Telecommunication Software Lab

ELP 718

Assignment 6

Name: **Chanchal Kumar**

Entry Number: **2017JTM2018**

September 12, 2017

INDIAN INSTITUTE OF TECHNOLOGY
NEW DELHI

Contents

1 Problem Statement 1:

You have to do parity check and bit stuffing to make a frame in this problem. 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 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 end of frame using bit oriented approach. It uses a special string of bits, called a flag for both idle fill and to indicate the beginning and the ending of frames. The string 0101 is used as the bit string or flag to indicate the end of the frame. The bit stuffing rule is to insert a 0 after each appearance of 010 in the original data. In addition, if the frame ends in 01, a 0 would be stuffed after the 1st 0 in the actual terminating string 0101.

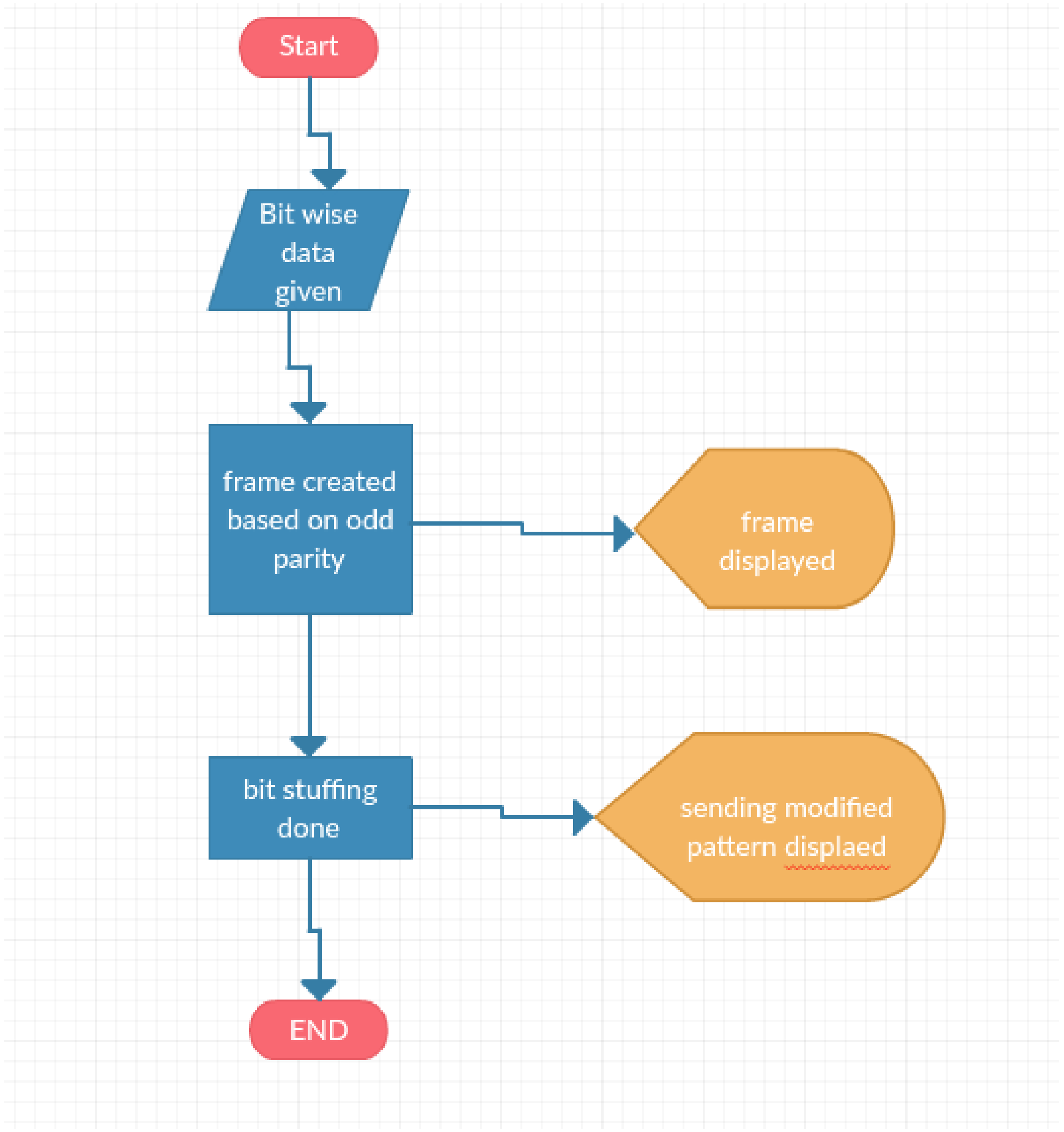
1.1 Assumptions

Parity check is following odd parity check.

1.2 Program Structure

- In first line of the program You have to enter Enter binary bit data which has to be transmitted.
- Program will Print binary bit data with parity bit.
- Then Print the modified string received at the other end.

1.3 Graphical Structure of Code


Start
Bit wise data given
frame created based on odd parity
frame displayed
bit stuffing done
sending modified pattern displaed
END

1.4 Algorithm and Implementation

For parity check i have counted number of 1 , if it is even frame is concatenated with 1 otherwise with 0.

1.5 Input and Output format

Input format:

Enter binary bit data which has to be transmitted.

Output format:

Print binary bit data with parity bit. Print the modified string received at the other end.

1.6 Test cases

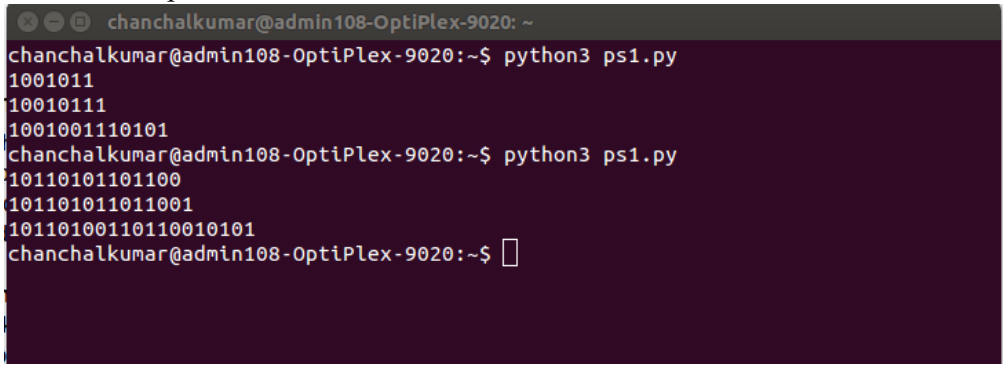
- Case 1: input 1001011 Frame with parity bit = 10010111 Modified string = 1001001110101
- Case 2: Input 10110101101100 Frame with parity bit = 101101011011001 Modified string received= 1011010011100100101

1.7 Difficulties/Issues faced

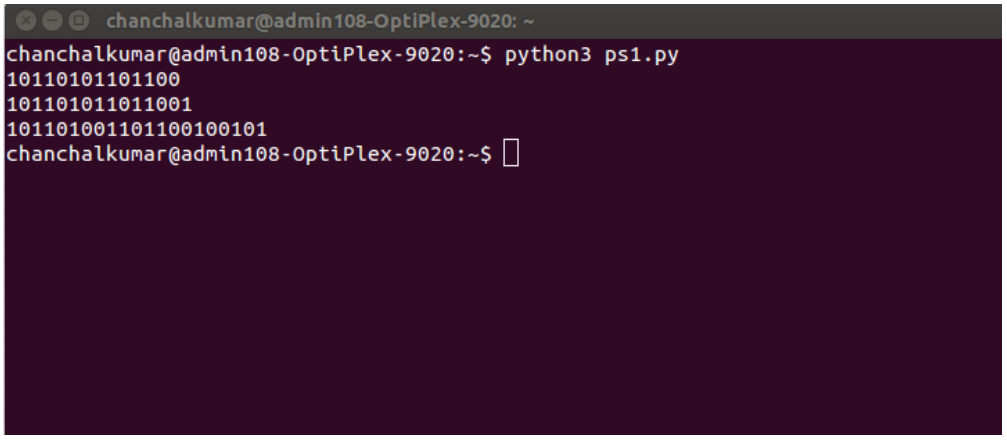
difficulty faced in bit stuffing while consecutive 010101 pattern arrives. in this it will bit stuff as 0100101 as program is searching and replacing for 010.

1.8 Screenshots

Test Cases Output Screenshots:



```
chanchalkumar@admin108-OptiPlex-9020: ~
chanchalkumar@admin108-OptiPlex-9020:~$ python3 ps1.py
1001011
10010111
1001001110101
chanchalkumar@admin108-OptiPlex-9020:~$ python3 ps1.py
10110101101100
101101011011001
10110100110110010101
chanchalkumar@admin108-OptiPlex-9020:~$ 
```



```
chanchalkumar@admin108-OptiPlex-9020: ~
chanchalkumar@admin108-OptiPlex-9020:~$ python3 ps1.py
10110101101100
101101011011001
1011010011101100100101
chanchalkumar@admin108-OptiPlex-9020:~$ 
```

2 Problem Statement 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 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 start 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 cell might be necessary to complete a different line. Note – Line can be horizontal, vertical or diagonal

Constraints:

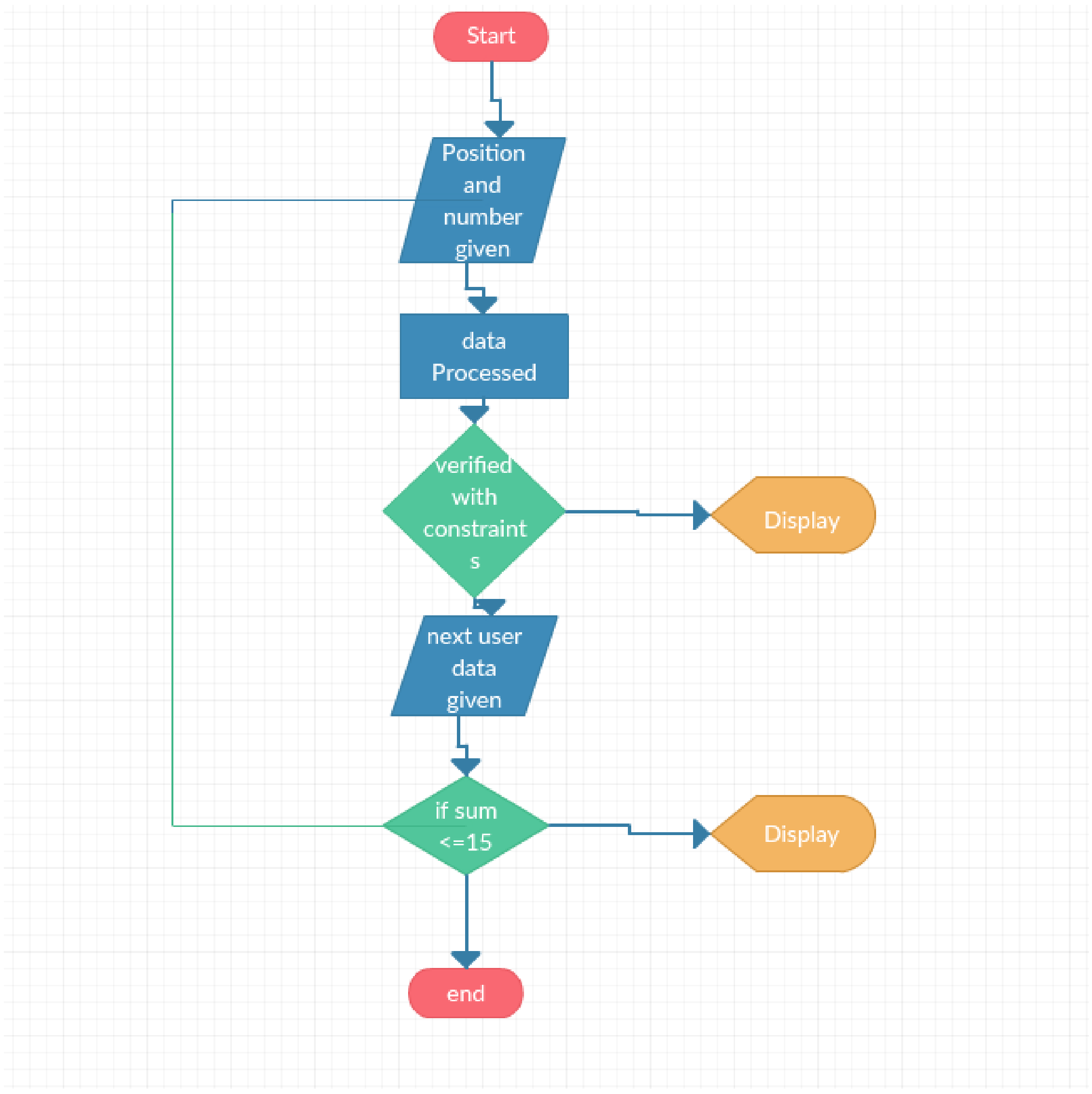
- position between 1 and 9
- number between 1 and 9
- sum between 1 and 15

2.1 Assumptions

2.2 Program Structure

- Welcome priented
- Position and number entered from user
- tic tac tow data showed
- continue till wining.

2.3 Graphical Structure of Code



2.4 Algorithm and Implementation

2.5 Input and Output format

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 user.
- Show tic tac toe with data.
- Continue till the game gets draw or some player wins and show result.

- Ask user whether to continue for next game or exit.

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

2.6 Test cases

- Case1: Enter the position and number to be entered: 5,3

2.7 Difficulties/Issues faced

difficulty in running program.

2.8 Screenshots

Test Cases Output Screen-shots:

References

[1] www.tutorialspoint.com/python

[2] https://www.programiz.com/python-programming

bibitemgeeksweb https://readwrite.com/2013/09/30/understanding-github-a-journey-for-beginners-part-1/

[3] http://www.linuxhowtos.org/

[4] https://inventwithpython.com/chapter10.html

Annexture

Source code for PS1

```
###### Assignment 1 parity check and framing .py file ###########

####### write your code here ##########

def parity(x):        # function for parity bit
    z=x.count('1')
    if z%2==1:
        return 0
    else:
        return 1

x= input()
x=str(x)
z=parity(x)
z=str(z)
frame=(x+z)
print(frame)    #frame after parity bit addition

m=frame.replace('010','0100')  #bit stuffing
u=m[-2:]    #for accessing last 2 character
d='01'
d=str(d)
if u == d:
    m=m+'0'

final= m+'0101'    #modified string received at other end
print(final)




###### this is the second .py file ###########

####### write your code here ##########


#  board
board= [0,1,2,
        3,4,5,
        6,7,8]

def disp():
    print board[0],'|',board[1],"|",board[2]
    print "-----------"
    print board[3],'|',board[4],"|",board[5]
    print "———————"
    print board[6],'|',board[7],"|",board[8]

def isWinner(bo, Player1):
   # Given a board and a player's letter, this function returns True if that playe
   # We use bo instead of board and le instead of letter so we don't have to type
    return ((bo[7] + bo[8] + bo[9]) >=15) or # across the top
    ((bo[4]+ bo[5] + bo[6]) >=15) or # across the middle
    ((bo[1] + bo[2] + bo[3]) >=15 ) or # across the bottom
    ((bo[7] +bo[4]+bo[1]) =>15) or # down the left side
    ((bo[8] + bo[5] + bo[2] )=>15) or # down the middle
    ((bo[9] + bo[6] + bo[3] )=>15) or # down the right side
    ((bo[7] +  bo[5] + bo[3] )=>15) or # diagonal
    ((bo[9] + bo[5] + bo[1] )=>15)) # diagonal
def playAgain():
   # This function returns True if the player wants to play again, otherwise it re
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')

while True:
    print "Welcome to the Game"
    position=
```

source code for ps2

```
###### this is the second .py file ###########

####### write your code here ##########


#  board
board= [0,1,2,
        3,4,5,
        6,7,8]

def disp():
    print board[0],'|',board[1],"|",board[2]
    print "———————"
    print board[3],'|',board[4],"|",board[5]
    print "———————"
    print board[6],'|',board[7],"|",board[8]

def isWinner(bo, Player1):
   # Given a board and a player's letter, this function returns True if that playe
   # We use bo instead of board and le instead of letter so we don't have to type
    return ((bo[7] + bo[8] + bo[9]) >=15) or # across the top
    ((bo[4]+ bo[5] + bo[6]) >=15) or # across the middle
    ((bo[1] + bo[2] + bo[3]) >=15 ) or # across the bottom
    ((bo[7] +bo[4]+bo[1]) =>15) or # down the left side
    ((bo[8] + bo[5] + bo[2] )=>15) or # down the middle
    ((bo[9] + bo[6] + bo[3] )=>15) or # down the right side
    ((bo[7] +  bo[5] + bo[3] )=>15) or # diagonal
    ((bo[9] + bo[5] + bo[1] )=>15)) # diagonal
def playAgain():
    # This function returns True if the player wants to play again, otherwise it re
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')

def getBoardCopy(board):
    # Make a duplicate of the board list and return it the duplicate.
    dupeBoard = []

    for i in board:
        dupeBoard.append(i)

    return dupeBoard

def isSpaceFree(board, move):
    # Return true if the passed move is free on the passed board.
    return board[move] == ' '

def getPlayerMove(board):
    # Let the player type in his move.
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(mo
        print('What is your next move? (1-9)')
        move = input()
    return int(move)

def chooseRandomMoveFromList(board, movesList):
    # Returns a valid move from the passed list on the passed board.
    # Returns None if there is no valid move.
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)

    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None
while True:
    print "Welcome to the Game"
    position=
```