Indian Institute of Technology
Delhi

ELP 718

TELECOM SOFTWARE LAB

Submitted By:
NAME: MOHIT VARSHNEY
ENTRY NO.: JTM172019
DATE: 12-09-2017
ASSIGNMENT NO.: 6

Contents

1 Problem Statement 1 **1**
1.1 Assumptions 1
1.2 Program Structure 1
1.3 Input and Output Format 1
1.4 Test Cases 1
1.5 Screenshots 2

2 Problem Statement 2 **3**
2.1 Assumptions 3
2.2 Program Structure 3
2.3 Input and Output Format 3
2.4 Screenshots 4

3 Bibliography **4**

4 Annexure **5**

1 Problem Statement 1

Design a parity check and bit oriented framing program such that odd parity is maintained and parity bit is added in the end of the message and 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

Data to be entered in Binary format

1.2 Program Structure

1. Enter a data which is to be transmitted.
2. Calculate numbers of 1 in data and add parity bit maintaining the odd parity.
3. Look for sequence 010 in original data and insert 0 whenever it occurs.
4. If frame ends with 01 then add 0 in end and then terminate all strings with 0101

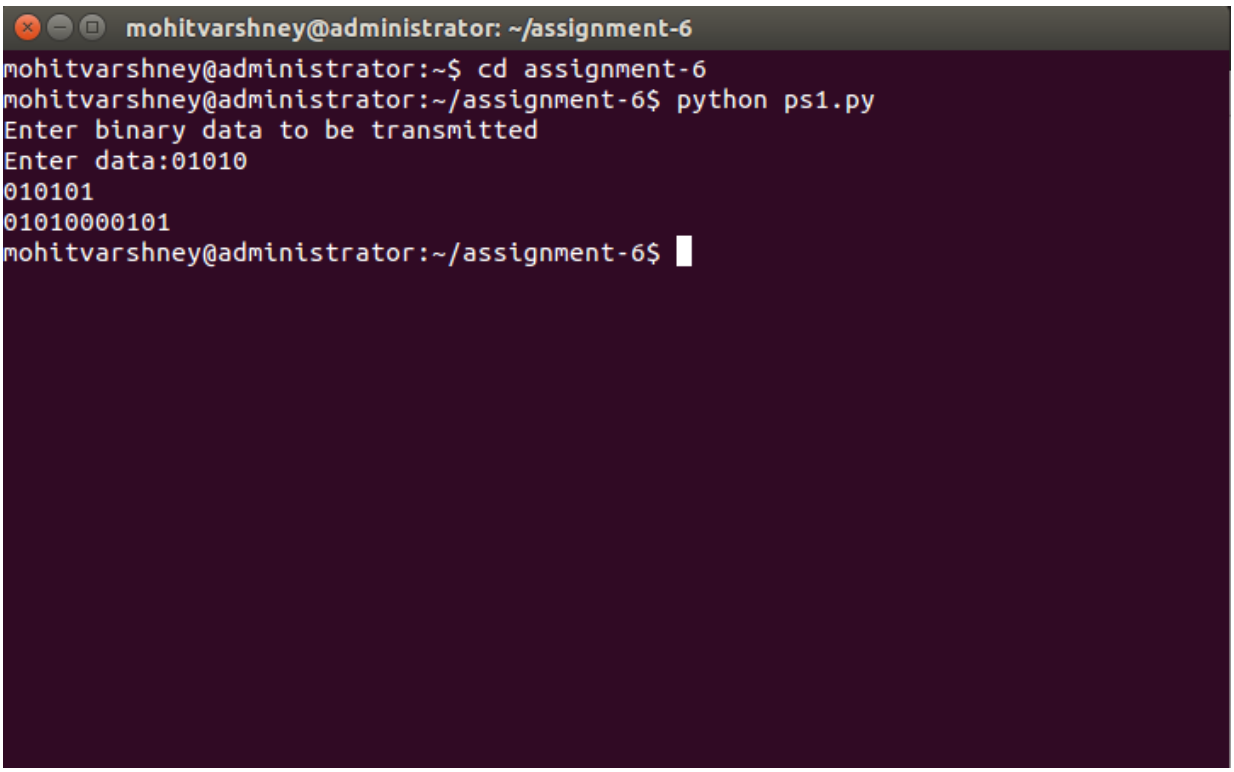
1.3 Input and Output Format

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.

1.4 Test Cases

Sample Input
01010
Sample Output
010101
0100100100101

1.5 Screenshots



2 Problem Statement 2

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 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.

2.1 Assumptions

Line can be horizontal, vertical or diagonal
$1 \leq$ Position ≤ 9
$1 \leq$ Number ≤ 9
$1 \leq$ Sum ≤ 15

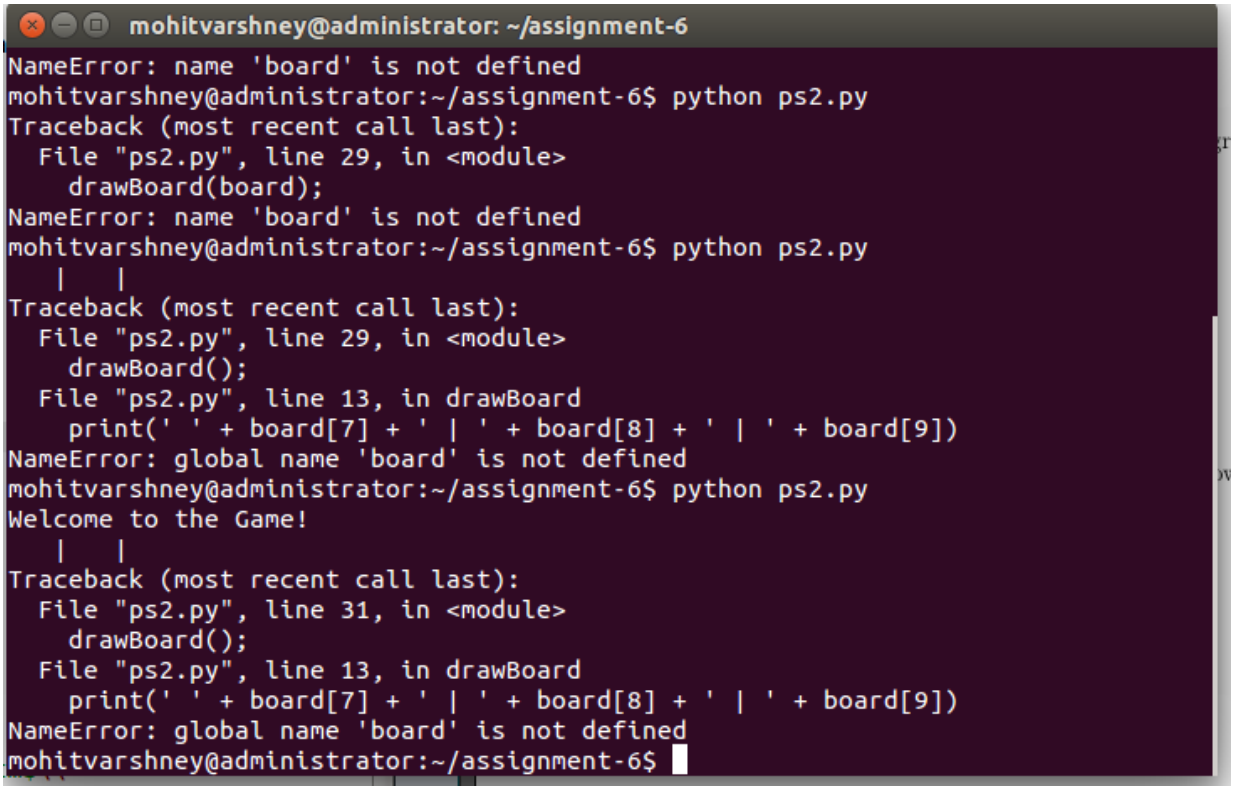
2.2 Program Structure

1. First write a paragraph in a text file.
2. write program to count number of characters in a paragraph
3. Find maximum and minimum occuring characters
4. Change cases of maximum occuring character

2.3 Input and Output Format

Print Welcome to the Game!.
Print whether it is Player 1s or Player 2s 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.

2.4 Screenshots



```
mohitvarshney@administrator: ~/assignment-6
NameError: name 'board' is not defined
mohitvarshney@administrator:~/assignment-6$ python ps2.py
Traceback (most recent call last):
  File "ps2.py", line 29, in <module>
    drawBoard(board);
NameError: name 'board' is not defined
mohitvarshney@administrator:~/assignment-6$ python ps2.py
   |   |
Traceback (most recent call last):
  File "ps2.py", line 29, in <module>
    drawBoard();
  File "ps2.py", line 13, in drawBoard
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
NameError: global name 'board' is not defined
mohitvarshney@administrator:~/assignment-6$ python ps2.py
Welcome to the Game!
   |   |
Traceback (most recent call last):
  File "ps2.py", line 31, in <module>
    drawBoard();
  File "ps2.py", line 13, in drawBoard
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
NameError: global name 'board' is not defined
mohitvarshney@administrator:~/assignment-6$
```

3 Bibliography

https://www.tutorialspoint.com/python/python_lists.htm
https://www.tutorialspoint.com/python/python_oops.htm

4 Annexure

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

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

print("Enter binary data to be transmitted")

#taking input of binary data to be transmitted
n=raw_input("Enter data:")
count=0
count1=0
index=0

#getting number of bits in data

l=len(n)

#finding number of 1's in data
for i in range(0,l):
    if n[i]=='1':
        count=count+1
#adding parity bit
if count%2==0:
    msg_parity=n[:l]+'1'
else:
    msg_parity=n[:l]+'0'
print msg_parity

#checking message for 010 sequence
l1=len(msg_parity)
#print l1
#for i in range(0,l1):
#   if msg_parity[i:i+2]=='010':
#       count1=count1+1
#print count1


string='010'
for i in range(0,l1):
    if msg_parity[i:i+3]==string:
        index=i+3
#adding 0 after 010
    msg1_parity=msg_parity[:index]+'0'+msg_parity[index+1:]
#print msg1_parity

#to add 0 if the frame ends with 01
string1='01'
if msg_parity[l1-2:]==string1:
    msg2_parity=msg1_parity[:len(msg1_parity)]+'0'
#print msg2_parity
```

```python

#Adding flag 0101 at the end of frame

final_string=msg2_parity[:len(msg2_parity)]+'0101'

print final_string
```

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

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

# Tic Tac Toe
import random
def drawBoard():

# This function prints out the board that it was passed.
# "board" is a list of 10 strings representing the board (ignore index 0)

 print('   |   |')
 print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])

 print('   |   |')
 print('-----------')

 print('   |   |')
 print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])

 print('   |   |')
 print('-----------')

 print('   |   |')
 print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])

 print('   |   |')


print("Welcome to the Game!")
print("Player 1's chance")
drawBoard();

def inputPlayerLetter():
 # Lets the player type which letter they want to be.
 # Returns a list with the player s letter as the first item, and the computer's

 letter = ''
 while not (letter == 'X' or letter == 'O'):
    print('Do you want to be X or O?')
        letter = input().upper()

 # the first element in the list is the player s letter, the second is the comp
```

```
if letter == 'X':
    return ['X', 'O']
else:
    return ['O', 'X']

def whoGoesFirst():

# Randomly choose the player who goes first.

if random.randint(0, 1) == 0:
     return 'computer'
else:
     return 'player'

def playAgain():
# This function returns True if the player wants to play again, otherwise it retu

print('Do you want to play again? (yes or no)')
return input().lower().startswith('y')

def makeMove(board, letter, move):
board[move] = letter

def isWinner(bo, le):
# 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] == le and bo[8] == le and bo[9] == le) or # across the top
(bo[4] == le and bo[5] == le and bo[6] == le) or # across the middle
(bo[1] == le and bo[2] == le and bo[3] == le) or # across the bottom
(bo[7] == le and bo[4] == le and bo[1] == le) or # down the left side
(bo[8] == le and bo[5] == le and bo[2] == le) or # down the middle
(bo[9] == le and bo[6] == le and bo[3] == le) or # down the right side
(bo[7] == le and bo[5] == le and bo[3] == le) or # diagonal
(bo[9] == le and bo[5] == le and bo[1] == le)) # diagonal

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 their move.
move = ' '
```

```
 while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move
    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

 def getComputerMove(board, computerLetter):

 # Given a board and the computer's letter, determine where to move and return tha

 if computerLetter == 'X':
      playerLetter = 'O'
 else:
      playerLetter = 'X'

# Here is our algorithm for our Tic Tac Toe AI:
# First, check if we can win in the next move

for i in range(1, 10):

    copy = getBoardCopy(board)
    if isSpaceFree(copy, i):
        makeMove(copy, computerLetter, i)
        if isWinner(copy, computerLetter):
            return i

# Check if the player could win on their next move, and block them.
for i in range(1, 10):
    copy = getBoardCopy(board)
    if isSpaceFree(copy, i):
        makeMove(copy, playerLetter, i)
    if isWinner(copy, playerLetter):
         return i
 # Try to take one of the corners, if they are free.

move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
        if move != None:
        return move
# Try to take the center, if it is free.
```

```
        if isSpaceFree(board, 5):

        return 5
```