Assignment-6
ELP - 718 Telecom Software Laboratory

Shefali Gupta

September 12, 2017

Entry No. - 2017JTM2767

Bharti School
Telecommunication Technology and Management
IIT DELHI
India

Contents

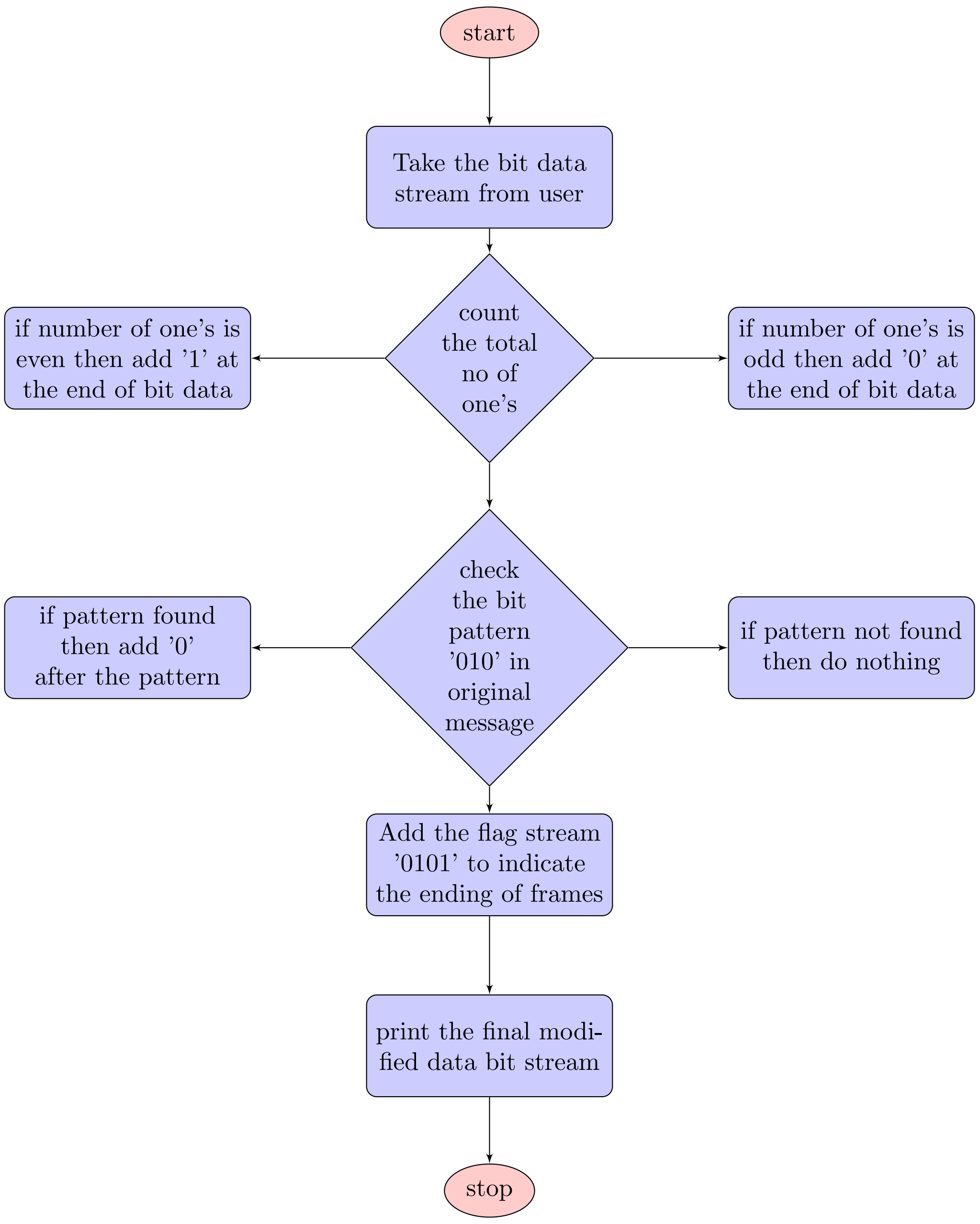
Problem Statement 1
.1 Program Structure 1
.2 Algorithm and Implementation 1
.3 Input and Output Format 2
.4 Test Cases 2
.5 Screenshots 2
Problem Statement-2 2
.1 Assumptions 2
.2 Program Structure 3
.3 Algorithm and Implementation 4
.4 Input and Output Format 4
.5 Test Cases 4
.6 Screenshots 4
Reference 4
Source code 5
.1 Source Code -1 5

Problem Statement

Parity Check Take the input bit stream and append a single bit, called a parity check. 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 Program Structure


start
Take the bit data stream from user
count the total no of one's
if number of one's is even then add '1' at the end of bit data
if number of one's is odd then add '0' at the end of bit data
check the bit pattern '010' in original message
if pattern found then add '0' after the pattern
if pattern not found then do nothing
Add the flag stream '0101' to indicate the ending of frames
print the final modified data bit stream
stop

.2 Algorithm and Implementation

- Take the input of data bits from user.
- Check the odd parity and add the parity bit at the end of bit stream.

- Check the bit pattern ’010’ in the updated message bits, if pattern found then add the ’0’ at the end of pattern.
- Check the last two bits of updated data bits, if frame ends in ’01’ then a ’0’ would be added after the 1st 0 in the actual terminating string 0101.
- At the end of frame add the flag ’0101’ to indicate the end of frame.

.3 Input and Output Format

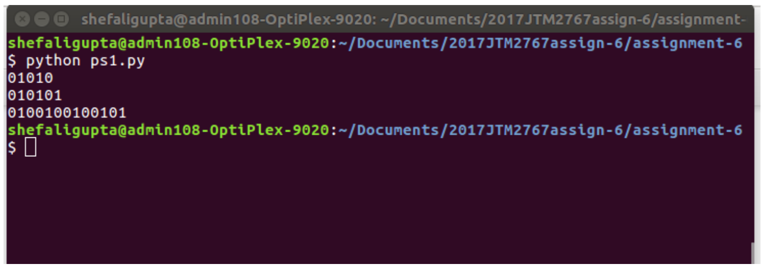
Input Take the input - 01010

Output First output after adding parity bit - 010101
Second output after bit stuffing - 0100100100101

.4 Test Cases

- Take the input data bit from user - 01010
- Check the odd parity. Number of 1’s is even then output should be - 010101
- check the pattern ’010’. output should be - 0100100100101

.5 Screenshots


shefaligupta@admin108-OptiPlex-9020: ~/Documents/2017JTM2767assign-6/assignment-

```
shefaligupta@admin108-OptiPlex-9020:~/Documents/2017JTM2767assign-6/assignment-6
$ python ps1.py
01010
010101
0100100100101
shefaligupta@admin108-OptiPlex-9020:~/Documents/2017JTM2767assign-6/assignment-6
$ 
```

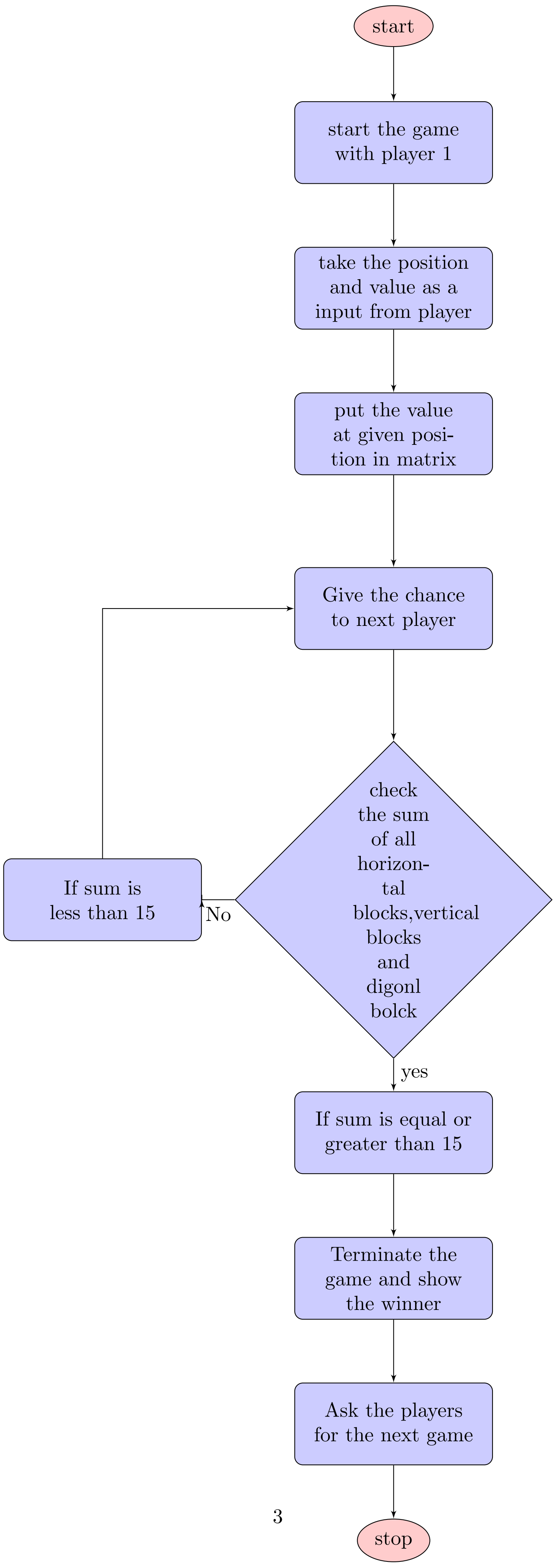
Problem Statement-2

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

.1 Assumptions

Assume that player 1 has all odd numbers and player 2 has all even numbers.

.2 Program Structure


start
start the game with player 1
take the position and value as a input from player
put the value at given position in matrix
Give the chance to next player
check the sum of all horizontal blocks,vertical blocks and digonl bolck
No
If sum is less than 15
yes
If sum is equal or greater than 15
Terminate the game and show the winner
Ask the players for the next game
stop

.3 Algorithm and Implementation

- Start the game with '0' at all position in matrix.
- Take the input of position and value from player.
- Put the value in matrix at given position.
- Give the chance to next player.
- And do same as defined in step 2 and 3.
- Calculate the sum of all horizontal blocks,vertical blocks and digonl bolck.
- If sum is less than 15 then continue the game.
- If sum is equal or greater than 15 then terminate the game and display the winner.
- Ask for next game to the players, if they want to play then repeat all above steps.

.4 Input and Output Format

Input Take the input - position,value
Enter the position and number to be entered from user
Ex. - 5,3

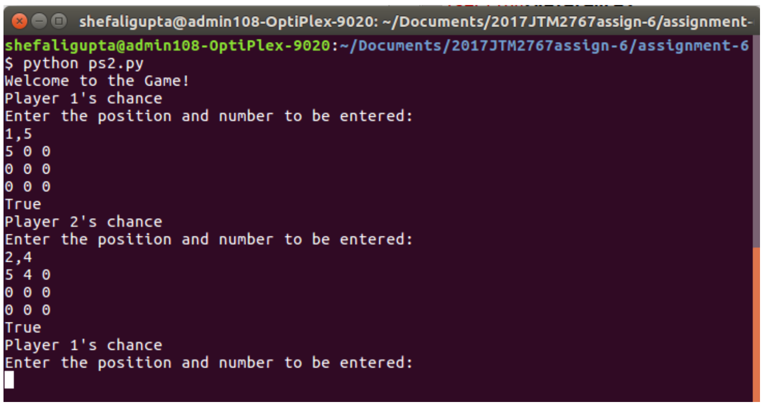
Output Output will look like this:

```
0 0 0
0 3 0
0 0 0
```

.5 Test Cases

- Take the position and number as input from user.
- Insert the number at given position in matrix.
- if sum of any horizontal block, vertical block and digonal block is greater than or equal 15 then game should be terminate.
- if sum is less than 15 then next player chance should come.

.6 Screenshots


shefaligupta@admin108-OptiPlex-9020: ~/Documents/2017JTM2767assign-6/assignment-

```
shefaligupta@admin108-OptiPlex-9020:~/Documents/2017JTM2767assign-6/assignment-6
$ python ps2.py
Welcome to the Game!
Player 1's chance
Enter the position and number to be entered:
1,5
5 0 0
0 0 0
0 0 0
True
Player 2's chance
Enter the position and number to be entered:
2,4
5 4 0
0 0 0
0 0 0
True
Player 1's chance
Enter the position and number to be entered:
```

Reference

Bibliography

[1] C Basic tutorial
http://www.cprogramming.com/tutorial/c-tutorial.html
http://www.tutorialspoint.com/cprogramming/
http://www.ime.usp.br/ pf/Kernighan-Ritchie/C-Programming-Ebook.pdf

[2] Vedio tutorial
https://www.youtube.com/watch?v=Yq6XFl-u00o
https://www.youtube.com/watch?v=sCtY–xRUyI
https://www.youtube.com/watch?v=xQ0ONbt-qPs

[3] GDB
https://www.cs.umd.edu/ srhuang/teaching/cmsc212/gdb-tutorial-handout.pdf

Source code

.1 Source Code -1

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

####### write your code here ##########
from collections import Counter

n = raw_input()    #take the input of bit stream

list1 = list(n)     #convert it into list

noOfones = list1.count('1')

rem = noOfones%2

if(rem==0):
    list1.append('1')

else:
    list1.append('0')

ubit = ''.join(list1)
print  ubit  #print the bit data with parity bit

#bit stuffing.......

sub = '010'

# To find the sub string into given string

def find_substring(string,sub_string):
        # str1 = string
          for i in range(len(string)):
          if(string[i]==sub_string[0]):

               flag = string.find(sub_string, i, i+len(sub_string))
```

```
            if(flag>=0):
            string = string[:i+len(sub_string)]+'0'+string[i+len(sub_string):]
        return string


ustr = find_substring(ubit,sub)

lbits = ustr[len(ustr)-2:len(ustr)]

if(lbits=='01'):
    ustr = ustr+'0'

#print lbits
# final modified string received at the other end
print ustr+'0101'
```

.2 Source Code - 2

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

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


print "Welcome to the Game!"


#matrix = "\n".join([" ".join(["0" for x in range(3)]) for x in range(3)])

#list1 = list(matrix)

player1 = [1,3,5,7,9]
player2 = [2,4,6,8]


list1 = [0,0,0,0,0,0,0,0,0]



def play_game(lst):
    flag = "True"
    rem = lst[0]%2
    #rem2 = lst2[0]%2
    if(rem!=0):
        print "Player 1's chance"
    else:
        print "Player 2's chance"

    print "Enter the position and number to be entered:"
    x,y = map(int, raw_input().split(','))
    if(x>=1 and x<=9):
        if(y>=1 and y<=9):
            list1[x-1]=y
            lst.remove(y)

    for i in xrange(0,9,3):
        print (str(list1[i])).ljust(2)+(str(list1[i+1])).ljust(2)+(str(list1[i+2])).ljust(2)

    h1 = list1[0]+list1[1]+list1[2]
    h2 = list1[3]+list1[4]+list1[5]
    h3 = list1[6]+list1[7]+list1[8]

    v1 = list1[0]+list1[3]+list1[6]
    v2 = list1[1]+list1[4]+list1[7]
    v3 = list1[2]+list1[5]+list1[8]
    d1 = list1[0]+list1[4]+list1[8]
    d2 = list1[2]+list1[4]+list1[6]

    summ = [h1,h2,h3,v1,v2,v3,d1,d2]
    for i in range(len(summ)):
        if(summ[i]>=15):
            print summ
            flag="False"
            break

    return flag





flag = "True"
while(flag=="True"):

    flag = play_game(player1)
    print flag
        flag = play_game(player2)
    print flag


print "Want to play another game"

resp = raw_input()

if(resp=="yes"):

    while(flag=="True"):

        flag = play_game(player1)
        print flag
        flag = play_game(player2)
        print flag


#for i in range(3):
#    print (str(list1[i])).ljust(2)+(str(list1[i+1])).ljust(2)+(str(list1[i+2])).ljust(2)
```