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

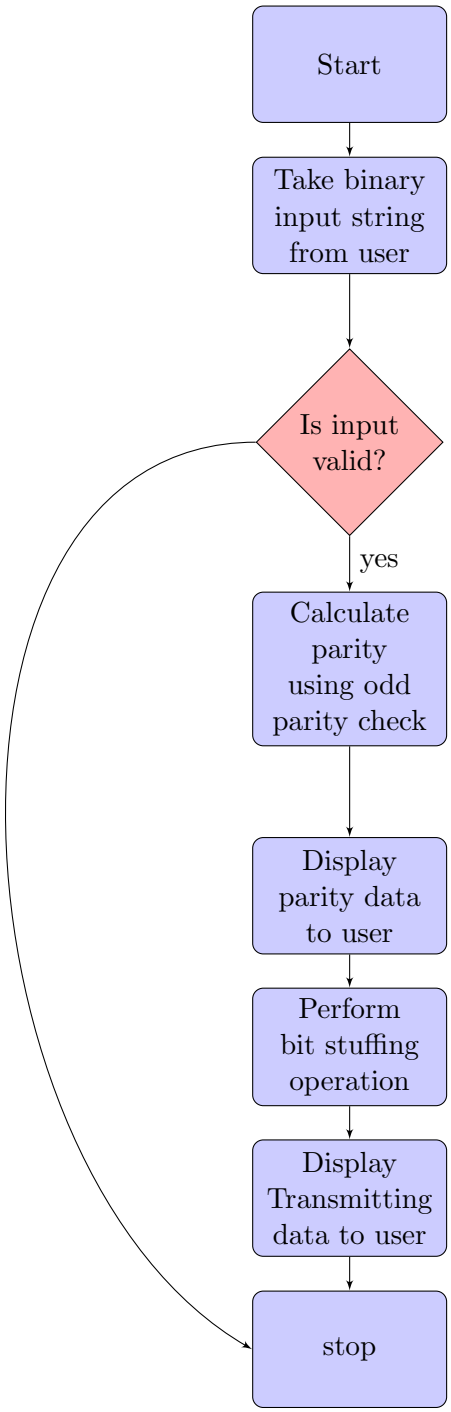
1.1 Problem Statement

- **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: 01001011110100010011010 1

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

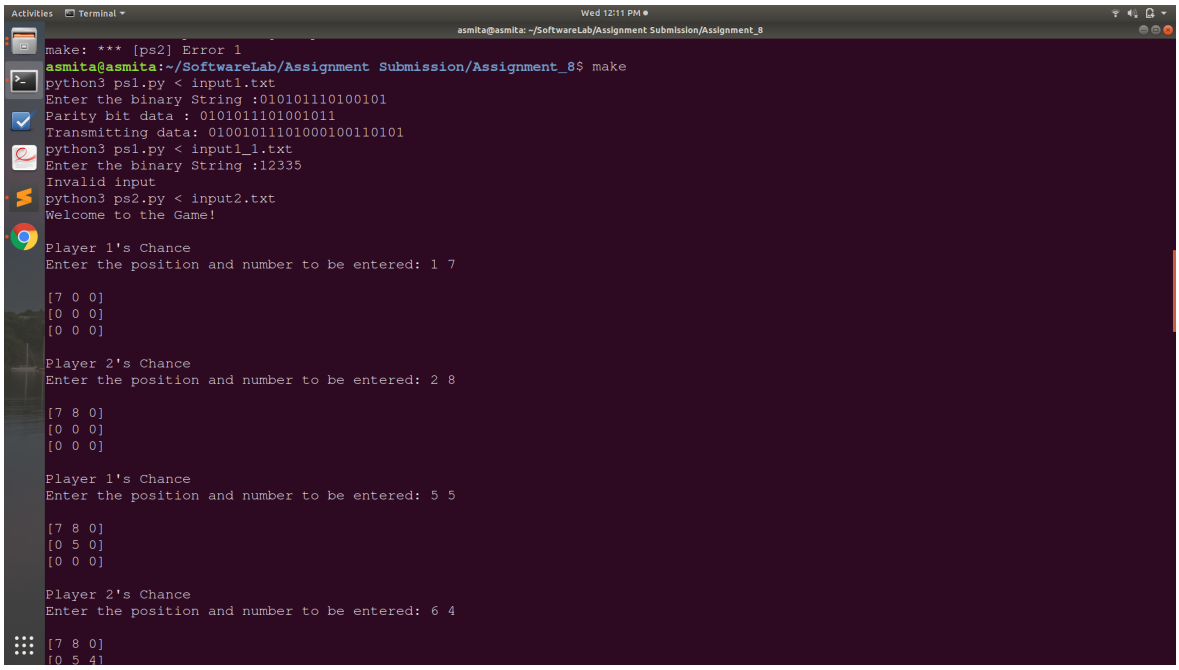


Figure 1.1: Execution of problem statement 1 and Running makefile

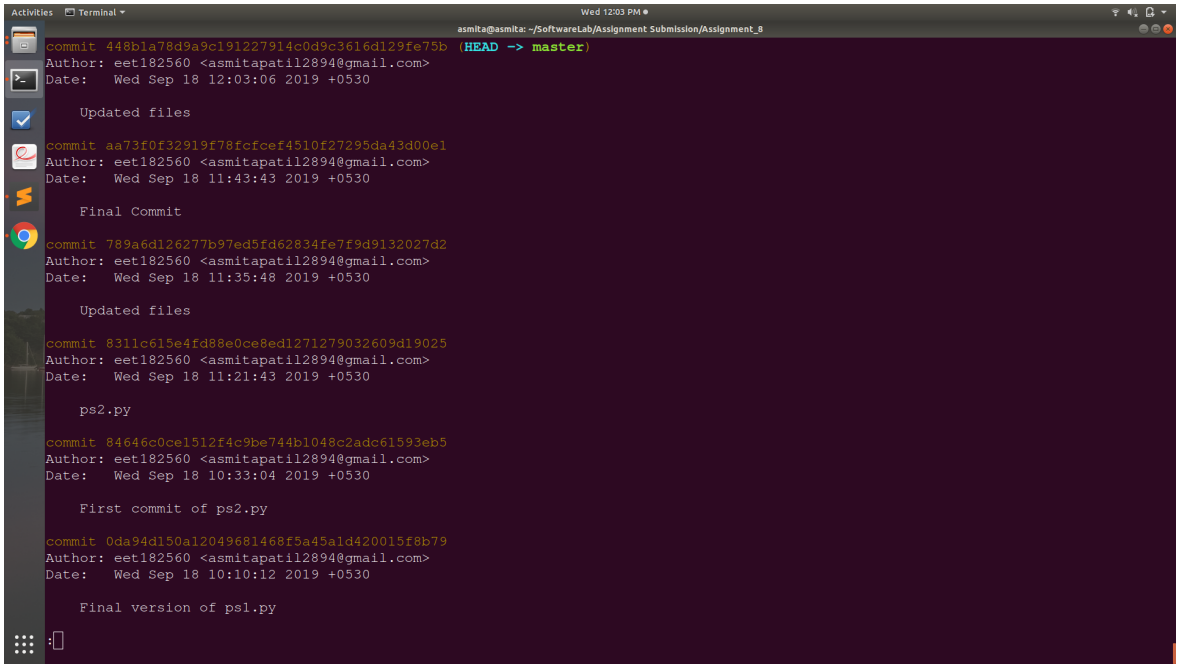


Figure 1.2: Execution of Git Command

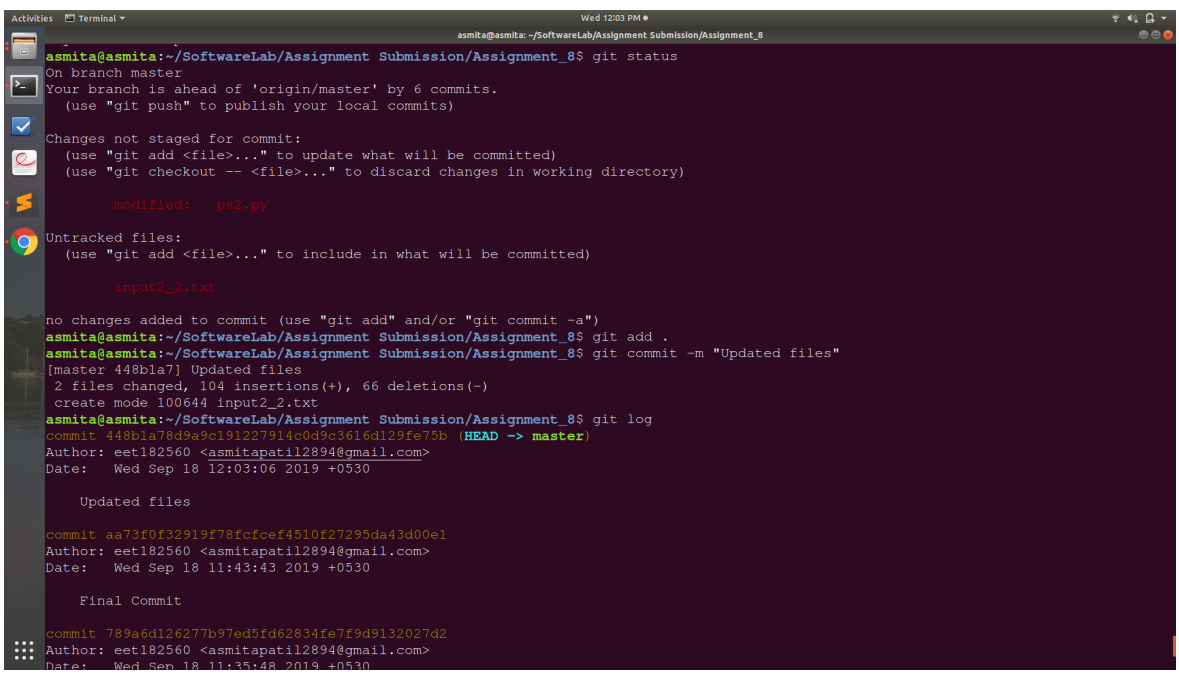


Figure 1.3: Execution of Git Command

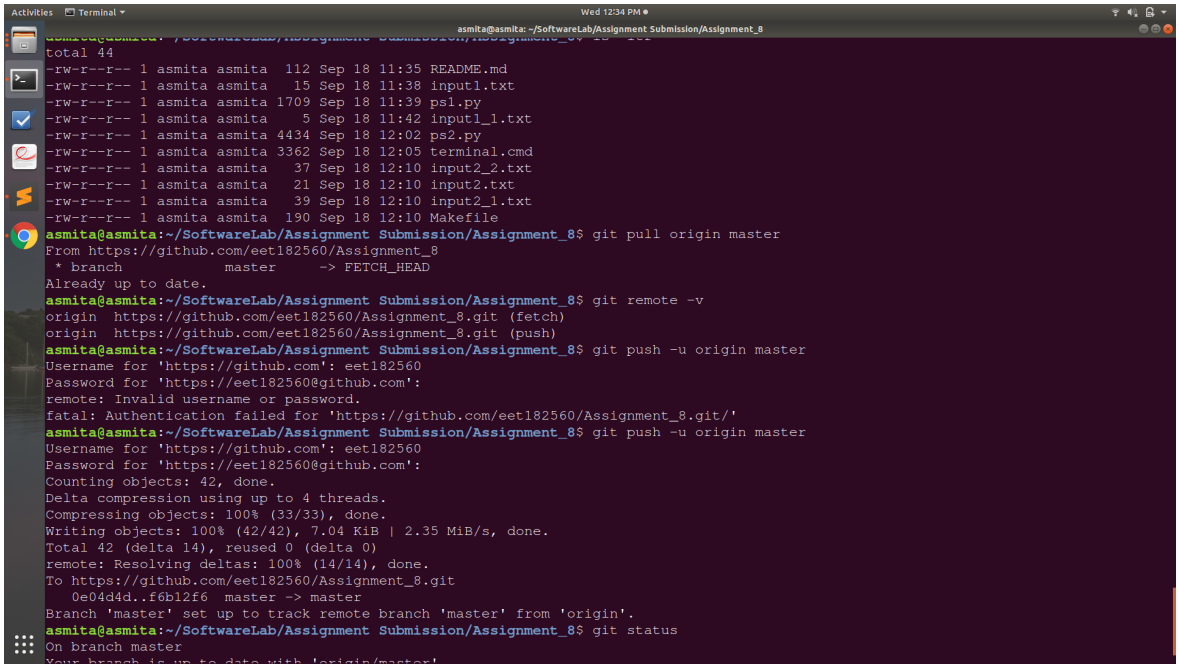


Figure 1.4: Execution of Git Command

2 Problem Statement-2

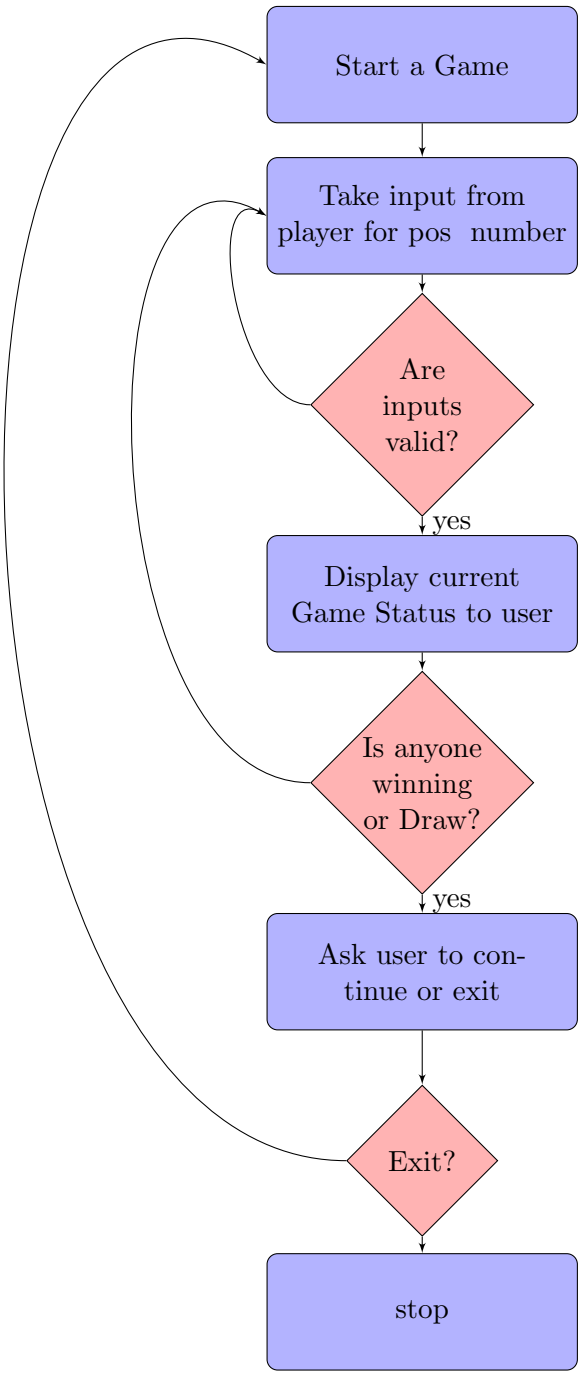
2.1 Problem Statement

- **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<=Position<=9
  - 1<=Number<=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

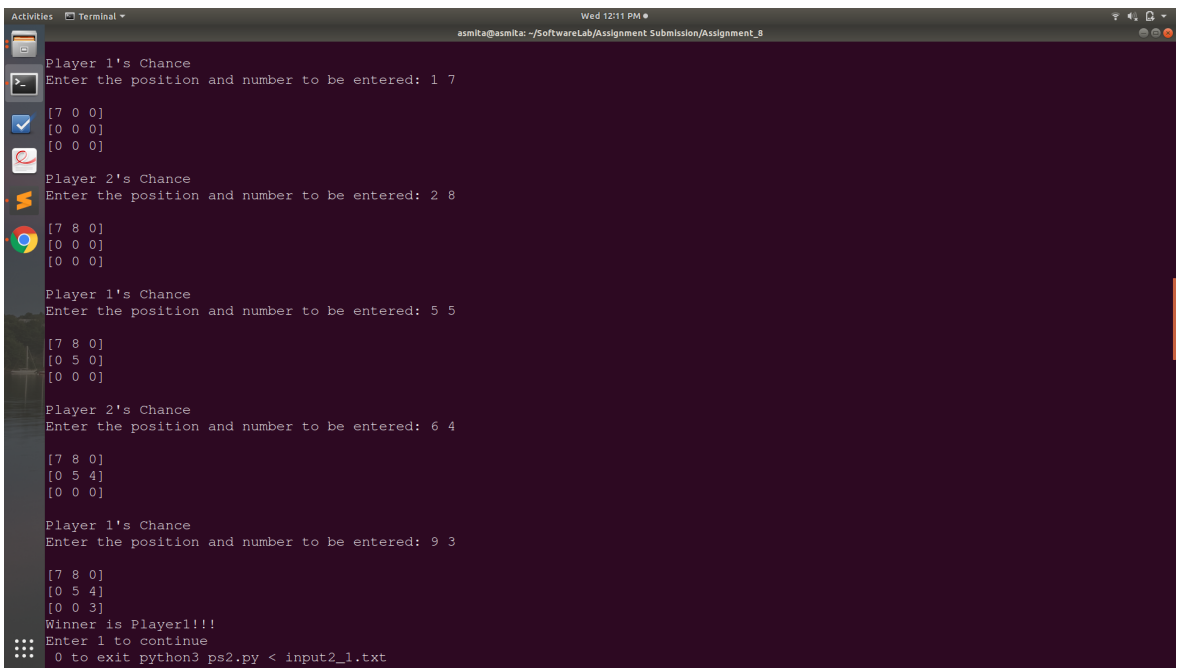
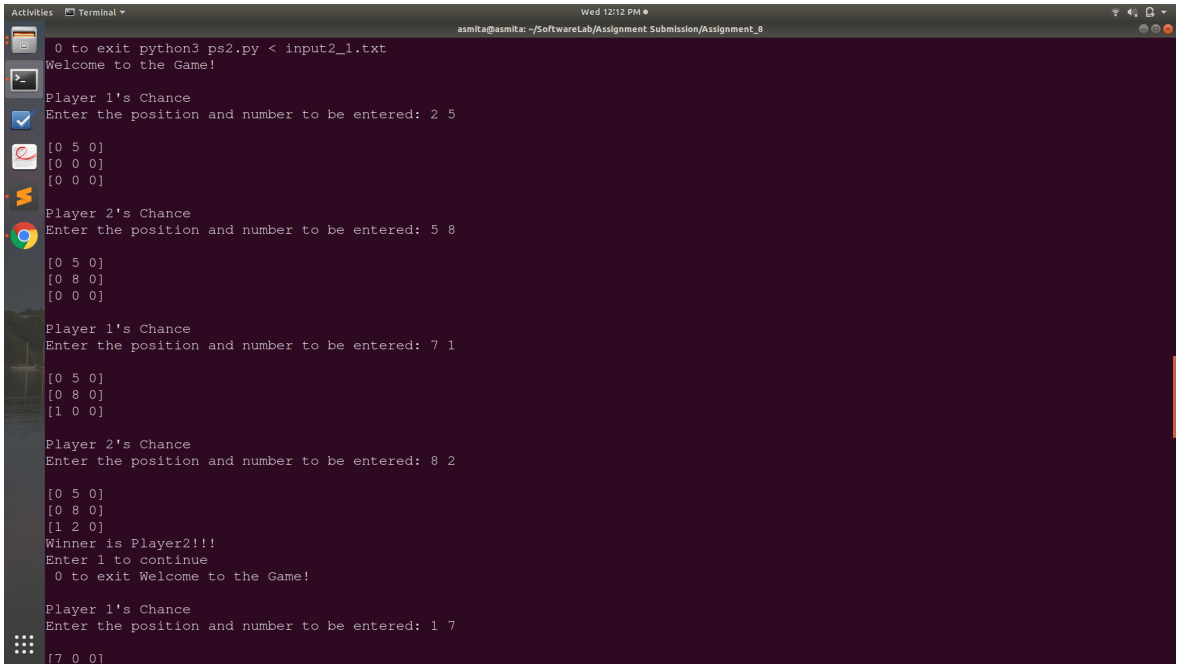
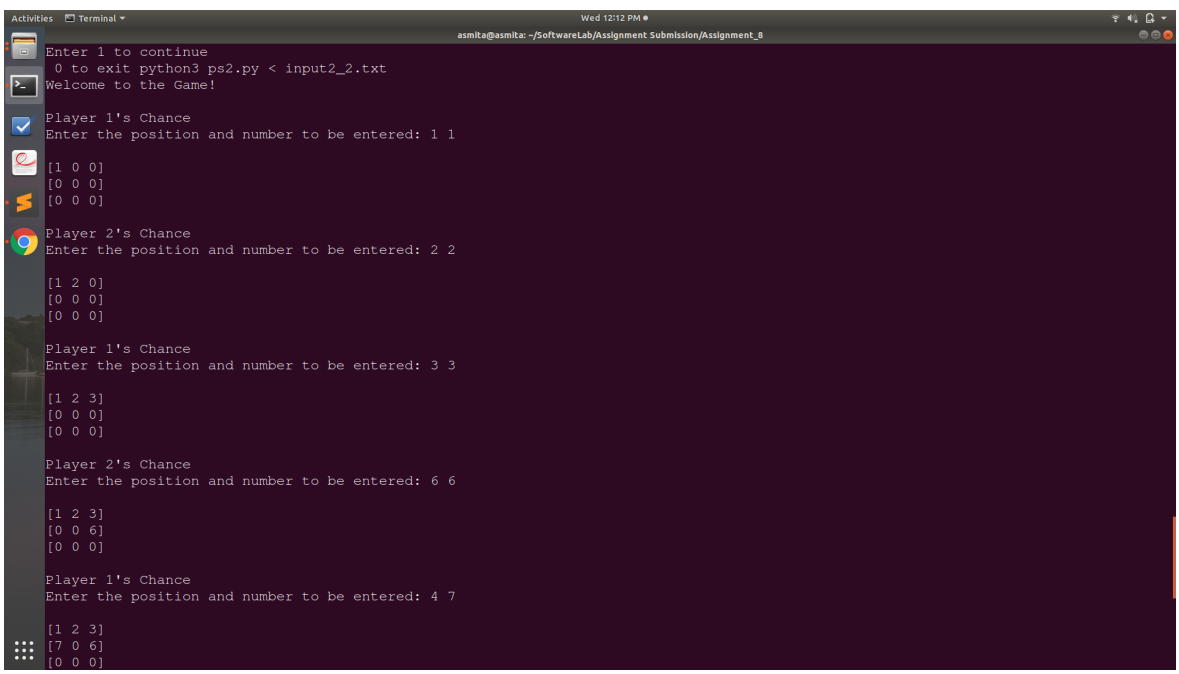


Figure 2.1: Execution of problem statement 2: Normal Exit



```
 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

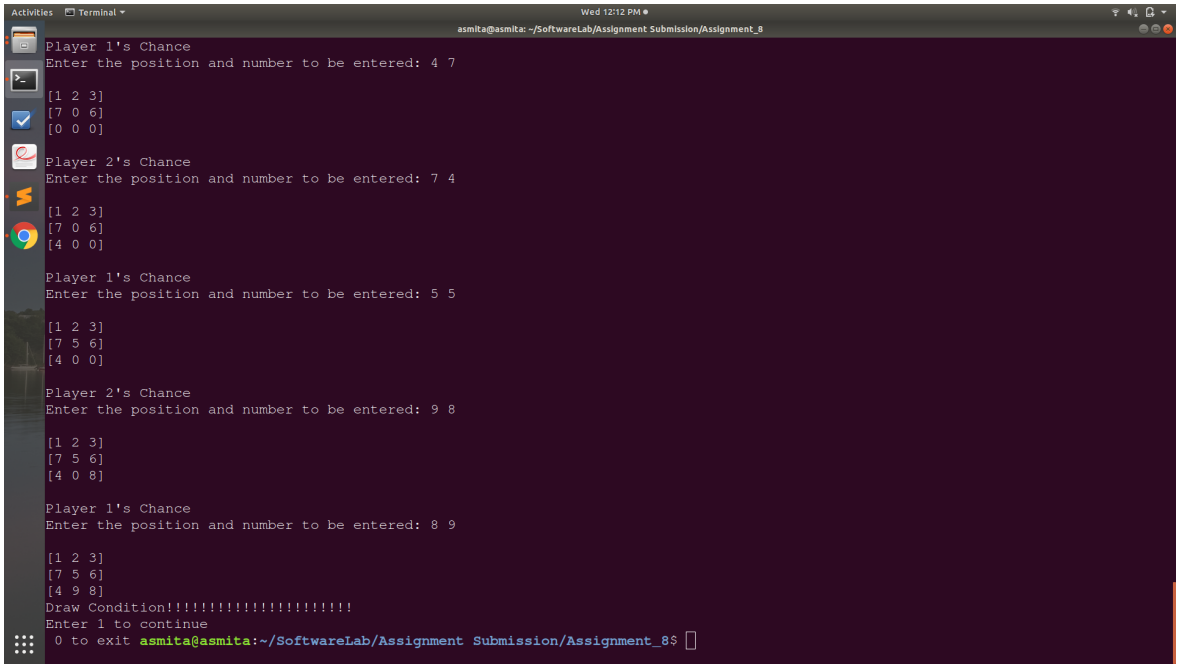


Figure 2.4: Execution of problem statement 2: Draw case2

Appendices

A Code for Problem Statement 1

```
# Function Defintions
# Function which check validity of string if it is binary or not
def checkVal(String):
  # Convert string into list
  newList = list(String)
  # Check string character by character
  for char in newList:
    if char == '0' or char == '1':
      continue
    else:
      return -1;

  # Return 0 if in case of valid string else -1
  return 0;


# Function to calculate parity of string
def calcParity(String):

  # Initialization of variables
  i = 0
  count = 0

  # Count numbers of 1's
  while i < len(String):
    if String[i] == '1':
      count += 1
    i += 1

  # Check number of 1's is odd or even
  if count % 2 == 0:
    count = 1
  else:
    count = 0

  # Return parity
  return count


# Function which generate transmission data after bit stuffing if required
def generateTransmissiondata(strParity):

  # Initialization of variables
  i = 0
```

```
  j = 0

  # Copying string into list to append
  newStr = list(strParity)

  # Check for pattern '010' in input data
  while i < len(strParity) - 3:

    # If pattern is found append it with 0
    if strParity[i : i + 3] == '010':
      newStr.insert(j + 3, '0')
      i += 2
      j += 3
    i += 1
    j += 1

  # Return string with flag appended '0101'
  return ''.join(newStr) + '0101'

## Execution start from here
# Take input from user
String = input("Enter the binary String :")
print(String)
#Check validity of String
if checkVal(String) == -1:
  print("Invalid input")
  exit(0)

# Calculate parity of string and print the string with parity
count = calcParity(String)
strParity = String + str(count)
print("Parity bit data : " + strParity)

# Calculate data to be transmitted and print the same
outPut = generateTransmissiondata(strParity)
print("Transmitting data: " + outPut)
```

B Code for Problem Statement 2

```
import numpy as np

# Function Declarations and Definitions
# Display the matrix / board
def display(mat):
  print()
  print(mat[0: 3])
  print(mat[3 : 6])
  print(mat[6 : 9])


# Function to check sum is 15 or not and exit based on it
def sumCheck(sum, player):
  if sum == 15:
    print("Winner is Player" + str(player) + "!!!")
    flag = input("Enter 1 to continue\n 0 to exit ")
    if flag == "1":
      Game()
    else:
      exit(0)


# Function to calculate sum rowwise based on pos
def rowSum(mat, pos, player, vis):

  #Initialization of variables
  sum = 0
  valid = False

  # Calculate sum for first row pos
  if pos in row1:
    sum = mat[0] + mat[1] + mat[2]
    if vis[0] and vis[1] and vis[2]:
      valid = True

  # Calculate sum for second row pos
  elif pos in row2:
    sum = mat[3] + mat[4] + mat[5]
    if vis[3] and vis[4] and vis[5]:
      valid = True

  # Calculate sum for third row pos
  else:
    sum = mat[6] + mat[7] + mat[8]
```

```
    if vis[6] and vis[7] and vis[8]:
      valid = True

  # Check validity of the sum and call sumCheck
  if valid:
    sumCheck(sum, player)


# Function to calculate sum column wise based on pos
def colSum(mat, pos, player, vis):

  #Initialization of variables
  sum = 0
  valid = False

  # Calculate sum for first column position
  if pos in col1:
    sum = mat[0] + mat[3] + mat[6]
    if vis[0] and vis[3] and vis[6]:
      valid = True

  # Calculate sum for second column position
  elif pos in col2:
    sum = mat[1] + mat[4] + mat[7]
    if vis[1] and vis[4] and vis[7]:
      valid = True

  # Calculate sum for third column position
  else:
    sum = mat[2] + mat[5] + mat[8]
    if vis[2] and vis[5] and vis[8]:
      valid = True

  # Check validity of the sum and call sumCheck
  if valid:
    sumCheck(sum, player)

# Function to calculate sum diagonally based on pos
def diagonal(mat, pos, player, vis):

  #Initialization of variables
  sum = 0
  valid = False

  # Calculate sum for second diagonal position
  if pos in diagonal1:
    sum = mat[0] + mat[4] + mat[8]
    if vis[0] and vis[4] and vis[8]:
      valid = True

  # Check validity of the sum and call sumCheck
  if valid:
    sumCheck(sum, player)

  # Calculate sum for second diagonal position
  if pos in diagonal2:
```

```
    sum = mat[2] + mat[4] + mat[6]
    if vis[2] and vis[4] and vis[6]:
      valid = True

  # Check validity of the sum and call sumCheck
  if valid:
    sumCheck(sum, player)


#Check for draw condition
def checkDraw(vis):
  flag = True
  for i in vis:
    if i == False:
      flag = False
      break

  # if draw then give user option to continue or exit
  if flag:
    print("Draw Condition!!!!!!!!!!!!!!!!!!!!!")
    flag = input("Enter 1 to continue\n 0 to exit ")
    if flag == "1":
      Game()
    else:
      exit(0)


# Function to start a game
def Game():

  # Initialization of all variables
  mat = np.zeros(9, dtype=np.int32)
  vis = [False] * 9
  numVis = [False] * 9

  # Start of game with player1
  print("Welcome to the Game!")
  player = 1

  # Infinite to execute game
  while True:

    # take user input for position and number to be used
    print()
    print("Player " + str(player) + "'s Chance")
    pos, num = input("Enter the position and number to be entered: ").split(",")
    print(pos, num)

    # Check validity of position
    pos = int(pos)
    if pos < 1 or pos > 9:
      print("Invalid position")
      continue

    # Check validity of number
```

```
    num = int(num)
    if num < 1 or num > 9:
      print("Invalid number")
      continue

    # Check validity of number based on player's chance
    if player == 1 and num not in player1:
      print("Please enter odd number")
      continue
    elif player == 2 and num not in player2:
      print("Please enter even number")
      continue

    # Check validity of number and position
    if vis[pos - 1] == True:
      print("Enter the position which is not used")
      continue
    if numVis[num - 1] == True:
      print("Enter the number which is not used yet")
      continue

    # update the variables for given input
    vis[pos - 1] = True
    numVis[num - 1] = True
    mat[pos - 1] = num

    # Display current board status
    display(mat)

    # Check for winner if exists end the game else continue
    rowSum(mat, pos, player, vis)
    colSum(mat, pos, player, vis)
    diagonal(mat, pos, player, vis)

    checkDraw(vis)

    # Switch the player
    if player == 1:
      player = 2
    else:
      player = 1


### Execution starts from here

player1 = {1, 3, 5, 7, 9}
player2 = {2, 4, 6, 8}
row1 = {1, 2, 3}
row2 = {4, 5, 6}
col1 = {1, 4, 7}
col2 = {2, 5, 8}
diagonal1 = {1, 5, 9}
diagonal2 = {3, 5, 7}
Game()
```

C Code for Makefile

```
all: ps1 ps2

ps1: ps1.py
  python3 ps1.py < input1.txt
  python3 ps1.py < input1_1.txt

ps2: ps2.py
  python3 ps2.py < input2.txt
  python3 ps2.py < input2_1.txt
  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.