

Assignment 8

ELP-780 Software Lab

Asmita Patil

2018EET2560

2019-20

A report presented for the assignment

Python and Github [GitHub Link](#)



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	Program Structure	2
1.4	Algorithm and Implementation	2
1.5	Input and Output format	3
1.6	Test cases	3
1.7	Difficulties/Issues faced	4
1.8	Screenshots	4
2	Problem Statement-2	7
2.1	Problem Statement	7
2.2	Assumptions	7
2.3	Program Structure	8
2.4	Algorithm and Implementation	9
2.5	Input and Output format	9
2.6	Test cases	10
2.7	Difficulties/Issues faced	10
2.8	Screenshots	10
	Appendices	13
A	Code for Problem Statement 1	14
B	Code for Problem Statement 2	16
C	Code for Makefile	20
D	Output of history command from terminal	21

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.

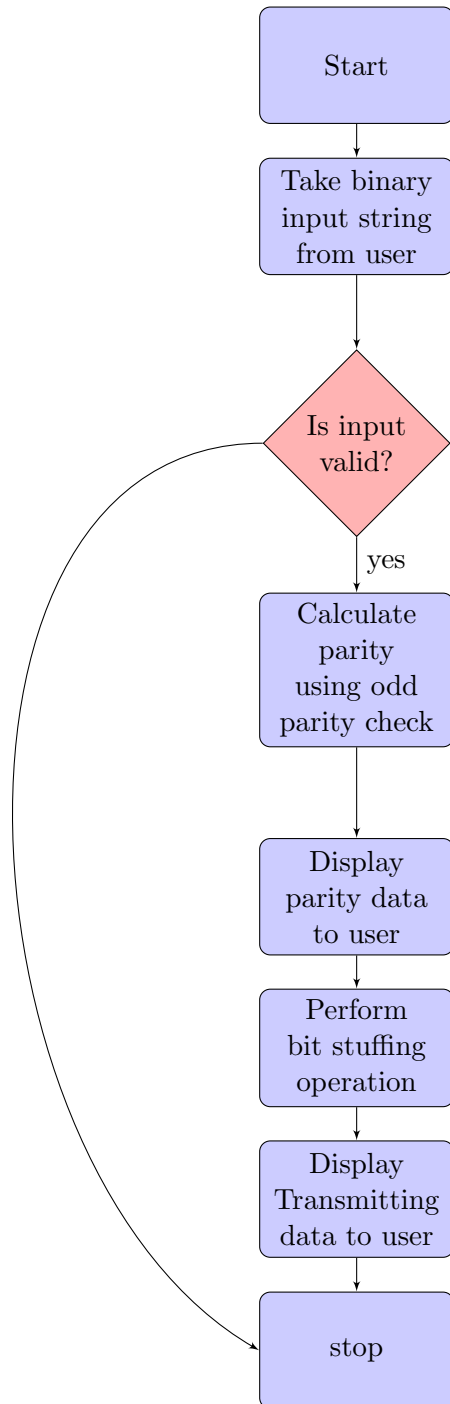
- **Bit-Oriented Framing**

- 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

- Compiler for Python and all required libraries are already installed.
- User is familiar with all concepts of Python required for assignment.[1]
- User have experience with Github and Makefile.[2]

1.3 Program Structure



1.4 Algorithm and Implementation

Problem can be solved when execution is done in following steps:

- *Take binary input string from user.*
- *Check the string is valid or not. If not valid exit with error.*
- *Calculate the parity for given string and append it at the end of string.*
- *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.*
- *Perform bit stuffing to handle flag properly.*
- *Initialize all variables required for execution and calculation.*
- *Print parity data and transmitting data to the user.*

Implementation is done in python and screen-shots and code snippet is also provided. During the implementation, I have fork and cloned GIT project and performed following git commands.

- git add
- git commit
- git status
- git log

1.5 Input and Output format

- **Input :**

- Enter binary bit data that has to be transmitted.
- **Sample Input**
010101110100101

- **Output :**

- Print binary bit data with parity bit.
- Print the modified string that is to be transmitted
- **Sample Output**
Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101

1.6 Test cases

Following test cases are executed :

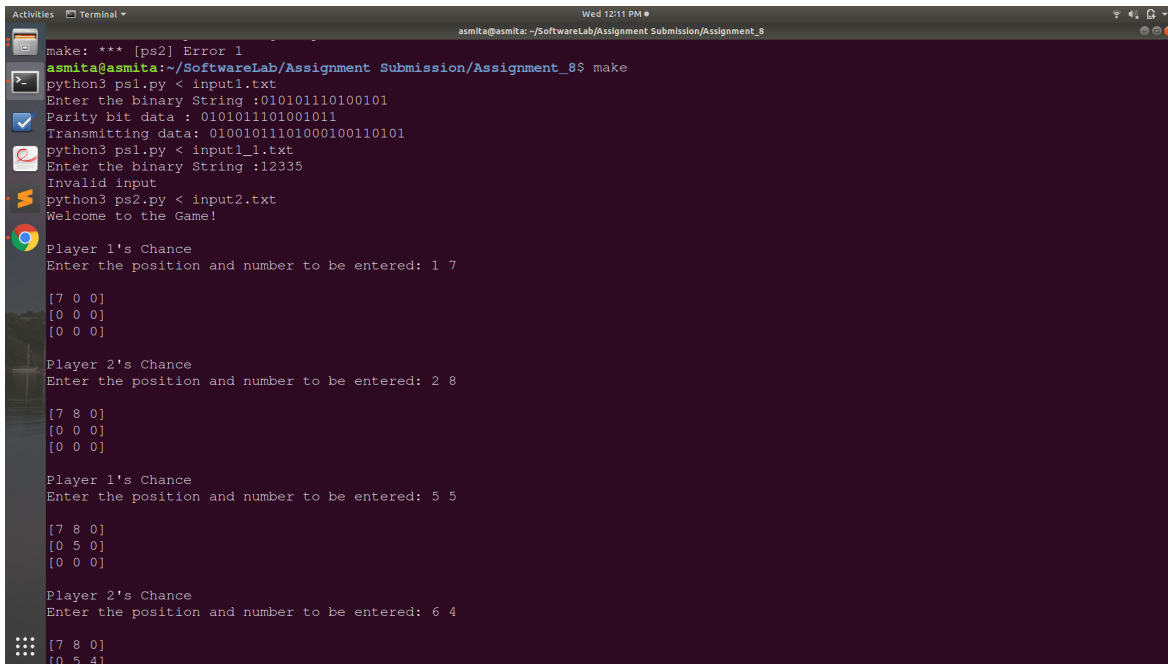
- Input which gives parity 0 is executed
- Input which gives parity 1 is executed
Screenshots are attached for the same

1.7 Difficulties/Issues faced

- Inserting 0 for bit stuffing. Created new list of character to solve this.

1.8 Screenshots

Following are the screenshots



```
make: *** [ps2] Error 1
asmitha@asmitha:~/SoftwareLab/Assignment Submission/Assignment_8$ make
python3 ps1.py < input1.txt
Enter the binary String :010101110100101
Parity bit data : 0101011101001011
Transmitting data: 01001011101000100110101
python3 ps1.py < input1_1.txt
Enter the binary String :12335
Invalid input
python3 ps2.py < input2.txt
Welcome to the Game!

Player 1's Chance
Enter the position and number to be entered: 1 7

[7 0 0]
[0 0 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 2 8

[7 8 0]
[0 0 0]
[0 0 0]

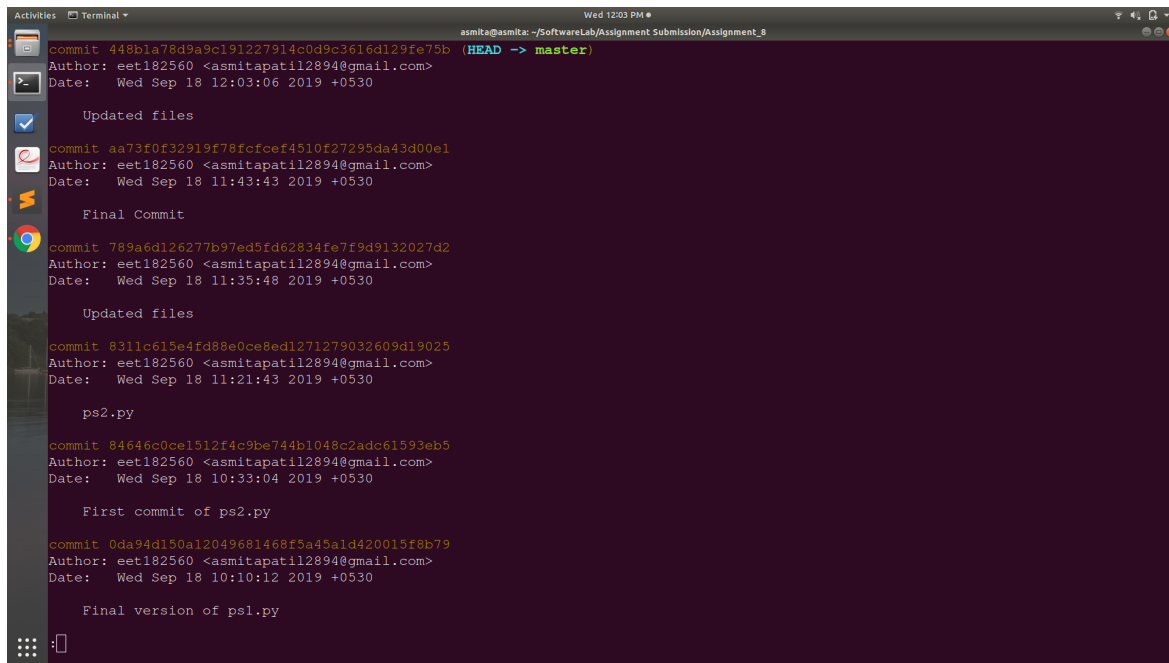
Player 1's Chance
Enter the position and number to be entered: 5 5

[7 8 0]
[0 5 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 6 4

[7 8 0]
[0 5 4]
```

Figure 1.1: Execution of problem statement 1 and Running makefile



```

commit 448b1a78d9a9c191227914c0d9c3616d129fe75b (HEAD -> master)
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 12:03:06 2019 +0530

    Updated files

commit aa73f0f32919f78fcfeef4510f27295da43d00e1
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 11:43:43 2019 +0530

    Final Commit

commit 789a6d126277b97ed5fd62834fe7f9d9132027d2
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 11:35:48 2019 +0530

    Updated files

commit 8311c615e4fd88e0ce8ed1271279032609d19025
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 11:21:43 2019 +0530

    ps2.py

commit 84646c0ce1512f4c9be744b1048c2adc61593eb5
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 10:33:04 2019 +0530

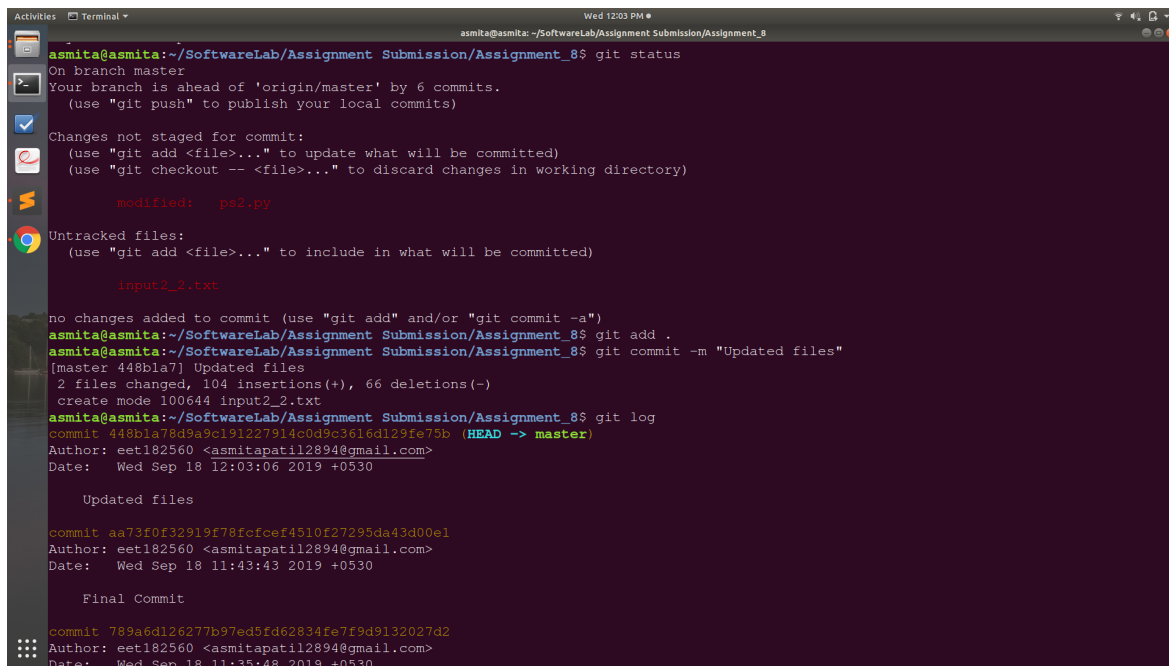
    First commit of ps2.py

commit 0da94d150a12049681468f5a45a1d420015f8b79
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 10:10:12 2019 +0530

    Final version of ps1.py

```

Figure 1.2: Execution of Git Command



```

asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git status
On branch master
Your branch is ahead of 'origin/master' by 6 commits.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ps2.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        input2_2.txt

no changes added to commit (use "git add" and/or "git commit -a")
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git add .
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git commit -m "Updated files"
[master 448b1a7] Updated files
 2 files changed, 104 insertions(+), 66 deletions(-)
 create mode 100644 input2_2.txt
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git log
commit 448b1a78d9a9c191227914c0d9c3616d129fe75b (HEAD -> master)
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 12:03:06 2019 +0530

    Updated files

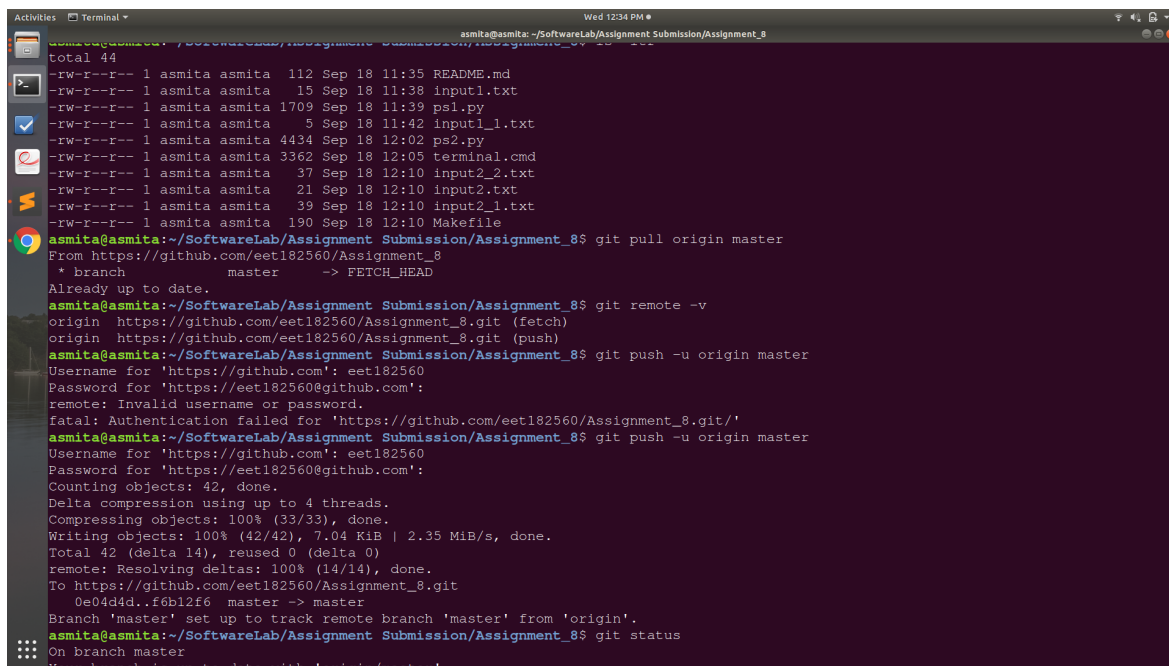
commit aa73f0f32919f78fcfeef4510f27295da43d00e1
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 11:43:43 2019 +0530

    Final Commit

commit 789a6d126277b97ed5fd62834fe7f9d9132027d2
Author: eet182560 <asmitapatil2894@gmail.com>
Date:   Wed Sep 18 11:35:48 2019 +0530

```

Figure 1.3: Execution of Git Command



```
asmita@asmita: ~/SoftwareLab/Assignment Submission/Assignment_8
total 44
-rw-r--r-- 1 asmita asmita 112 Sep 18 11:35 README.md
-rw-r--r-- 1 asmita asmita 15 Sep 18 11:38 input1.txt
-rw-r--r-- 1 asmita asmita 1709 Sep 18 11:39 ps1.py
-rw-r--r-- 1 asmita asmita 5 Sep 18 11:42 input1_1.txt
-rw-r--r-- 1 asmita asmita 4434 Sep 18 12:02 ps2.py
-rw-r--r-- 1 asmita asmita 3362 Sep 18 12:05 terminal.cmd
-rw-r--r-- 1 asmita asmita 37 Sep 18 12:10 input2_2.txt
-rw-r--r-- 1 asmita asmita 21 Sep 18 12:10 input2.txt
-rw-r--r-- 1 asmita asmita 39 Sep 18 12:10 input2_1.txt
-rw-r--r-- 1 asmita asmita 190 Sep 18 12:10 Makefile
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git pull origin master
From https://github.com/eet182560/Assignment_8
* branch      master      -> FETCH_HEAD
Already up to date.
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git remote -v
origin https://github.com/eet182560/Assignment_8.git (fetch)
origin https://github.com/eet182560/Assignment_8.git (push)
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git push -u origin master
Username for 'https://github.com': eet182560
Password for 'https://eet182560@github.com':
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/eet182560/Assignment_8.git/'
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git push -u origin master
Username for 'https://github.com': eet182560
Password for 'https://eet182560@github.com':
Counting objects: 42, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (33/33), done.
Writing objects: 100% (42/42), 7.04 KiB | 2.35 MiB/s, done.
Total 42 (delta 14), reused 0 (delta 0)
remote: Resolving deltas: 100% (14/14), done.
To https://github.com/eet182560/Assignment_8.git
0e04d4d..f6b12f6 master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$ git status
On branch master
Your branch is up to date with 'origin/master'
```

Figure 1.4: Execution of Git Command

2 Problem Statement-2

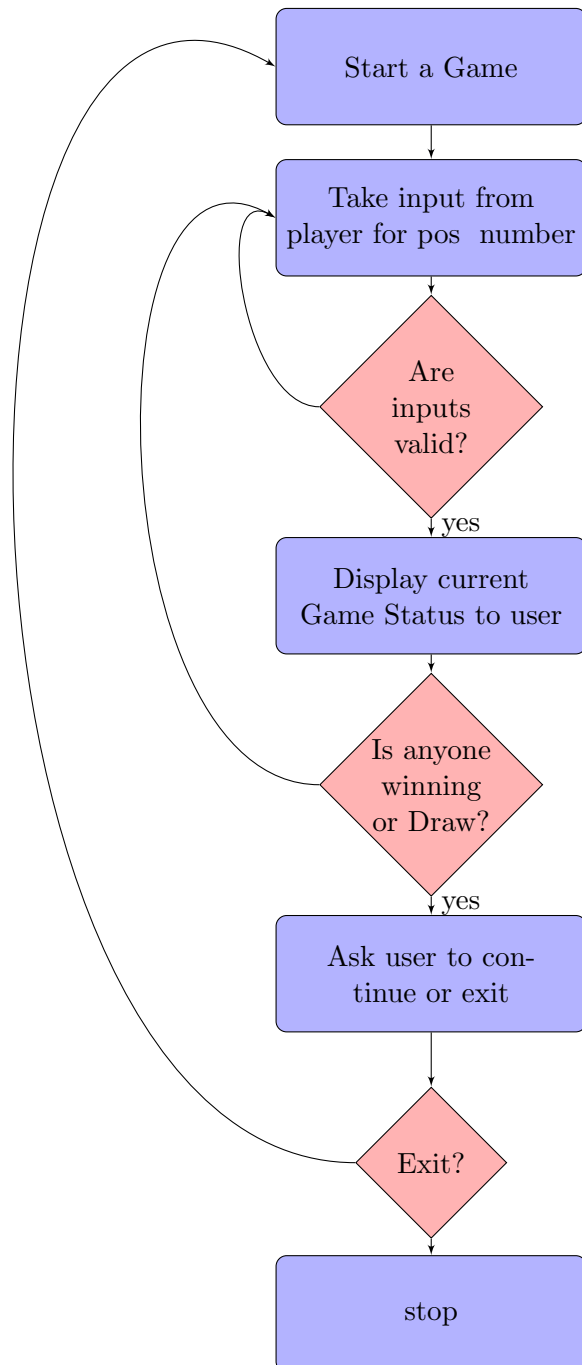
2.1 Problem Statement

- **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 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

2.2 Assumptions

- All basics commands of GIT are known to user and user have some hands on experience with GIT
- Basics concepts of String and list in python are known to user.

2.3 Program Structure



2.4 Algorithm and Implementation

Problem can be solved when execution is done in following steps:

- *Take input from Player for position and number.*
- *Check validity of position and number for given constraints and current player.*
- *Display current board status/ Game Status to user.*
- *Check for winner if exists then ask user to exit the game or continue with new game.*
- *Winning condition will be checked across rows, columns and diagonally.*
- *Switch the player and go to step 1.*
- *Handle boundary conditions and flag error appropriately.*
- *Display required output when required.*

Implementation is done in Python and screenshot, code snippet is attached for the same.

2.5 Input and Output format

- **Input :**
 - Get the position and number to be entered from the user.
 - Get input from user in case of draw or any of the two players is winner to continue with new game or to exit
- **Output :**
 - Print Welcome to the Game!.
 - Print whether it is Player 1s or Player 2s chance.
 - Show tic tac toe with data.
 - **Sample output:**
Welcome to the Game!
Player 1s chance
Enter the position and number to be entered: 5,3
- **Constraint :**
 - $1 \leq \text{Position} \leq 9$
 - $1 \leq \text{Number} \leq 9$

2.6 Test cases

Following test cases are executed :

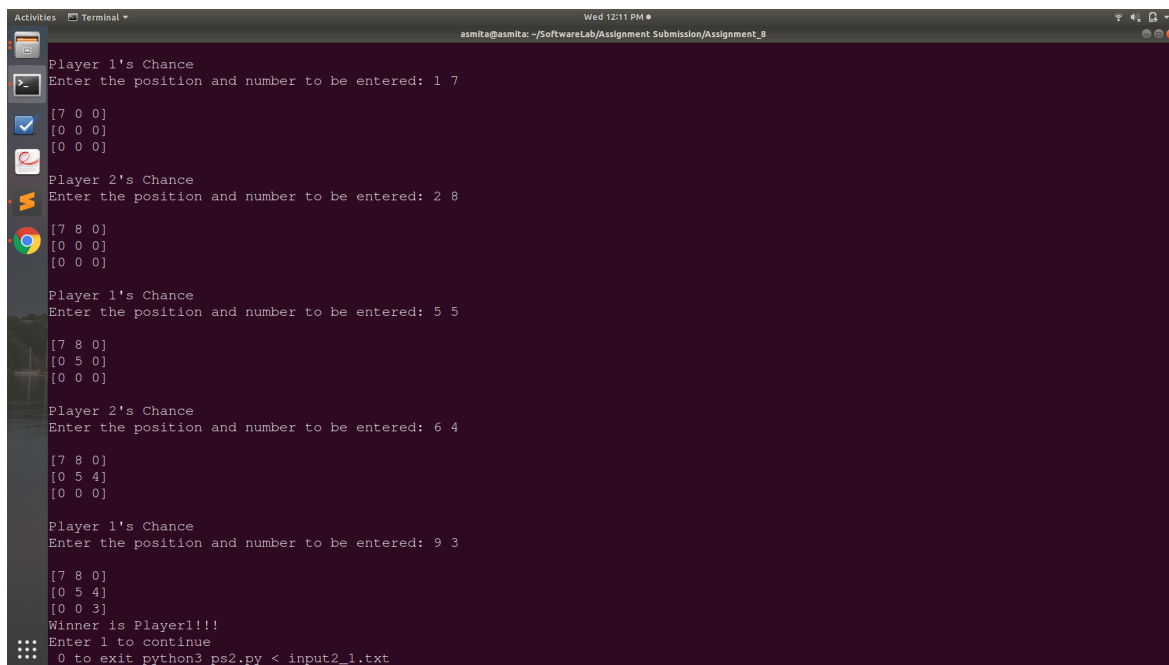
- Input which gives normal output is executed
- Input which results into error is executed
Screenshots of output are shown below.

2.7 Difficulties/Issues faced

- Difficult to find position to calculate sum row wise, column wise and diagonally. Solved using pre-defined arrays.

2.8 Screenshots

Following are the screenshots



```
Activities Terminal
asmkta@asmkta: ~/SoftwareLab/Assignment Submission/Assignment_8
Wed 12:11 PM

Player 1's Chance
Enter the position and number to be entered: 1 7

[7 0 0]
[0 0 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 2 8

[7 8 0]
[0 0 0]
[0 0 0]

Player 1's Chance
Enter the position and number to be entered: 5 5

[7 8 0]
[0 5 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 6 4

[7 8 0]
[0 5 4]
[0 0 0]

Player 1's Chance
Enter the position and number to be entered: 9 3

[7 8 0]
[0 5 4]
[0 0 3]

Winner is Player1!!!
Enter 1 to continue
0 to exit python3 ps2.py < input2_1.txt
```

Figure 2.1: Execution of problem statement 2: Normal Exit

```

0 to exit python3 ps2.py < input2_1.txt
Welcome to the Game!

Player 1's Chance
Enter the position and number to be entered: 2 5

[0 5 0]
[0 0 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 5 8

[0 5 0]
[0 8 0]
[0 0 0]

Player 1's Chance
Enter the position and number to be entered: 7 1

[0 5 0]
[0 8 0]
[1 0 0]

Player 2's Chance
Enter the position and number to be entered: 8 2

[0 5 0]
[0 8 0]
[1 2 0]

Winner is Player2!!!
Enter 1 to continue
0 to exit Welcome to the Game!

Player 1's Chance
Enter the position and number to be entered: 1 7

[7 0 0]

```

Figure 2.2: Execution of problem statement 2: Continue with new Game

```

Enter 1 to continue
0 to exit python3 ps2.py < input2_2.txt
Welcome to the Game!

Player 1's Chance
Enter the position and number to be entered: 1 1

[1 0 0]
[0 0 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 2 2

[1 2 0]
[0 0 0]
[0 0 0]

Player 1's Chance
Enter the position and number to be entered: 3 3

[1 2 3]
[0 0 0]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 6 6

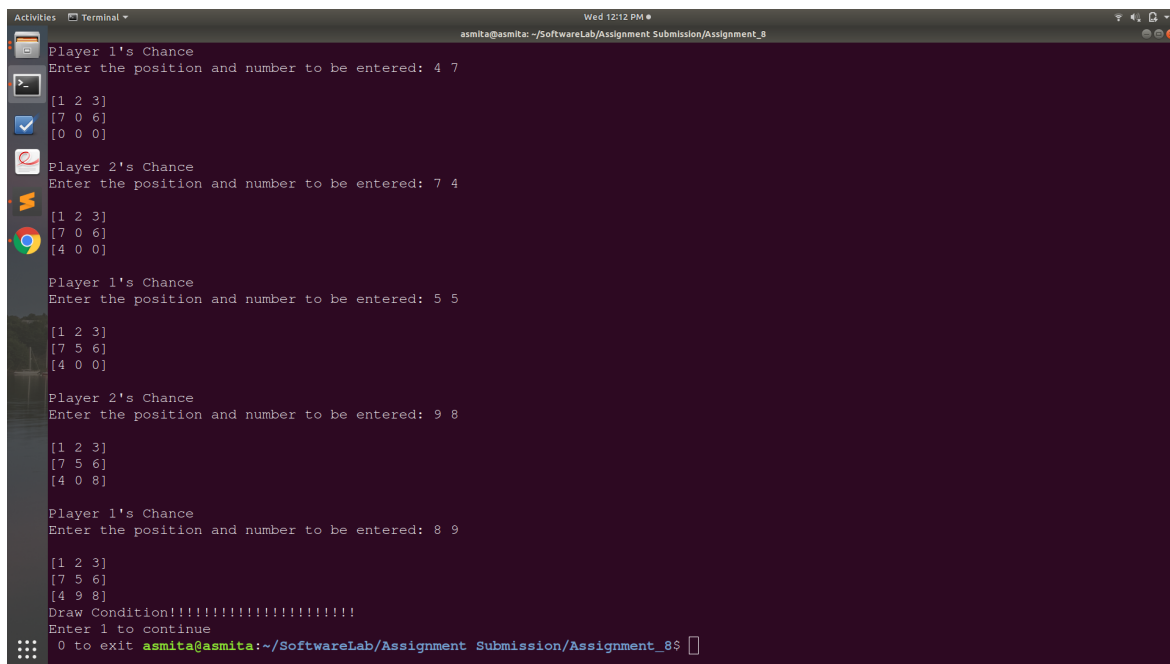
[1 2 3]
[0 0 6]
[0 0 0]

Player 1's Chance
Enter the position and number to be entered: 4 7

[1 2 3]
[7 0 6]
[0 0 0]

```

Figure 2.3: Execution of problem statement 2: Draw Case1



```
Player 1's Chance
Enter the position and number to be entered: 4 7

[1 2 3]
[7 0 6]
[0 0 0]

Player 2's Chance
Enter the position and number to be entered: 7 4

[1 2 3]
[7 0 6]
[4 0 0]

Player 1's Chance
Enter the position and number to be entered: 5 5

[1 2 3]
[7 5 6]
[4 0 0]

Player 2's Chance
Enter the position and number to be entered: 9 8

[1 2 3]
[7 5 6]
[4 0 8]

Player 1's Chance
Enter the position and number to be entered: 8 9

[1 2 3]
[7 5 6]
[4 9 8]

Draw Condition!!!!!!!!!!!!!!!!!!!!!!
Enter 1 to continue
0 to exit asmita@asmita:~/SoftwareLab/Assignment Submission/Assignment_8$
```

Figure 2.4: Execution of problem statement 2: Draw case2

Appendices

A Code for Problem Statement 1

```
1 # Function Defintions
2 # Function which check validity of string if it is binary or not
3 def checkVal(String):
4     # Convert string into list
5     newList = list(String)
6     # Check string character by character
7     for char in newList:
8         if char == '0' or char == '1':
9             continue
10        else:
11            return -1;
12
13    # Return 0 if in case of valid string else -1
14    return 0;
15
16
17 # Function to calculate parity of string
18 def calcParity(String):
19
20     # Initialization of variables
21     i = 0
22     count = 0
23
24     # Count numbers of 1's
25     while i < len(String):
26         if String[i] == '1':
27             count += 1
28             i += 1
29
30     # Check number of 1's is odd or even
31     if count % 2 == 0:
32         count = 1
33     else:
34         count = 0
35
36     # Return parity
37     return count
38
39
40 # Function which generate transmission data after bit stuffing if required
41 def generateTransmissiondata(strParity):
42
43     # Initialization of variables
44     i = 0
```



```
45 j = 0
46
47 # Copying string into list to append
48 newStr = list(strParity)
49
50 # Check for pattern '010' in input data
51 while i < len(strParity) - 3:
52
53     # If pattern is found append it with 0
54     if strParity[i : i + 3] == '010':
55         newStr.insert(j + 3, '0')
56         i += 2
57         j += 3
58     i += 1
59     j += 1
60
61 # Return string with flag appended '0101'
62 return ''.join(newStr) + '0101'
63
64 ## Execution start from here
65 # Take input from user
66 String = input("Enter the binary String :")
67 print(String)
68 #Check validity of String
69 if checkVal(String) == -1:
70     print("Invalid input")
71     exit(0)
72
73 # Calculate parity of string and print the string with parity
74 count = calcParity(String)
75 strParity = String + str(count)
76 print("Parity bit data : " + strParity)
77
78 # Calculate data to be transmitted and print the same
79 outPut = generateTransmissiondata(strParity)
80 print("Transmitting data: " + outPut)
```

B Code for Problem Statement 2

```
1 import numpy as np
2
3 # Function Declarations and Definitions
4 # Display the matrix / board
5 def display(mat):
6     print()
7     print(mat[0: 3])
8     print(mat[3 : 6])
9     print(mat[6 : 9])
10
11
12 # Function to check sum is 15 or not and exit based on it
13 def sumCheck(sum, player):
14     if sum == 15:
15         print("Winner is Player" + str(player) + "!!!")
16         flag = input("Enter 1 to continue\n 0 to exit ")
17         if flag == "1":
18             Game()
19         else:
20             exit(0)
21
22
23 # Function to calculate sum rowwise based on pos
24 def rowSum(mat, pos, player, vis):
25
26     #Initialization of variables
27     sum = 0
28     valid = False
29
30     # Calculate sum for first row pos
31     if pos in row1:
32         sum = mat[0] + mat[1] + mat[2]
33         if vis[0] and vis[1] and vis[2]:
34             valid = True
35
36     # Calculate sum for second row pos
37     elif pos in row2:
38         sum = mat[3] + mat[4] + mat[5]
39         if vis[3] and vis[4] and vis[5]:
40             valid = True
41
42     # Calculate sum for third row pos
43     else:
44         sum = mat[6] + mat[7] + mat[8]
```

```

45     if vis[6] and vis[7] and vis[8]:
46         valid = True
47
48     # Check validity of the sum and call sumCheck
49     if valid:
50         sumCheck(sum, player)
51
52
53 # Function to calculate sum column wise based on pos
54 def colSum(mat, pos, player, vis):
55
56     #Initialization of variables
57     sum = 0
58     valid = False
59
60     # Calculate sum for first column position
61     if pos in col1:
62         sum = mat[0] + mat[3] + mat[6]
63         if vis[0] and vis[3] and vis[6]:
64             valid = True
65
66     # Calculate sum for second column position
67     elif pos in col2:
68         sum = mat[1] + mat[4] + mat[7]
69         if vis[1] and vis[4] and vis[7]:
70             valid = True
71
72     # Calculate sum for third column position
73     else:
74         sum = mat[2] + mat[5] + mat[8]
75         if vis[2] and vis[5] and vis[8]:
76             valid = True
77
78     # Check validity of the sum and call sumCheck
79     if valid:
80         sumCheck(sum, player)
81
82 # Function to calculate sum diagonally based on pos
83 def diagonal(mat, pos, player, vis):
84
85     #Initialization of variables
86     sum = 0
87     valid = False
88
89     # Calculate sum for second diagonal position
90     if pos in diagonal1:
91         sum = mat[0] + mat[4] + mat[8]
92         if vis[0] and vis[4] and vis[8]:
93             valid = True
94
95     # Check validity of the sum and call sumCheck
96     if valid:
97         sumCheck(sum, player)
98
99     # Calculate sum for second diagonal position
100    if pos in diagonal2:

```

```

101     sum = mat[2] + mat[4] + mat[6]
102     if vis[2] and vis[4] and vis[6]:
103         valid = True
104
105     # Check validity of the sum and call sumCheck
106     if valid:
107         sumCheck(sum, player)
108
109
110 #Check for draw condition
111 def checkDraw(vis):
112     flag = True
113     for i in vis:
114         if i == False:
115             flag = False
116             break
117
118 # if draw then give user option to continue or exit
119 if flag:
120     print("Draw Condition!!!!!!!!!!!!!!!!!!!!!!")
121     flag = input("Enter 1 to continue\n 0 to exit ")
122     if flag == "1":
123         Game()
124     else:
125         exit(0)
126
127
128 # Function to start a game
129 def Game():
130
131     # Initialization of all variables
132     mat = np.zeros(9, dtype=np.int32)
133     vis = [False] * 9
134     numVis = [False] * 9
135
136     # Start of game with player1
137     print("Welcome to the Game!")
138     player = 1
139
140     # Infinite to execute game
141     while True:
142
143         # take user input for position and number to be used
144         print()
145         print("Player " + str(player) + "'s Chance")
146         pos, num = input("Enter the position and number to be entered: ").split(
147             ",")
148         print(pos, num)
149
150         # Check validity of position
151         pos = int(pos)
152         if pos < 1 or pos > 9:
153             print("Invalid position")
154             continue
155
156         # Check validity of number

```

```

156     num = int(num)
157     if num < 1 or num > 9:
158         print("Invalid number")
159         continue
160
161     # Check validity of number based on player's chance
162     if player == 1 and num not in player1:
163         print("Please enter odd number")
164         continue
165     elif player == 2 and num not in player2:
166         print("Please enter even number")
167         continue
168
169     # Check validity of number and position
170     if vis[pos - 1] == True:
171         print("Enter the position which is not used")
172         continue
173     if numVis[num - 1] == True:
174         print("Enter the number which is not used yet")
175         continue
176
177     # update the variables for given input
178     vis[pos - 1] = True
179     numVis[num - 1] = True
180     mat[pos - 1] = num
181
182     # Display current board status
183     display(mat)
184
185     # Check for winner if exists end the game else continue
186     rowSum(mat, pos, player, vis)
187     colSum(mat, pos, player, vis)
188     diagonal(mat, pos, player, vis)
189
190     checkDraw(vis)
191
192     # Switch the player
193     if player == 1:
194         player = 2
195     else:
196         player = 1
197
198
199     ### Execution starts from here
200
201     player1 = {1, 3, 5, 7, 9}
202     player2 = {2, 4, 6, 8}
203     row1 = {1, 2, 3}
204     row2 = {4, 5, 6}
205     col1 = {1, 4, 7}
206     col2 = {2, 5, 8}
207     diagonal1 = {1, 5, 9}
208     diagonal2 = {3, 5, 7}
209     Game()

```

C Code for Makefile

```
1 all: ps1 ps2
2
3 ps1: ps1.py
4     python3 ps1.py < input1.txt
5     python3 ps1.py < input1_1.txt
6
7 ps2: ps2.py
8     python3 ps2.py < input2.txt
9     python3 ps2.py < input2_1.txt
10    python3 ps2.py < input2_2.txt
```

D Output of history command from terminal

```
1881 git -version
1882 git --version
1883 sudo apt-get install git
1884 gedit .bashrc
1885 git config --global http.proxy 10.10.78.62:3128
1886 git config --global https.proxy 10.10.78.62:3128
1887 git config --global
1888 git config --list
1889 mkdir eet182560_8
1890 touch ps1.py
1891 touch ps2.py
1892 touch Makefile
1893 ls -lrt
1894 mv *.py Makefile ./eet182560_8
1895 ls -lrt
1896 cd eet182560_8
1897 ls -lrt
1898 vim Makefile
1899 make
1900 touch .gitignore
1901 ls -lrt
1902 rm Makefile~
1903 ls -lrta
1904 rm .Makefile.un~
1905 ls -lrta
1906 make
1907 clear
1908 exit
1909 python ps1.py
1910 python3 ps1.py
1911 S
1912 D
1913 s
```

```
1914 S
1915 D
1916 S
1917 D
1918 S
1919 python3 ps1.py
1920 clear
1921 python3 ps2.py
1922 exit
1923 git clone https://github.com/eet182560/hello-world.git
1924 ls -lrt
1925 cd hello-world/
1926 ls -lrt
1927 ls -la
1928 git config --list
1929 clear
1930 git config --global user.name "eet182560"
1931 git config --global user.email "asmitapatil2894@gmail.com"
1932 git config --list
1933 git branch
1934 git branch newBr
1935 git branch
1936 git checkout newBr
1937 git branch
1938 git checkout master
1939 ls -lrt
1940 cat README.md
1941 clear
1942 vim README.md
1943 git status
1944 ls -ltr
1945 git branch
1946 git add README.md
1947 git commit -m "Commit 1"
1948 git log
1949 git status
1950 touch .gitignore
1951 vim .gitignore
1952 cat .gitignore
1953 git status
1954 git add .gitignore
1955 git status
1956 rm ..gitignore.un~ .gitignore~
1957 git status
1958 git commit -m "Commit 2"
```



```
1959 git status
1960 git remote add origin https://github.com/eet182560/hello-world.git
1961 git push -u origin master
1962 touch test.txt
1963 git status
1964 git add test.txt
1965 git status
1966 git commit -m "Commit 3"
1967 git log
1968 git push -u origin master
1969 git pull origin master
1970 git status
1971 cd ..
1972 git clone https://github.com/eet182560/Spoon-Knife.git
1973 clear
1974 git remote -v
1975 ls -lrt
1976 cd Spoon-Knife/
1977 ls -lrt
1978 git remote -v
1979 git remote add upstream https://github.com/octocat/Spoon-Knife.git
1980 git remote -v
1981 git fetch upstream
1982 git branch
1983 git merge upstream/master
1984 exit
1985 make
1986 make clean
1987 make ps1
1988 make ps2
1989 cat ouput
1990 cat ouput.txt
1991 make clean
1992 cat Makefile
1993 make clean
1994 make ps2
1995 cat output.txt
1996 make ps1
1997 make clean
1998 git clone https://github.com/eet182560/Assignment_8.git
1999 ls -lrt
2000 cd Assignment_8/
2001 ls -lrt
2002 cat README.md
2003 cat ps1.py
```

```
2004 cat ps2.py
2005 clear
2006 ls -lrt
2007 python3 ps1.py
2008 clear
2009 git status
2010 git add .
2011 git commit -m "Initial Commit"
2012 git log
2013 python3 ps1.py
2014 clear
2015 python3 ps1.py
2016 clear
2017 python3 ps1.py
2018 clear
2019 python3 ps1.py
2020 git status
2021 git add .
2022 git commit -m "Final version of ps1.py"
2023 git log
2024 git status
2025 clear
2026 python3 ps2.py
2027 git status
2028 git add .
2029 git commit -m "First commit of ps2.py"
2030 git status
2031 python3 ps2.py
2032 git commit -m "First commit of ps2.py"
2033 python3 ps2.py
2034 git status
2035 git commit -a "Commit"
2036 git commit -a
2037 git log
2038 git add .
2039 git commit -m "ps2.py"
2040 ls -lrt
2041 git log
2042 clear
2043 python3 ps2.py
2044 git status
2045 git add .
2046 git commit -m "Updated files"
2047 git log
2048 python3 ps1.py
```

```
2049 ls -lrt
2050 make
2051 mv input1_1.txt input2_1.txt
2052 make
2053 git status
2054 git add .
2055 git commit -m "Final Commit"
2056 git status
2057 git log
2058 ls -lrt
2059 python3 ps2.py
2060 2,8
2061 5,5
2062 6,4
2063 python3 ps2.py
2064 python3 ps2.py < input2_2.txt
2065 python3 ps2.py
2066 git status
2067 git add .
2068 git commit -m "Updated files"
2069 git log
2070 history > termina.cmd
2071 ls -lrt
2072 mv termina.cmd terminal.cmd
2073 ls -lrt
2074 git log
2075 git status
2076 git add .
2077 git commit -m "Command file"
2078 git status
2079 git add .
2080 git commit -m "Command file"
2081 git log
2082 clear
2083 make
2084 git status
2085 git add .
2086 git commit -m "Final Commit"
2087 ls -ltr
2088 git pull origin master
2089 git remote -v
2090 git push -u origin master
2091 git status
2092 history > terminal.cmd
```

Bibliography

- [1] “Python strings,” [Online]. Available: https://www.tutorialspoint.com/python/python_strings.htm.
- [2] “Forking a repository,” [Online]. Available: <https://help.github.com/en/articles/fork-a-repo>.