ARTIFICIAL INTELLIGENCE LAB MANUAL

[ACADEMIC YEAR: 2018-19]

SEMESTER - I

COURSE NAME: ARTIFICIAL INTELLIGENCE

COURSE CODE: 15CS3111

YEAR: III/IV B. TECH

DEPT: COMPUTER SCIENCE & ENGINEERING

List of Experiments

1. (a). Write a python program to print the multiplication table for the given number?
   (b). Write a python program to check whether the given number is prime or not?
   (c) Write a python program to find factorial of the given number?

2. Write a python program to implement simple Chatbot?

3. (a) Write a python program to implement List operations (Nested List, Length, Concatenation, Membership, Iteration, Indexing and Slicing)?
   (b) Write a python program to implement List methods (Add, Append, Extend & Delete).

4. (a). Write a python program to Illustrate Different Set Operations?
   (b). Write a python program to generate Calendar for the given month and year?
   (c). Write a python program to implement Simple Calculator program?

5. (a). Write a python program to Add Two Matrices.
   (b). Write a python program to Transpose a Matrix.

6. Write a python program to implement Breadth First Search Traversal?

7. Write a python program to implement Water Jug Problem?
8. (a) Write a python program to remove punctuations from the given string?
   (b) Write a python program to sort the sentence in alphabetical order?
9. Write a program to implement Hangman game using python.

10. Write a program to implement Tic-Tac-Toe game using python.
11. (a) Write a python program to remove stop words for a given passage from a text file using NLTK?
   (b) Write a python program to implement stemming for a given sentence using NLTK?
   (c) Write a python program to POS (Parts of Speech) tagging for the give sentence using NLTK?

12. (a) Write a python program to implement Lemmatization using NLTK?
   (b) Write a python program to for Text Classification for the give sentence using NLTK?

1. **(a). Write a python program to print the multiplication table for the given number?**

Code:

```
 # Python program to find the multiplication table (from 1 to 10) of a number input by the user

# take input from the user

num = int(input("Display multiplication table of? "))

    # use for loop to iterate 10 times

for i in range(1,11):

   print(num,'x',i,'=',num*i)
```

Output:

```
Display multiplication table of? 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

(b). Write a python program to check whether the given number is prime or not?

Code:

```
# Python program to check if the input number is prime or not

#num = 407

# take input from the user
num = int(input("Enter a number: "))

# prime numbers are greater than 1
if num > 1:
  rem=1

  for i in range(2,num):
     rem=num%i
     if rem == 0:
       break
```

```
        if rem == 0:
           print(num,"is not a prime number")
        else:
           print(num,"is a prime number")

    # if input number is less than
    # or equal to 1, it is not prime
    else:
       print(num,"is not a prime number")
```

Output:

Enter a number: 5

5 is a prime number

(c). Write a python program to find factorial of the given number?

Code:

```
def recur_factorial(n):
    """Function to return the factorial
    of a number using recursion"""
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)

# Change this value for a different result
#num = 5
# uncomment to take input from the user
num = int(input("Enter a number: "))
# check is the number is negative
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
```

```
print("The factorial of",num,"is",recur_factorial(num))
```

Output:

```
Enter a number: 5
The factorial of 5 is 120
```

2. **Write a python program to implement simple Chatbot?**

Code:

```
print("Simple Question and Answering Program")
print("====================================")
print(" You may ask any one of these questions")
print("Hi")
print("How are you?")
print("Are you working?")
print("What is your name?")
print("what did you do yesterday?")
print("Quit")
while True:
    question = input("Enter one question from above list:")
    question = question.lower()
     if question in ['hi']:
        print("Hello")
    elif question in ['how are you?','how do you do?']:
         print("I am fine")
    elif question in ['are you working?','are you doing any job?']:
         print("yes. I'am working in KLU")
    elif question in ['what is your name?']:
        print("My name is Emilia")
        name=input("Enter your name?")
```

```
            print("Nice name and Nice meeting you",name)
        elif question in ['what did you do yesterday?']:
            print("I saw Bahubali 5 times")


        elif question in ['quit']:
             break
        else:
           print("I don't understand what you said")
```

Output:

```
Simple Question and Answering Program
==============================
 You may ask any one of these questions
Hi
How are you?
Are you working?
What is your name?
what did you do yesterday?
Quit
Enter one question from above list:hi
Hello
Enter one question from above list:how are you?
I am fine
Enter one question from above list:are you working?
yes. I'am working in KLU
Enter one question from above list:what is your name?
My name is Emilia
Enter your name?sachin
```

Nice name and Nice meeting you sachin

Enter one question from above list:quit

3. (a) Write a python program to implement list operations (Nested List, Length, Concatenation,Membership,Iteration,Indexing and Slicing)?

Code:

```
my_list = ['p','r','o','b','e']
# Output: p
print(my_list[0])
print("Length",my_list,":",len(my_list))
# Output: o
print(my_list[2])
# Output: e
print(my_list[4])
# Error! Only integer can be used for indexing
#my_list[4.2]
# Nested List
n_list = [[1,3,5,7], [2,4,6,8,10]]
# Nested indexing
# Output: 3
print(n_list[0][1])
# Output: 8
print(n_list[1][3])
# Nested List2
n_list2 = ["Happy", [2,0,1,5]]
```

```
# Nested indexing
# Output: a
print(n_list2[0][1])
# Output: 5
print(n_list2[1][3])
concat1=[1,3,5]
concat2=[7,8,9]
print("Concatenation:",concat1,concat2,":",concat1+concat2)
repetion_string="Hello"
print("Repetition:",repetion_string,repetion_string * 4)
```

Output:

```
p
Length ['p', 'r', 'o', 'b', 'e']: 5
o
e
3
8
a
5
Concatenation: [1, 3, 5] [7, 8, 9] : [1, 3, 5, 7, 8, 9]
Repetition: Hello HelloHelloHelloHello
```

(b) Write a python program to implement list operations (add, append, extend & delete)

Code:

```
myList = ["Earth", "revolves", "around", "Sun"]

print(myList)

print(len(myList))

print(myList[0])

print(myList[3])

#Slice elements from a list

print(myList[1:3])

#Use negative index

print(myList[-1])

#print(myList[4])

#add element to a list

myList.insert(0,"The")

print(myList)

print(len(myList))

myList.insert(4,"the")

print(myList)

#append an element to a list

myList.append("continuously")

print(myList)

print(len(myList))

#When use extend the argument should be another list

#the elements of that list will be added

#to the current list as individual elements

myList.extend(["for", "sure"])
```

```
print(myList)
print(len(myList))
#you can append a sublist to the current list using append
myList.append(["The","earth","rotates"])
print(myList)
print(len(myList))
#delete a element in the list using element
myList.remove("The")
#delete a element in the list using index
myList.remove(myList[3])
print(myList)
```

Output

```
['Earth', 'revolves', 'around', 'Sun']
4
Earth
Sun
['revolves', 'around']
Sun
['The', 'Earth', 'revolves', 'around', 'Sun']
5
['The', 'Earth', 'revolves', 'around', 'the', 'Sun']
['The', 'Earth', 'revolves', 'around', 'the', 'Sun', 'continuously']
7
['The', 'Earth', 'revolves', 'around', 'the', 'Sun', 'continuously', 'for', 'sure']
9
['The', 'Earth', 'revolves', 'around', 'the', 'Sun', 'continuously', 'for', 'sure', ['The',
'earth', 'rotates']]
```

10

['Earth', 'revolves', 'around', 'Sun', 'continuously', 'for', 'sure', ['The', 'earth', 'rotates']]

4 (a). Write a python program to Illustrate Different Set Operations?

Code:

```
# Program to perform different set operations like in mathematics

# define three sets

E = {0, 2, 4, 6, 8};

N = {1, 2, 3, 4, 5};

# set union

print("Union of E and N is",E | N)

# set intersection

print("Intersection of E and N is",E & N)

# set difference

print("Difference of E and N is",E - N)

# set symmetric difference

print("Symmetric difference of E and N is",E ^ N)
```

Output:

Union of E and N is {0, 1, 2, 3, 4, 5, 6, 8}

Intersection of E and N is {2, 4}

Difference of E and N is {0, 8, 6}

Symmetric difference of E and N is {0, 1, 3, 5, 6, 8}

(b). Write a python program to generate Calendar for the given month and year?

Code:

```
# Python program to display calendar of given month of the year

# import module
import calendar
```

```
yy = 2014
mm = 11

# To ask month and year from the user# Python program to display calendar of given month of the year

# import module
import calendar

yy = 2014
mm = 11

# To ask month and year from the user
# yy = int(input("Enter year: "))
# mm = int(input("Enter month: "))

# display the calendar
print(calendar.month(yy, mm))
# yy = int(input("Enter year: "))
# mm = int(input("Enter month: "))

# display the calendar
print(calendar.month(yy, mm))
```

Output:

```
November 2014
Mo Tu We Th Fr Sa Su
             1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

(c). Write a python program to implement Simple Calculator program?

Code:

```
# Program make a simple calculator that can add, subtract, multiply and divide using functions

  # define functions

def add(x, y):

      """This function adds two numbers"""
```

```
    return x + y
def subtract(x, y):
    """This function subtracts two numbers"""
        return x - y
def multiply(x, y):
    """This function multiplies two numbers"""


    return x * y
def divide(x, y):
    """This function divides two numbers"""
        return x / y
# take input from the user
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
choice = input("Enter choice(1/2/3/4):")
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
    if choice == '1':
   print(num1,"+",num2,"=", add(num1,num2))
  elif choice == '2':
    print(num1,"-",num2,"=", subtract(num1,num2))
elif choice == '3':
    print(num1,"*",num2,"=", multiply(num1,num2))
```

```
elif choice == '4':
    print(num1,"/",num2,"=", divide(num1,num2))
else:
    print("Invalid input")
```

Output:

```
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):1
Enter first number: 21
Enter second number: 22
21 + 22 = 43
```

5. (a). Write a python program to Add Two Matrices.

Code:

```
# Program to add two matrices using nested loop
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]
Z = [[0,0,0],
    [0,0,0],
    [0,0,0]]
```

```
# iterate through rows
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       Z[i][j] = X[i][j] + Y[i][j]
for r in Z:
   print(r)
```

Output:

```
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
```

(b). Write a python program to Transpose a Matrix.

Code:

```
 # Program to transpose a matrix using nested loop
X = [[12,7],
    [4 ,5],
    [3 ,8]]


result = [[0,0,0],
          [0,0,0]]
# iterate through rows
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       result[j][i] = X[i][j]
for r in result:
```

```
print(r)
```

Output:

[12, 4, 3]

[7, 5, 8]

6.Write a python program to implement Breadth first search Traversal?

Code:

```
"""
This file contains an implementation of breadth-first search (BFS) for
traversing a graph, and for getting the shortest path between 2 nodes
of a graph.
"""
# visits all the nodes of a graph (connected component) using BFS
def bfs_connected_component(graph, start):
    # keep track of all visited nodes
    explored = []
    # keep track of nodes to be checked
    queue = [start]

    # keep looping until there are nodes still to be checked
    while queue:
        # pop shallowest node (first node) from queue
        node = queue.pop(0)
        if node not in explored:
            # add node to list of checked nodes
            explored.append(node)
            neighbours = graph[node]
```

```
            # add neighbours of node to queue
            for neighbour in neighbours:
                queue.append(neighbour)
    return explored
# finds shortest path between 2 nodes of a graph using BFS
def bfs_shortest_path(graph, start, goal):
    # keep track of explored nodes
    explored = []
    # keep track of all the paths to be checked
    queue = [[start]]
    # return path if start is goal
    if start == goal:
        return "That was easy! Start = goal"
    # keeps looping until all possible paths have been checked
    while queue:
        # pop the first path from the queue
        path = queue.pop(0)
        # get the last node from the path
        node = path[-1]
        if node not in explored:
            neighbours = graph[node]
            # go through all neighbour nodes, construct a new path and
            # push it into the queue
            for neighbour in neighbours:
                new_path = list(path)
                new_path.append(neighbour)
                queue.append(new_path)
```

```
            # return path if neighbour is goal
            if neighbour == goal:
                return new_path
        # mark node as explored
        explored.append(node)
    # in case there's no path between the 2 nodes
    return "So sorry, but a connecting path doesn't exist :("
if __name__ == '__main__':
    # sample graph represented by a dictionary
   graph = {'A': ['B', 'C'],
        'B': ['A','D'],
        'C': ['A','E','F'],
        'D': ['B'],
        'E': ['C',],
        'F': ['C'],
        }

#            A
#           / \
#          /   \
#         B     C
#        /     /\
#       /     /  \
#      D     E    F

   print("\nHere's the nodes of the graph visited by "
        "breadth-first search, starting from node 'A': ",
```

```
    bfs_connected_component(graph, 'A'))

print("\nHere's the shortest path between nodes 'D' and 'F':",

    bfs_shortest_path(graph, 'D', 'F'))
```

OUTPUT: -

HERE'S THE NODES OF THE GRAPH VISITED BY BREADTH-FIRST SEARCH, STARTING FROM NODE 'A': ['A', 'B', 'C', 'D', 'E', 'F']

HERE'S THE SHORTEST PATH BETWEEN NODES 'D' AND 'F': ['D', 'B', 'A', 'C', 'F']

7. Write a python program to implement Water Jug Problem?

A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

Let X represent the content of the water in 4-gallon jug.
Let Y represent the content of the water in 3-gallon jug.

Write a program in python to define a set of operators (Rules) that will take us from one state to another:

Start from initial state (X=0, Y=0)
Reach any of the Goal states
(X=2, Y=0)
(X=2, Y=1)
(X=2, Y=2)
(X=2, Y=3)
Find the minimum number of steps to reach any the above mentioned goal states.

Code:

```
print("Water Jug Problem")
x=int(input("Enter X:"))
y=int(input("Enter Y:"))
while True:

    rno=int(input("Enter the Rule No"))

    if rno==1:
       if x<4:x=4

    if rno==2:
       if y<3:y=3
```

```
if rno==5:
   if x>0:x=0

if rno==6:
   if y>0:y=0

if rno==7:
   if x+y>= 4 and y>0:x,y=4,y-(4-x)

if rno==8:
   if x+y>=3 and x>0:x,y=x-(3-y),3

if rno==9:
   if x+y<=4 and y>0:x,y=x+y,0

if rno==10:
   if x+y<=3 and x>0:x,y=0,x+y


print("X =" ,x)
print("Y =" ,y)
if (x==2):
  print(" The result is a Goal state")
  break
```

Output:

```
Water Jug Problem
Enter X:0
Enter Y:0
Enter the Rule No2
X = 0
Y = 3
Enter the Rule No9
X = 3
Y = 0
Enter the Rule No2
X = 3
Y = 3
Enter the Rule No7
X = 4
Y = 2
Enter the Rule No5
X = 0
Y = 2
```

Enter the Rule No9
X = 2
Y = 0
The result is a Goal state.

8. (a) Write a python program to remove punctuations from the given string?

Code:

```
 # define punctuation

punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''

my_str = "Hello!!!, he said ---and went."

# To take input from the user

# my_str = input("Enter a string: ")

# remove punctuation from the string

no_punct = ""

for char in my_str:

  if char not in punctuations:

      no_punct = no_punct + char

# display the unpunctuated string

print(no_punct)
```

Output:

Hello he said and went

(b) Write a python program to sort the sentence in alphabetical order?

Code:

```
 # Program to sort alphabetically the words form a string provided by the user

 # change this value for a different result

 my_str = "Hello this Is an Example With cased letters"

 # uncomment to take input from the user

 #my_str = input("Enter a string: ")

 # breakdown the string into a list of words
```

```
words = my_str.split()
#print(words)
# sort the list
words.sort()
# display the sorted words
print("The sorted words are:")
for word in words:
  print(word)
```

Output:

```
The sorted words are:
Example
Hello
Is
With
an
cased
letters
this
```

9. Write a program to implement Hangman game using python.

Description:

Hangman is a classic word-guessing game. The user should guess the word correctly by entering alphabets of the user choice. The Program will get input as single alphabet from the user and it will matchmaking with the alphabets in the original word. If the user given alphabet matches with the alphabet from the original word then user must guess the remaining alphabets. Once the user successfully found the word he will be declared as winner otherwise user lose the game.

Python Code:

```
def getAvailableLetters(lettersGuessed):
    '''
    lettersGuessed: list, what letters have been guessed so far
    returns: string, comprised of letters that represents what letters have not
      yet been guessed.
    '''
    import string
    fullstring = string.ascii_lowercase
    lettersLeft = ''
    for letter in fullstring:
        if letter not in lettersGuessed:
            lettersLeft = lettersLeft + letter
    return lettersLeft


def getGuessedWord(secretWord, lettersGuessed):
    '''
    secretWord: string, the word the user is guessing
    lettersGuessed: list, what letters have been guessed so far
    returns: string, comprised of letters and underscores that represents
      what letters in secretWord have been guessed so far.
```

```
    '''
    wordGuessed = ''
    for letter in secretWord:
        if letter in lettersGuessed:
            wordGuessed = wordGuessed + letter
        else:
            wordGuessed = wordGuessed + '_ '
    return wordGuessed


def isWordGuessed(secretWord, lettersGuessed):
    '''
    secretWord: string, the word the user is guessing
    lettersGuessed: list, what letters have been guessed so far
    returns: boolean, True if all the letters of secretWord are in lettersGuessed;
      False otherwise
    '''
    numCorrect = 0
    for letter in secretWord:
        if letter in lettersGuessed:
            numCorrect += 1
        else:
            return False
    return True
def hangman(secretWord):
    guessesLeft = 8
    lettersGuessed =[]
    print('Welcome to the game Hangman!')
```

```
    print('I am thinking of a word that is ' + str(len(secretWord)) + ' letters long.' )
    print('-----------')
    while guessesLeft > 0:
        if isWordGuessed(secretWord, lettersGuessed):
            return print('Congratulations, you won!')
        print('You have ' + str(guessesLeft) + ' guesses left.')
        print('Available Letters: ' + getAvailableLetters(lettersGuessed))
        user_input = input('Please guess a letter: ')
        user_input = str(user_input)
        user_input = user_input.lower()
        if user_input not in lettersGuessed:
            lettersGuessed.append(user_input)
            if user_input in secretWord:
                print("Good guess: " + getGuessedWord(secretWord, lettersGuessed))
                print('-----------')
            else:
                print("Oops! That letter is not in my word: " + getGuessedWord(secretWord,
lettersGuessed))
                print('-----------')
                guessesLeft -= 1
        else:
            print("Oops! You've already guessed that letter: " + getGuessedWord(secretWord,
lettersGuessed))
            print('-----------')
    return print("Sorry, you ran out of guesses. The word was " + str(secretWord))
hangman('apple')
```

Output:

Welcome to the game Hangman!

I am thinking of a word that is 5 letters long.

You have 8 guesses left.

Available Letters: abcdefghijklmnopqrstuvwxyz

Please guess a letter: a

Good guess: a_ _ _ _

You have 8 guesses left.

Available Letters: bcdefghijklmnopqrstuvwxyz

Please guess a letter: p

Good guess: app_ _

You have 8 guesses left.

Available Letters: bcdefghijklmnoqrstuvwxyz

Please guess a letter: p

Oops! You've already guessed that letter: app_ _

You have 8 guesses left.

Available Letters: bcdefghijklmnoqrstuvwxyz

Please guess a letter: l

Good guess: appl_

You have 8 guesses left.

Available Letters: bcdefghijkmnoqrstuvwxyz

Please guess a letter: e

Good guess: apple

Congratulations, you won!

10 . Write a program to implement Tic-Tac-Toe game using python.

Description:

In the Tic Tac Toe computer program the player chooses if they want to be X or O. Who takes the first turn is randomly chosen. Then the player and computer take turns making moves. The boxes on the left side of the flow chart are what happens during the player's turn. The right side shows what happens on the computer's turn. After the player or computer makes a move, the program checks if they won or caused a tie, and then the game switches turns. After the game is over, the program asks the player if they want to play again.

Python Code:

```
# Tic Tac Toe

import random

def drawBoard(board):

    # 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('   |   |')
```

```
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 as the second.
    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 tuple is the player's letter, the second is the computer's letter.
    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 returns False.
    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 player has won.
```

```
    # We use bo instead of board and le instead of letter so we don't have to type as much.
    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 his 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 that move.
    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 his 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
    # Move on one of the sides.
    return chooseRandomMoveFromList(board, [2, 4, 6, 8])
def isBoardFull(board):
    # Return True if every space on the board has been taken. Otherwise return False.
    for i in range(1, 10):
        if isSpaceFree(board, i):
            return False
    return True
print('Welcome to Tic Tac Toe!')
while True:
    # Reset the board
    theBoard = [' '] * 10
    playerLetter, computerLetter = inputPlayerLetter()
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')
```

```
    gameIsPlaying = True
    while gameIsPlaying:
        if turn == 'player':
            # Player's turn.
            drawBoard(theBoard)
            move = getPlayerMove(theBoard)
            makeMove(theBoard, playerLetter, move)
            if isWinner(theBoard, playerLetter):
                drawBoard(theBoard)
                print('Hooray! You have won the game!')
                gameIsPlaying = False
            else:
                if isBoardFull(theBoard):
                    drawBoard(theBoard)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer'
        else:
            # Computer's turn.
            move = getComputerMove(theBoard, computerLetter)
            makeMove(theBoard, computerLetter, move)

            if isWinner(theBoard, computerLetter):
                drawBoard(theBoard)
                print('The computer has beaten you! You lose.')
                gameIsPlaying = False
```

```
            else:
                if isBoardFull(theBoard):
                    drawBoard(theBoard)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'player'
    if not playAgain():
        break
```

OUTPUT: -

```
 WELCOME TO TIC TAC TOE!
DO YOU WANT TO BE X OR O?
X
THE PLAYER WILL GO FIRST.
   |     |
   |     |
   |     |
-----------
   |     |
   |     |
   |     |
-----------
   |     |
   |     |
   |     |
WHAT IS YOUR NEXT MOVE? (1-9)
```

```
5
  |     |
  |     |
  |     |
-----------
   |     |
   | X  |
   |    |
-----------
    |     |
 O |     |
    |     |


WHAT IS YOUR NEXT MOVE? (1-9)
3
    |     |
 O |     |
    |     |
-----------
    |     |
    | X   |
    |     |
-----------
    |     |
 O |     |  X
    |     |
WHAT IS YOUR NEXT MOVE? (1-9)
```

4

```
   |     |
 O |     |
   |     |
-----------
   |     |
 X | X   | O
   |     |
-----------
   |     |
 O |     | X
   |     |
```

WHAT IS YOUR NEXT MOVE? (1-9)

3

WHAT IS YOUR NEXT MOVE? (1-9)

8

```
   |     |
 O | X   |
   |     |
-----------
   |     |
 X | X   | O
   |     |
-----------
   |     |
 O | O   | X
   |     |
```

WHAT IS YOUR NEXT MOVE? (1-9)

3

WHAT IS YOUR NEXT MOVE? (1-9)

2

WHAT IS YOUR NEXT MOVE? (1-9)

9

```
   |   |
 O | X | X
   |   |
-----------
   |   |
 X | X | O
   |   |
-----------
   |   |
 O | O | X
   |   |
```

THE GAME IS A TIE!

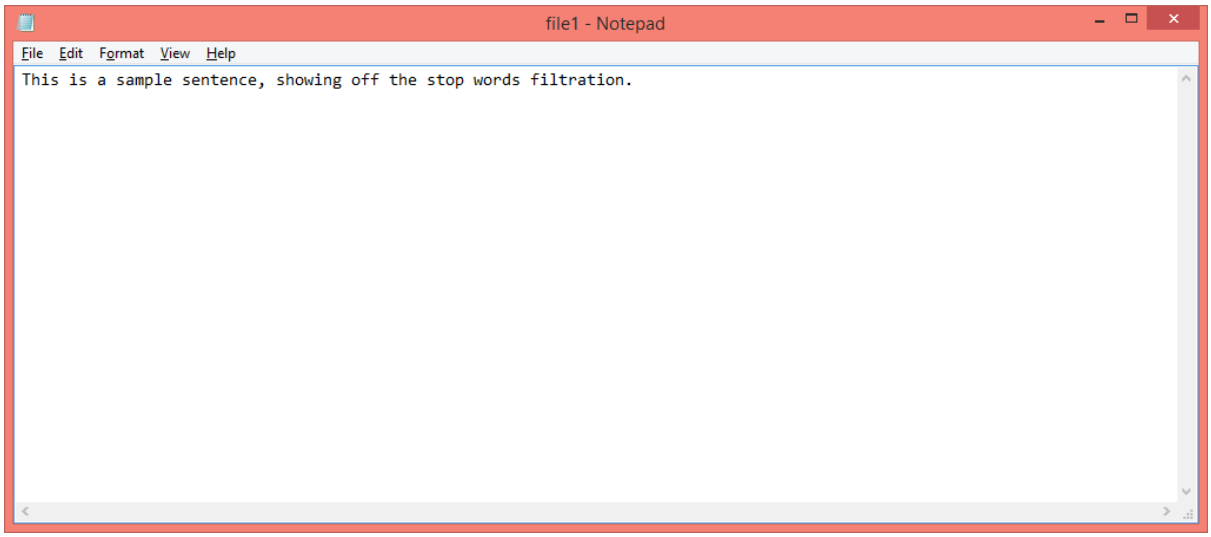
DO YOU WANT TO PLAY AGAIN? (YES OR NO)

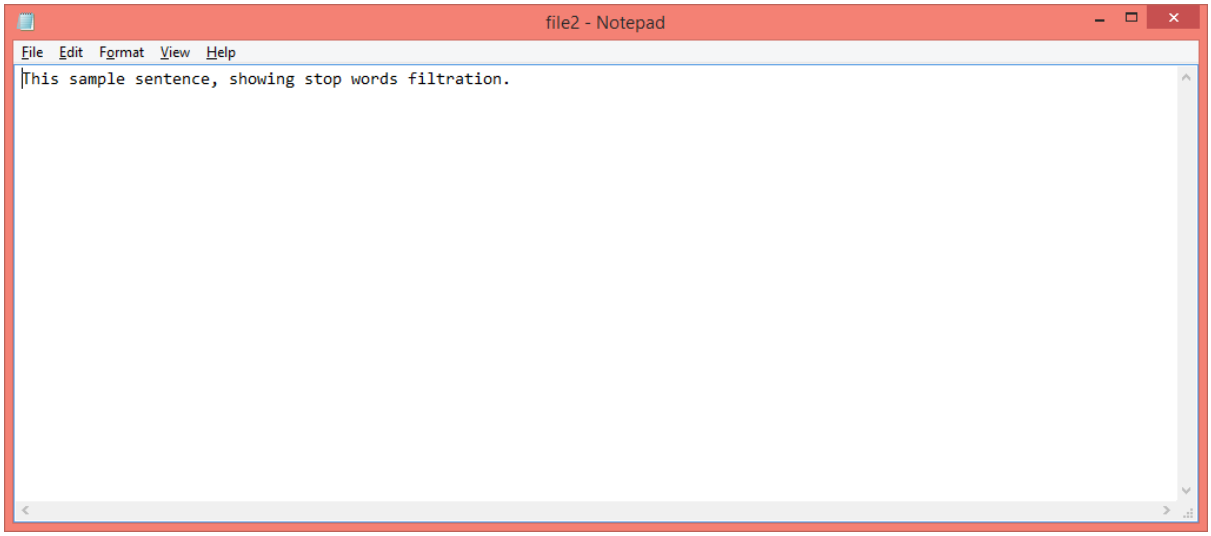
11.(a) Write a python program to remove stop words for a given passage from a text file using NLTK?

Code:

```
import nltk
from nltk.corpus import stopwords
f1 = open("file1.txt","r")
f2 = open("file2.txt","w")
stop = stopwords.words('english')
for line in f1:
    w = line.split(" ")
    for word in w:
        if word not in stop:
            f2.write(word)
            f2.write(" ")
f1.close()
f2.close()
```

Output:





(b) Write a python program to implement stemming for a given sentence using NLTK?

Code:

```
from nltk.stem import PorterStemmer

from nltk.tokenize import word_tokenize

ps= PorterStemmer()

example_words = [" python,pythonly,phythoner,pythonly"]

for w in example_words:

    print(ps.stem(w))
```

Output:

python,pythonly,phythoner,pythonli

(c) Write a python program to POS (Parts of Speech) tagging for the give sentence using NLTK?

Code:

```
import nltk

text = nltk.word_tokenize('ive into NLTK: Part-of-speech tagging and POS Tagger')

pos = nltk.pos_tag(text)

print(text)

print(pos)
```

Output:

['ive', 'into', 'NLTK', ':', 'Part-of-speech', 'tagging', 'and', 'POS', 'Tagger']

[('ive', 'JJ'), ('into', 'IN'), ('NLTK', 'NNP'), (':', ':'), ('Part-of-speech', 'JJ'), ('tagging', 'NN'), ('and', 'CC'), ('POS', 'NNP'), ('Tagger', 'NNP')]

12. (a) Write a python program to implement Lemmatization using NLTK?

Code:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("cats"))

print(lemmatizer.lemmatize("cacti"))

print(lemmatizer.lemmatize("geese"))

print(lemmatizer.lemmatize("rocks"))

print(lemmatizer.lemmatize("pyth"))

print(lemmatizer.lemmatize("cats"))
```

Output:

```
cat

cactus

goose

rock

pyth

cat
```

(b) Write a python program to for Text Classification for the give sentence using NLTK?

Code:

```
import nltk

import random
```

```
from nltk.corpus import movie_reviews
documents = [(list(movie_reviews.words(fileid)), category)
        for category in movie_reviews.categories()
        for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
print(documents[1])
all_words = []
for w in movie_reviews.words():
   all_words.append(w.lower())
all_words = nltk.FreqDist(all_words)
print(all_words.most_common(15))
print(all_words["stupid"])
```

Output:

([u'making', u'your', u'first', u'feature', u'film', u'ain', u'"', u't', u'easy', u'.', u'assemble', u'a', u'decent', u',', u'if', u'not', u',', u'strong', u'cast', u',', u'as', u'writer', u'/', u'director', u'robert', u'moresco', u'has', u'done', u'with', u'one', u'eyed', u'king', u',', u'and', u'you', u'"', u're', u'already', u'ahead', u'of', u'the', u'game', u'.', u'but', u'rehash', u'old', u'plot', u'lines', u',', u'tired', u'dialogue', u',', u'and', u'standard', u'clich', u'?', u's', u',', u'and', u'a', u'well', u'-', u'intentioned', u'effort', u'such', u'as', u'this', u'one', u'could', u'jeopardize', u'your', u'chance', u'at', u'a', u'second', u'feature', u'film', u'.', u'how', u'many', u'more', u'movies', u'do', u'we', u'need', u'about', u'a', u'rough', u'neighborhood', u'full', u'of', u'lifelong', u'friends', u'hopelessly', u'turned', u'to', u'crime', u'or', u'worse', u'?', u'the', u'enormous', u'catalog', u'of', u'such', u'movies', u'might', u'dissuade', u'a', u'filmmaker', u'from', u'making', u'yet', u'another', u',', u'but', u'here', u'we', u'have', u'it', u'.', u'again', u'.', u'five', u'irish', u'kids', u'in', u'nyc', u'"', u's', u'hell', u'"', u's', u'kitchen', u'make', u'an', u'overemotional', u'pact', u'over', u'some', u'stolen', u'rings', u'on', u'an', u'anonymous', , u'and', u'slow', u'motion', u'.', u'in', u'the', u'film', u'"', u's', u'first', u'scene', u'.', , '], u'neg')