTER to continue")
        input()
        player=size*size
        break
    # changing player
    else:
```

```
                numbers[val-1]=True
                player=player+1
```

2.7 PS2 Output Screenshots

```
         Welcome to the Game!


        _       _       _
        _       _       _
        _       _       _


        player 1 's chance
Enter Position :
```

```
         Welcome to the Game!


        _       _       _
        _       _       _
        _       _       _


        player 1 's chance
Enter Position : 1
Enter value : 2

ERROR : Player 1 is only allowed to enter odd values

Press ENTER to continue
```

```
         Welcome to the Game!


        1       2       3
        4       5       _
        _       _       _


        player 2 's chance
Enter Position : 6
Enter value : 6
        Player 2 wins the game

Press ENTER to continue
```

3 makefile

3.1 makefile code

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

# clean:
#  rm *.out
```

3.2 makefile output

input : make ps1

```
$ make all
lex ps1.l
gcc -o ps1.out lex.yy.c
./ps1.out ps1.txt
Rohit 101 50.500000
Virat 200 100.000000
Yuvi 52 52.000000
# yacc ps2.y
lex ps2.l
# cc lex.yy.c y.tab.c -o ps2.out
gcc -o ps2.out lex.yy.c
ps2.l: In function 'main':
ps2.l:31:30: warning: implicit declaration of function 'toupper' [-Wimplicit-function-declaration]
        }
                             ^
./ps2.out ps2.txt output.txt
```

input : make ps2

```
          Welcome to the Game!


         _       _       _
         _       _       _
         _       _       _


         player 1 's chance
 Enter Position :
```

4 Git and Github

4.1 Github Account

Github Account Link : $https://github.com/2018EET2865/Assignment_8.git$

4.2 GIT Commit Screenshots

```
$ make ps1
python3 ps1.py
Enter bit data   : 010101110100101
Parity bit data  : 0101011101001011
Transmitting data: 01001011101000100110101
```

```
$ make ps1
python3 ps1.py
Enter bit data   : 010101110100101
Parity bit data  : 0101011101001011
Transmitting data: 01001011101000100110101
```

References

[1] Tutorialspoint python
https://tutorialspoint.com/python/index.html

[2] Atlassian Git tutorial *https://www.atlassian.com/git/tutorials/*