

Assignment-8

ELP-780 Software Lab

Naina Mehta

2018EET2559

2019-20

A report presented for the assignment Python and GitHub



Computer Technology
Department of Electrical Engineering
IIT DELHI
India
September 18, 2019

Contents

| | | |
|----------|--|-----------|
| 1 | Problem Statement-1 | 1 |
| 1.1 | Problem Statement | 1 |
| 1.2 | Assumptions | 1 |
| 1.3 | Algorithm and Implementation | 1 |
| 1.4 | Input and Output format | 2 |
| 1.5 | Program Structure | 2 |
| 1.6 | Test cases | 3 |
| 1.7 | Difficulties/Issues faced | 3 |
| 1.8 | Screenshots | 3 |
| 2 | Problem Statement-2 | 4 |
| 2.1 | Problem Statement | 4 |
| 2.2 | Assumptions | 4 |
| 2.3 | Algorithm and Implementation | 4 |
| 2.4 | Program Structure | 5 |
| 2.5 | Input and Output format | 6 |
| 2.6 | Test cases | 6 |
| 2.7 | Difficulties/Issues faced | 6 |
| 2.8 | Screenshots | 6 |
| | Appendices | 9 |
| A | Code for Problem Statement 1 | 9 |
| B | Code for Problem Statement 2 | 11 |
| C | Git Screenshots | 15 |

1 Problem Statement-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 1s in the bit string is even and has the value 0 otherwise, i.e., Odd Parity Check.

- **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 Assumptions

- Python3 is installed on the system.
- Bit string has to be entered by the user.

1.3 Algorithm and Implementation

- Read Bit String from user.
- Check for number of ones in the string.
- Append appropriate parity check bit.
- Search for 010 pattern in the string and store the corresponding indices
- Stuff bit 0 at the appropriate position
- Append the string with 0101 to denote end of the string

1.4 Input and Output format

- **Input Format**
Enter binary bit data that has to be transmitted.
- **Output Format**
Print binary bit data with parity bit.
Print the modified string that is to be transmitted

1.5 Program Structure

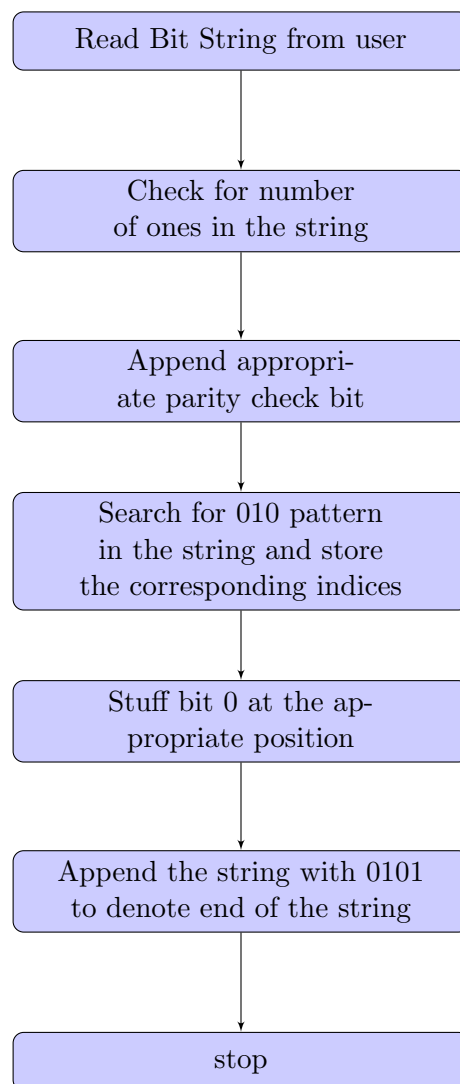


Figure 1.1: Flow Chart PS1

1.6 Test cases

The code was test for the following cases:

- 010101110100101
- 010010010

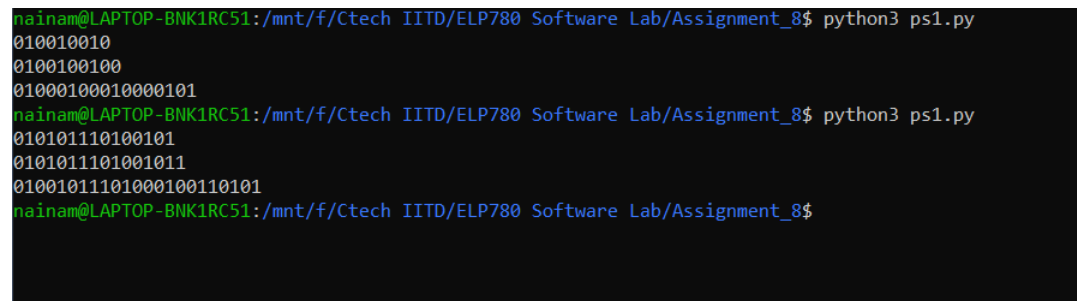
Output for the test cases have been shown in the scree-shot section.

1.7 Difficulties/Issues faced

- The position to insert the 0 bit after 010 pattern has been observed should be identified correctly.
- Appropriate data types should be used for various functions.

1.8 Screenshots

Following are the screenshots



```
nainam@LAPTOP-BNK1RC51:/mnt/f/Ctech IITD/ELP780 Software Lab/Assignment_8$ python3 ps1.py
010010010
0100100100
01000100010000101
nainam@LAPTOP-BNK1RC51:/mnt/f/Ctech IITD/ELP780 Software Lab/Assignment_8$ python3 ps1.py
010101110100101
0101011101001011
01001011101000100110101
nainam@LAPTOP-BNK1RC51:/mnt/f/Ctech IITD/ELP780 Software Lab/Assignment_8$
```

Figure 1.2: Output for PS1

2 Problem Statement-2

2.1 Problem Statement

3X3 Numeric Tic-Tac-Toe

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

2.2 Assumptions

- Positions and numbers lie in range 1 to 9.
- Player 1 starts first.
- Player 1 places odd numbers and Player 2 places even numbers.
- Once a position is occupied it can not be used again.
- All the numbers can only be used once.

2.3 Algorithm and Implementation

- Display welcome message.
- Take position and number from user.
- Check the validity of the input and raise appropriate flag.
- Place the number at corresponding position.
- Check if the sum in any row, column or diagonal has become 15.
- Declare the result.
- Ask for new game or to exit.

2.4 Program Structure

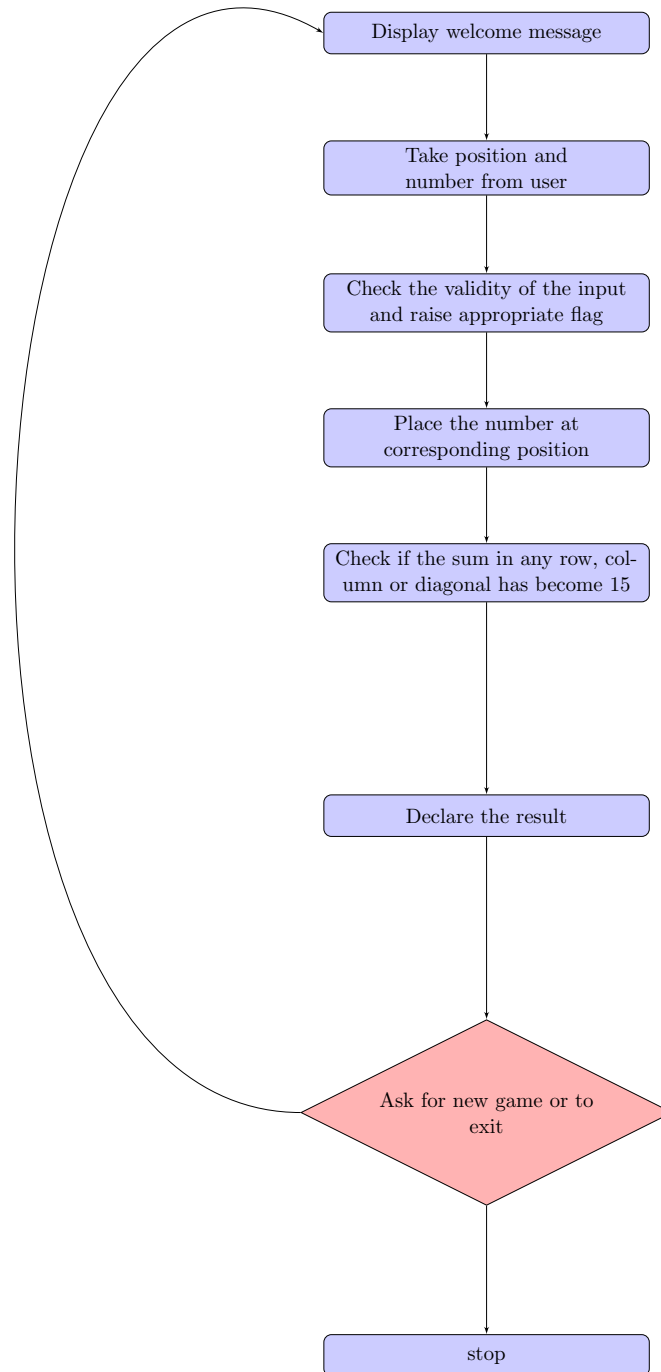


Figure 2.1: Flow Chart PS2

2.5 Input and Output format

- **Input Format** Take position and number as input (comma separated)
For example - Enter the position and number to be entered: 5,3
- **Output Format** Display the updated game data. Also, display the appropriate results such draw or player 1 wins.

2.6 Test cases

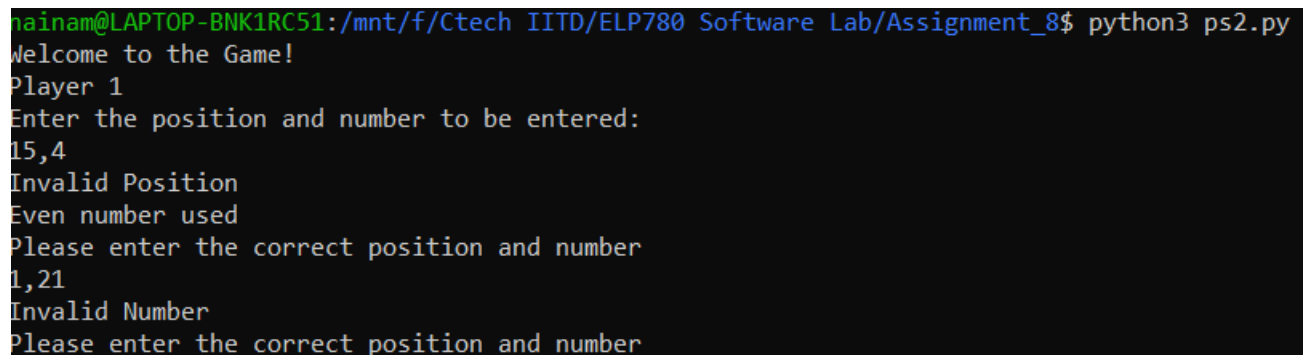
The code was tested for different test cases with player 2 as winner, draw condition. The code was also checked for boundary conditions. The results have been shown in screen-shot section.

2.7 Difficulties/Issues faced

- Proper variable passing should be done while calling various functions.
- Appropriate data-types should be used for corresponding problem.

2.8 Screenshots

Following are the screenshots



```
nainam@LAPTOP-BNK1RC51:/mnt/f/Ctech IITD/ELP780 Software Lab/Assignment_8$ python3 ps2.py
Welcome to the Game!
Player 1
Enter the position and number to be entered:
15,4
Invalid Position
Even number used
Please enter the correct position and number
1,21
Invalid Number
Please enter the correct position and number
```

Figure 2.2: PS2 Boundary Condition


```
nainam@LAPTOP-BNK1RC51:/mnt/f/Ctech IITD/ELP780 Software Lab/Assignment_8$ python3 ps2.py
Welcome to the Game!
Player 1
Enter the position and number to be entered:
1,5
5 0 0
0 0 0
0 0 0
Player 2
Enter the position and number to be entered:
2,2
5 2 0
0 0 0
0 0 0
Player 1
Enter the position and number to be entered:
5,1
5 2 0
0 1 0
0 0 0
Player 2
Enter the position and number to be entered:
2,8
Position already occupied
Please enter the correct position and number
3,8
5 2 8
0 1 0
0 0 0
Player 2 Wins!
End of the Game!
Enter 1 to play another game and 0 to exit
```

Figure 2.3: PS2 Output

```
Enter the position and number to be entered:
7,4
1 2 3
7 0 6
4 0 0
Player 1
Enter the position and number to be entered:
5,5
1 2 3
7 5 6
4 0 0
Player 2
Enter the position and number to be entered:
9,8
1 2 3
7 5 6
4 0 8
Player 1
Enter the position and number to be entered:
8,9
1 2 3
7 5 6
4 9 8
It's a Draw
End of the Game!
```

Figure 2.4: PS2 Output draw condition

A Code for Problem Statement 1

```
#!/usr/bin/python3.6

# Take bit string as input from user
s = input()

# variable to keep the count of '1' encountered
c = 0

# Loop to find c
for i in range(len(s)):
    if(s[i] == '1'):
        c += 1

# Appending appropriate parity bit
if(c%2 == 0):
    s_pc = s + '1'
else:
    s_pc = s + '0'

# List to store the index where '0' is to be stuffed
idx = []
i = 0
while(i < len(s)):
    if(s[i:i+3] == '010'): # Find '010'
        i += 3
        idx.append(i)
    else:
        i += 1

# List containing original string and parity bit
s_pc_s = list(s_pc)

for i in range(len(idx)):
    s_pc_s.insert(idx[i]+i,'0')

# Original string along with stuffed '0' bits and parity bits
s_pc_s = "".join(s_pc_s)
```

```
# Append '0101' to denote end of string
s_f = s_pc_s + '0101'

# Print original string with parity
print(s_pc)

# print final string
print(s_f)
```

B Code for Problem Statement 2

```
#!/usr/bin/python3.6

def main():
    # Dictionary to store the positions occupied along with the number
    print("Welcome to the Game!")
    play = 1
    while(play == 1):
        play_game()
        print("Enter 1 to play another game and 0 to exit");
        play = input()
        play = int(play)

# Function to check if sum of any complete row, column or diagonal has become 15
def check(game):
    finish = 0
    h1 = 0
    h2 = 0
    h3 = 0
    v1 = 0
    v2 = 0
    v3 = 0
    d1 = 0
    d2 = 0

    # Calculate sum in row 1 when filled completely
    if( (1 in game.keys()) and (2 in game.keys()) and (3 in game.keys())):
        h1 = game[1] + game[2] + game[3]

    # Calculate sum in row 2 when filled completely
    if( (4 in game.keys()) and (5 in game.keys()) and (6 in game.keys())):
        h2 = game[4] + game[5] + game[6]

    # Calculate sum in row 3 when filled completely
    if( (7 in game.keys()) and (8 in game.keys()) and (9 in game.keys())):
        h3 = game[7] + game[8] + game[9]

    # Calculate sum in col 1 when filled completely
```

```

    if( (1 in game.keys()) and (4 in game.keys()) and (7 in game.keys())):
        v1 = game[1] + game[4] + game[7]

    # Calculate sum in col 2 when filled completely
    if( (2 in game.keys()) and (5 in game.keys()) and (8 in game.keys())):
        v2 = game[2] + game[5] + game[8]

    # Calculate sum in col 3 when filled completely
    if( (3 in game.keys()) and (6 in game.keys()) and (9 in game.keys())):
        v3 = game[3] + game[6] + game[9]

    # Calculate sum in diagonal 1 when filled completely
    if( (1 in game.keys()) and (5 in game.keys()) and (9 in game.keys())):
        d1 = game[1] + game[5] + game[9]

    # Calculate sum in diagonal 2 when filled completely
    if( (3 in game.keys()) and (5 in game.keys()) and (7 in game.keys())):
        d2 = game[3] + game[5] + game[7]

    #Check if any of the sum is 15
    if( h1 == 15 or h2 == 15 or h3 == 15 or v1 == 15 or v2 == 15 or v3 == 15 or d1 == 15 or d2
        finish = 1

    # Return 1 if sum has reached 15 and 0 otherwise
    return finish

# Function to display the game status
def disp(game):
    c = {}
    for i in range(1,10):
        if(i in game.keys()):
            c[i] = game[i]
        else:
            c[i] = 0

    print(c[1],c[2],c[3])
    print(c[4],c[5],c[6])
    print(c[7],c[8],c[9])

# function to play the game
def play_game():
    chance = 1
    exit = 0
    player = 1
    flg = 0
    game = {}

    # Maximum 9 chances possible since there are 9 positions
    # Exit is made 1 when the game ends after wining or when all the positions are occupied

```

```

while(exit == 0 and chance < 10):
    if(chance%2 == 1):
        print("Player 1")
        player = 1
    if(chance%2 == 0):
        print("Player 2")
        player = 2
    print("Enter the position and number to be entered: ")
    p,n = input().split(',')
    p = int(p)
    n = int(n)

    # Boundary Condition Check
    if((p < 1) or ( p > 9)):
        print("Invalid Position")
        flg = 1

    if((n < 1) or ( n > 9)):
        print("Invalid Number")
        flg = 1

    if( n in game.values()):
        print("Number already used")
        flg = 1

    if( p in game.keys()):
        print("Position already occupied")
        flg = 1

    if((player == 1 and n%2 != 1) or (player == 2 and n%2 != 0)):
        if(player%2 == 1):
            print("Even number used")
        if(player%2 == 0):
            print("Odd number used")
        flg = 1

    # Loop till correct input is entered
    while(flg == 1):
        print("Please enter the correct position and number")
        p,n = input().split(',')
        p = int(p)
        n = int(n)
        if((p < 1) or ( p > 9)):
            print("Invalid Position")
            flg = 1
        elif((n < 1) or ( n > 9)):
            print("Invalid Number")

```

```
        flg = 1
    elif( n in game.values()):
        print("Number already used")
        flg = 1
    elif( p in game.keys()):
        print("Position already occupied")
        flg = 1
    elif((player == 1 and n%2 != 1) or (player == 0 and n%2 != 0)):
        if(player%2 == 1):
            print("Even number used")
        if(player%2 == 0):
            print("Odd number used")
        flg = 1
    else:
        flg = 0

# Add position and number in the dictionary
game[p] = n

disp(game)

# Call function to check if sum has reached 15 and declare the winner
if(check(game) == 1):
    print("Player",player,"Wins!")
    exit = 1
    print("End of the Game!")
elif (len(game) == 9):
    print("It's a Draw")
    print("End of the Game!")
    exit = 1
chance += 1

# Calling the main function to start execution of program
main()
```


C Git Screenshots

Following are the screenshot from git log.

```
nainam@LAPTOP-BNK1RC51:/mnt/f/Ctech IITD/ELP780 Software Lab/Assignment_8$ git log
commit 6aca2175b86b4532581f87a05e07da59c7f3517f (HEAD -> master)
Author: naina <nainam@LAPTOP-BNK1RC51.localdomain>
Date:   Wed Sep 18 12:36:35 2019 +0530

    PS2 final commit

commit daab1ff694c909b90cfdbdd87dd28d667cc512a5
Author: naina <nainam@LAPTOP-BNK1RC51.localdomain>
Date:   Wed Sep 18 11:29:27 2019 +0530

    PS2 without comments

commit c10004dbdb0e7cb46e60cc65ef29dfb146314898
Author: naina <nainam@LAPTOP-BNK1RC51.localdomain>
Date:   Wed Sep 18 10:29:45 2019 +0530

    PS1 final

commit f318d5157c9be633a943dad868c1b3cec2403ca6
Author: naina <nainam@LAPTOP-BNK1RC51.localdomain>
Date:   Wed Sep 18 10:14:02 2019 +0530

    PS1 without comments

commit 0e04d4d1e810f1541fc559219d7e501b8b79f02e (origin/master, origin/HEAD)
Author: 2018JTM2250 <43585054+2018JTM2250@users.noreply.github.com>
Date:   Tue Sep 17 17:04:57 2019 +0530

    Create ps2.py

commit 5b9eeddda3aff8dabeaa1238bbe70a68d32faa19
Author: 2018JTM2250 <43585054+2018JTM2250@users.noreply.github.com>
Date:   Tue Sep 17 17:04:45 2019 +0530

    Create ps1.py

commit c653c1e981f7f86739ecedc05d3fd5fcec7b4dd2
```

Figure C.1: Git Log

Bibliography

- [1] Python programming Tutorial
<https://www.tutorialspoint.com/python/index.htm>
- [2] GitHub Tutorial
<https://guides.github.com/activities/hello-world/>