**NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY**



**Artificial Intelligence Lab File**

**(2nd Year/3rd Semester)**

**COMPUTER SCIENCE & ENGINEERING**

**(AI – B)**

**SUBMITTED BY: SUBMITTED TO:**

Name: Rishav Kumar Dr. Vikas Sagar

Roll no.:2101331520109

Question-1

*To make simple chatboat using NLTK Libary in python*

import nltk

from nltk.chat.util import Chat, reflections

reflections = {

"i am" : "you are",

"i was" : "you were",

"i" : "you",

"i'm" : "you are",

"i'd" : "you would",

"i've" : "you have",

"i'll" : "you will",

"my" : "your",

"you are" : "I am",

"you were" : "I was",

"you've" : "I have",

"you'll" : "I will",

"your" : "my",

"yours" : "mine",

"you" : "me",

"me" : "you"

}

pairs = [

[

r"my name is (.\*)",

["Hello %1 ,I am NIET Chatbot , How can i assist you ?",]

],

[

r"hi|hey|hello",

["Hello, Wellcome to NIET Greater Noida!\nPlease enter your course",]

],

[

r"tell me about your college?",

["Accredited by NAAC(A Grade, 3.23) and NBA(CSE, ECE, IT, BT, ME, MBA, MCA and B.Pharm)\nIt has been ranked 43rd in NIET Pharmacy Institute and 171st rank in NIET Engineering Institute",]

],

[

r"why should i choose niet?",

["NIET is one of the premier Engineering and Management institutes of Delhi NCR\nIt is a first private Autonomous institute in Uttar Pradesh !",]

],

[

r"is the transport facility available?",

["Yes! Bus is available.",]

[

r"can i apply for scholarship?",

["Yes! you can apply.",]

],

[

r"what is the average package?",

["The average package offered in NIET placement record 2021 is 4.72 LPA.",]

],

[

r"what is admission process?",

["Fill the Application form of AKTU or visit the collage with documents",]

],

[

r"how to reach niet campus?",

["you can reach NIET campus (Plot no 19, Knowledge Park II) by bus, metro from Noida/Delhi NCR",]

],

[

r"what are the courses you provide?",

["B.Tech in CSE,CSE(AI),CSE(AIML),CSE(DS),CSE(IT),BT,ME,ECE \nM.Tech in

CSE,AI,BT and other D.Pharma, B.Pharma,M.Pharma,MBA,MCA",]

],

[

r"b.tec",

["Duration-4 years, Fee-5.6 L, Placement-4.5L avg \nCourses-

CSE,CSE(AI),CSE(AIML),CSE(DS),CSE(IT),BT,ME,ECE , Affiliated to AKTU",]

],

[

r"m.tech",

["Duration-2 years, Fee-2.6 L, Placement-4.5L avg,Courses-CSE,AI,BT,ME",]

],

[

r"b.pharma",

["Duration-3 years, Fee-2.6 L, Placement-3.5L avg",]

],

[

r"m.pharma",

["Duration-2 years, Fee-2.6 L, Placement-3.5L avg",]

],

[

r"mba",

['Duration-2 years, Fee-2.8 L, Placement-4.7L avg, Affiliated to AKTU',]

],

[

r"mca",

["Duration-2 years, Fee-2.8 L, Placement-4.0L avg, Affiliated to AKTU"]

],

[

r"eligibility criteria for admission in b.tech ?",

["candidate must have passed with minimum 65 percent in 12th from any board",]

[

r"eligibility criteria for admission in m.tech?",

["candidate must have passed with minimum 50 percent in degree course",]

],

r"eligibility criteria for admission in b.pharma?", ["candidate must have passed in 12th from any board",]

],

[

r"eligibility criteria for admission in m.pharma ?",

["candidate must have passed with minimum 45 percent in degree course",]

],

[

r"eligibility criteria for admission in mba?",

["candidate must have passed with minimum 50 percent in degree course.",]

],

[

r"eligibility criteria for admission in mca?",

["candidate must have passed with minimum 45 percent in 12th from any board.",]

],

[

r"direct admission?",

["Total 15 percent of the approved intake of a course is filled by direct admission please visit collage for direct admission",]

],

[

r"what are the documents required for admission?",

["Aadhar Card,10th and 12th marksheet,photos,\nproof of Entrance Exam,Graduation degree and mark sheets (in case of PG courses and lateral entry)",]

],

[

r"how can i deposit college fee?",

["Go to the collage website or direct cash or DD",]

],

[

r"what are the timing for admission?", ["9:00 a.m. to 5 p.m. ,Monday to Saturday"]

],

[

r"is there any hostel accommodation facility?",

["Yes! We provides separate boys and girls hostel accommodation"]

],

[

r"can i visit campus?",

["Yes! visit to departments, Innovation labs, Library and Hostel and other facilities."]

],

[

r"contact details?",

["Mobile No-0000000000,gmail-niet@gmail.com"]

],

]

def chat():

print("Hi! I am a NIET chatbot created by Rishav Kumar") chat = Chat(pairs,

reflections) chat.converse() if name == " main ": chat()

Output:

*Hi! I am a NIET chatbot created by Rishav Kumar*

*>hi*

*Hello, Wellcome to NIET Greater Noida!*

*Please enter your course*

*>b.tech*

*Duration-4 years, Fee-5.6 L, Placement-4.5L avg*

*Courses-CSE,CSE(AI),CSE(AIML),CSE(DS),CSE(IT),BT,ME,ECE , Affiliated to AKTU >tell me about your college?*

*Accredited by NAAC(A Grade, 3.23) and NBA(CSE, ECE, IT, BT, ME, MBA, MCA and B.Pharm)*

*It has been ranked 43rd in NIET Pharmacy Institute and 171st rank in NIET Engineering*

*Institute*

*>why should i choose niet?*

*NIET is one of the premier Engineering and Management institutes of Delhi NCR It is a first private Autonomous institute in Uttar Pradesh !*

*>what is admission process?*

*Fill the Application form of AKTU or visit the collage with documents >direct admission?*

*Total 15 percent of the approved intake of a course is filled by direct admission please visit collage for direct admission >can i apply for scholarship?*

*Yes! you can apply.*

*>is the transport facility available?*

*Yes! Bus is available.*

*>eligibility criteria for admission in b.tech ?*

*candidate must have passed with minimum 65 percent in 12th from any board >what are the documents required for admission? Aadhar Card,10th and 12th marksheet,photos,*

*proof of Entrance Exam,Graduation degree and mark sheets (in case of PG courses and lateral entry)*

*>how to reach niet campus? you can reach NIET campus (Plot no 19, Knowledge Park II) by bus, metro from Noida/Delhi NCR >contact details?*

*Mobile No-0000000000,gmail-niet@gmail.com*

*>quit*

*Thanks :)*

Question-2

import os import time

board = [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ']

player = 1

########win Flags##########

Win = 1

Draw = -1

Running = 0

Stop = 1

###########################

Game = Running Mark = 'X' def DrawBoard(): print(" %c | %c | %c " % (board[1],board[2],board[3])) print("\_\_\_|\_\_\_|\_\_\_")

print(" %c | %c | %c " % (board[4],board[5],board[6])) print("\_\_\_|\_\_\_|\_\_\_")

print(" %c | %c | %c " % (board[7],board[8],board[9])) print(" | | ") def CheckPosition(x): if(board[x] == ' '): return True else: return False def CheckWin(): global Game

if(board[1] == board[2] and board[2] == board[3] and board[1] != ' '): Game = Win elif(board[4] == board[5] and board[5] == board[6] and board[4] != ' '): Game = Win elif(board[7] == board[8] and board[8] == board[9] and board[7] != ' '): Game = Win elif(board[1] == board[4] and board[4] == board[7] and board[1] != ' '): Game = Win elif(board[2] == board[5] and board[5] == board[8] and board[2] != ' '): Game = Win elif(board[3] == board[6] and board[6] == board[9] and board[3] != ' '): Game=Win elif(board[1] == board[5] and board[5] == board[9] and board[5] != ' '): Game = Win elif(board[3] == board[5] and board[5] == board[7] and board[5] != ' '): Game=Win

elif(board[1]!=' ' and board[2]!=' ' and board[3]!=' ' and board[4]!=' ' and board[5]!=' ' and board[6]!=' ' and board[7]!=' ' and board[8]!=' ' and board[9]!=' '): Game=Draw else:

Game=Running

print("Tic-Tac-Toe Game Designed By Sourabh Somani") print("Player 1 [X] --- Player 2 [O]\n") print() print()

print("Please Wait...") time.sleep(3) while(Game == Running): os.system('cls') DrawBoard() if(player % 2 != 0): print("Player 1's chance") Mark = 'X' else:

print("Player 2's chance") Mark = 'O'

choice = int(input("Enter the position between [1-9] where you want to mark : ")) if(CheckPosition(choice)): board[choice] = Mark

player+=1 CheckWin()

os.system('cls') DrawBoard() if(Game==Draw):

print("Game Draw") elif(Game==Win): player-=1 if(player%2!=0):

print("Player 1 Won") else:

print("Player 2 Won")

Output:

Tic-Tac-Toe Game Designed By Sourabh Somani

Player 1 [X] --- Player 2 [O]

Please Wait...

| |

\_\_\_|\_\_\_|\_\_\_

| |

\_\_\_|\_\_\_|\_\_\_

| |

| |

Player 1's chance

Question- 3

Implement alpha-beta pruning graphically with proper example and justify the pruning.

# working of Alpha-Beta Pruning

# Initial values of Alpha and Beta

MAX, MIN = 1000, -1000

# Returns optimal value for current player

#(Initially called for root and maximizer)

def minimax(depth, nodeIndex, maximizingPlayer,

values, alpha, beta):

# Terminating condition. i.e

# leaf node is reached

if depth == 3:

return values[nodeIndex]

if maximizingPlayer:

best = MIN

# Recur for left and right children

for i in range(0, 2):

val = minimax(depth + 1, nodeIndex \* 2 + i,

False, values, alpha, beta)

best = max(best, val)

alpha = max(alpha, best)

# Alpha Beta Pruning

if beta <= alpha:

break

return best

else:

best = MAX

# Recur for left and

# right children

for i in range(0, 2):

val = minimax(depth + 1, nodeIndex \* 2 + i,

True, values, alpha, beta)

best = min(best, val)

beta = min(beta, best)

# Alpha Beta Pruning

if beta <= alpha:

break

return best

if \_\_name\_\_ == "\_\_main\_\_":

values = [3, 5, 6, 9, 1, 2, 0, -1]

print("The optimal value is :", minimax(0, 0, True, values, MIN, MAX))

OUTPUT:  
The optimal value is : 5

# Question-4 : Write a python program to implement Water Jug Problem.

from collections import deque

def BFS(a, b, target):

# Map is used to store the states, every

# state is hashed to binary value to

# indicate either that state is visited

# before or not

m = {}

isSolvable = False

path = []

# Queue to maintain states

q = deque()

# Initialing with initial state

q.append((0, 0))

while (len(q) > 0):

# Current state

u = q.popleft()

# q.pop() #pop off used state

# If this state is already visited

if ((u[0], u[1]) in m):

continue

# Doesn't met jug constraints

if ((u[0] > a or u[1] > b or

u[0] < 0 or u[1] < 0)):

continue

# Filling the vector for constructing

# the solution path

path.append([u[0], u[1]])

# Marking current state as visited

m[(u[0], u[1])] = 1

# If we reach solution state, put ans=1

if (u[0] == target or u[1] == target):

isSolvable = True

if (u[0] == target):

if (u[1] != 0):

# Fill final state

path.append([u[0], 0])

else:

if (u[0] != 0):

# Fill final state

path.append([0, u[1]])

# Print the solution path

sz = len(path)

for i in range(sz):

print("(", path[i][0], ",",

path[i][1], ")")

break

# If we have not reached final state

# then, start developing intermediate

# states to reach solution state

q.append([u[0], b]) # Fill Jug2

q.append([a, u[1]]) # Fill Jug1

for ap in range(max(a, b) + 1):

# Pour amount ap from Jug2 to Jug1

c = u[0] + ap

d = u[1] - ap

# Check if this state is possible or not

if (c == a or (d == 0 and d >= 0)):

q.append([c, d])

# Pour amount ap from Jug 1 to Jug2

c = u[0] - ap

d = u[1] + ap

# Check if this state is possible or not

if ((c == 0 and c >= 0) or d == b):

q.append([c, d])

# Empty Jug2

q.append([a, 0])

# Empty Jug1

q.append([0, b])

# No, solution exists if ans=0

if (not isSolvable):

print("No solution")

# Driver code

if \_\_name\_\_ == '\_\_main\_\_':

Jug1, Jug2, target = 4, 3, 2

print("Path from initial state "

"to solution state ::")

BFS(Jug1, Jug2, target)

OUTPUT:  
Path from initial state to solution state ::

( 0 , 0 )

( 0 , 3 )

( 4 , 0 )

( 4 , 3 )

( 3 , 0 )

( 1 , 3 )

( 3 , 3 )

( 4 , 2 )

( 0 , 2 )

# Question-5: Use Heuristic Search Techniques to Implement Best first search (Best-Solution But not always optimal) and A\* algorithm

from queue import PriorityQueue

v = 14

graph = [[] for i in range(v)]

# Function For Implementing Best First Search

# Gives output path having lowest cost

def best\_first\_search(actual\_Src, target, n):

visited = [False] \* n

pq = PriorityQueue()

pq.put((0, actual\_Src))

visited[actual\_Src] = True

while pq.empty() == False:

u = pq.get()[1]

# Displaying the path having lowest cost

print(u, end=" ")

if u == target:

break

for v, c in graph[u]:

if visited[v] == False:

visited[v] = True

pq.put((c, v))

print()

# Function for adding edges to graph

def addedge(x, y, cost):

graph[x].append((y, cost))

graph[y].append((x, cost))

# The nodes shown in above example(by alphabets) are

# implemented using integers addedge(x,y,cost);

addedge(0, 1, 3)

addedge(0, 2, 6)

addedge(0, 3, 5)

addedge(1, 4, 9)

addedge(1, 5, 8)

addedge(2, 6, 12)

addedge(2, 7, 14)

addedge(3, 8, 7)

addedge(8, 9, 5)

addedge(8, 10, 6)

addedge(9, 11, 1)

addedge(9, 12, 10)

addedge(9, 13, 2)

source = 0

target = 9

best\_first\_search(source, target, v)

OUTPUT:

0 1 3 2 8 9

# Question 6: Use Heuristic Search Techniques to Implement Hill-Climbing Algorithm

import random

def randomSolution(tsp):

cities = list(range(len(tsp)))

solution = []

for i in range(len(tsp)):

randomCity = cities[random.randint(0, len(cities) - 1)]

solution.append(randomCity)

cities.remove(randomCity)

return solution

def routeLength(tsp, solution):

routeLength = 0

for i in range(len(solution)):

routeLength += tsp[solution[i - 1]][solution[i]]

return routeLength

def getNeighbours(solution):

neighbours = []

for i in range(len(solution)):

for j in range(i + 1, len(solution)):

neighbour = solution.copy()

neighbour[i] = solution[j]

neighbour[j] = solution[i]

neighbours.append(neighbour)

return neighbours

def getBestNeighbour(tsp, neighbours):

bestRouteLength = routeLength(tsp, neighbours[0])

bestNeighbour = neighbours[0]

for neighbour in neighbours:

currentRouteLength = routeLength(tsp, neighbour)

if currentRouteLength < bestRouteLength:

bestRouteLength = currentRouteLength

bestNeighbour = neighbour

return bestNeighbour, bestRouteLength

def hillClimbing(tsp):

currentSolution = randomSolution(tsp)

currentRouteLength = routeLength(tsp, currentSolution)

neighbours = getNeighbours(currentSolution)

bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

while bestNeighbourRouteLength < currentRouteLength:

currentSolution = bestNeighbour

currentRouteLength = bestNeighbourRouteLength

neighbours = getNeighbours(currentSolution)

bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

return currentSolution, currentRouteLength

def main():

tsp = [

[0, 400, 500, 300],

[400, 0, 300, 500],

[500, 300, 0, 400],

[300, 500, 400, 0]

]

print(hillClimbing(tsp))

if \_\_name\_\_ == "\_\_main\_\_":

main()

# Question 7 : Write a program to implement Hangman game using pyhton.

import random

import time

# Initial Steps to invite in the game:

print("\nWelcome to Hangman game\n")

name = input("Enter your name: ")

print("Hello " + name + "! Best of Luck!")

time.sleep(2)

print("The game is about to start!\n Let's play Hangman!")

time.sleep(3)

# The parameters we require to execute the game:

def main():

global count

global display

global word

global already\_guessed

global length

global play\_game

words\_to\_guess = ["january","border","image","film","promise","kids","lungs","doll","rhyme","damage"

,"plants"]

word = random.choice(words\_to\_guess)

length = len(word)

count = 0

display = '\_' \* length

already\_guessed = []

play\_game = ""

# A loop to re-execute the game when the first round ends:

def play\_loop():

global play\_game

play\_game = input("Do You want to play again? y = yes, n = no \n")

while play\_game not in ["y", "n","Y","N"]:

play\_game = input("Do You want to play again? y = yes, n = no \n")

if play\_game == "y":

main()

elif play\_game == "n":

print("Thanks For Playing! We expect you back again!")

exit()

# Initializing all the conditions required for the game:

def hangman():

global count

global display

global word

global already\_guessed

global play\_game

limit = 5

guess = input("This is the Hangman Word: " + display + " Enter your guess: \n")

guess = guess.strip()

if len(guess.strip()) == 0 or len(guess.strip()) >= 2 or guess <= "9":

print("Invalid Input, Try a letter\n")

hangman()

elif guess in word:

already\_guessed.extend([guess])

index = word.find(guess)

word = word[:index] + "\_" + word[index + 1:]

display = display[:index] + guess + display[index + 1:]

print(display + "\n")

elif guess in already\_guessed:

print("Try another letter.\n")

else:

count += 1

if count == 1:

time.sleep(1)

print(" \_\_\_\_\_ \n"

" | \n"

" | \n"

" | \n"

" | \n"

" | \n"

" | \n"

"\_\_|\_\_\n")

print("Wrong guess. " + str(limit - count) + " guesses remaining\n")

elif count == 2:

time.sleep(1)

print(" \_\_\_\_\_ \n"

" | | \n"

" | |\n"

" | \n"

" | \n"

" | \n"

" | \n"

"\_\_|\_\_\n")

print("Wrong guess. " + str(limit - count) + " guesses remaining\n")

elif count == 3:

time.sleep(1)

print(" \_\_\_\_\_ \n"

" | | \n"

" | |\n"

" | | \n"

" | \n"

" | \n"

" | \n"

"\_\_|\_\_\n")

print("Wrong guess. " + str(limit - count) + " guesses remaining\n")

elif count == 4:

time.sleep(1)

print(" \_\_\_\_\_ \n"

" | | \n"

" | |\n"

" | | \n"

" | O \n"

" | \n"

" | \n"

"\_\_|\_\_\n")

print("Wrong guess. " + str(limit - count) + " last guess remaining\n")

elif count == 5:

time.sleep(1)

print(" \_\_\_\_\_ \n"

" | | \n"

" | |\n"

" | | \n"

" | O \n"

" | /|\ \n"

" | / \ \n"

"\_\_|\_\_\n")

print("Wrong guess. You are hanged!!!\n")

print("The word was:",already\_guessed,word)

play\_loop()

if word == '\_' \* length:

print("Congrats! You have guessed the word correctly!")

play\_loop()

elif count != limit:

hangman()

main()

hangman()

OUTPUT:

Welcome to Hangman game by IT SOURCECODE

Enter your name: akriti

Hello akriti! Best of Luck!

The game is about to start!

Let's play Hangman!

This is the Hangman Word: \_\_\_\_\_\_\_ Enter your guess:

doll

Invalid Input, Try a letter

This is the Hangman Word: \_\_\_\_\_\_\_ Enter your guess:

y

\_\_\_\_\_\_y

This is the Hangman Word: \_\_\_\_\_\_y Enter your guess:

Y

\_\_\_\_\_

|

|

|

|

|

|

\_\_|\_\_

Wrong guess. 4 guesses remaining

This is the Hangman Word: \_\_\_\_\_\_y Enter your guess:

n

\_\_n\_\_\_y

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

Y

\_\_\_\_\_

| |

| |

|

|

|

|

\_\_|\_\_

Wrong guess. 3 guesses remaining

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

N

\_\_\_\_\_

| |

| |

| |

|

|

|

\_\_|\_\_

Wrong guess. 2 guesses remaining

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

january

Invalid Input, Try a letter

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

yu

Invalid Input, Try a letter

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

ghgh

Invalid Input, Try a letter

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

hello

Invalid Input, Try a letter

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

kids

Invalid Input, Try a letter

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

h

\_\_\_\_\_

| |

| |

| |

| O

|

|

\_\_|\_\_

Wrong guess. 1 last guess remaining

This is the Hangman Word: \_\_n\_\_\_y Enter your guess:

g

\_\_\_\_\_

| |

| |

| |

| O

| /|\

| / \

\_\_|\_\_

Wrong guess. You are hanged!!!

The word was: ['y', 'n'] ja\_uar\_

Do You want to play again? y = yes, n = no

n

Thanks For Playing! We expect you back again!

# Question 8: Write program to solve monkey banana problem.

move(state(middle,onbox,middle,hasnot),

grasp,

state(middle,onbox,middle,has)).

move(state(P,onfloor,P,H),

climb,

state(P,onbox,P,H)).

move(state(P1,onfloor,P1,H),

drag(P1,P2),

state(P2,onfloor,P2,H)).

move(state(P1,onfloor,B,H),

walk(P1,P2),

state(P2,onfloor,B,H)).

canget(state(\_,\_,\_,has)).

canget(State1) :-

move(State1,\_,State2),

canget(State2).

Output:

| ?- [monkey\_banana].

compiling D:/TP Prolog/Sample\_Codes/monkey\_banana.pl for byte code...

D:/TP Prolog/Sample\_Codes/monkey\_banana.pl compiled, 17 lines read - 2167 bytes written, 19 ms

(31 ms) yes

| ?- canget(state(atdoor, onfloor, atwindow, hasnot)).

true ?

yes

| ?- trace

.

The debugger will first creep -- showing everything (trace)

yes

{trace}

| ?- canget(state(atdoor, onfloor, atwindow, hasnot)).

1 1 Call: canget(state(atdoor,onfloor,atwindow,hasnot)) ?

2 2 Call: move(state(atdoor,onfloor,atwindow,hasnot),\_52,\_92) ?

2 2 Exit:move(state(atdoor,onfloor,atwindow,hasnot),walk(atdoor,\_80),state(\_80,onfloor,atwindow,hasnot)) ?

3 2 Call: canget(state(\_80,onfloor,atwindow,hasnot)) ?

4 3 Call: move(state(\_80,onfloor,atwindow,hasnot),\_110,\_150) ?

4 3 Exit: move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,onbox,atwindow,hasnot)) ?

5 3 Call: canget(state(atwindow,onbox,atwindow,hasnot)) ?

6 4 Call: move(state(atwindow,onbox,atwindow,hasnot),\_165,\_205) ?

6 4 Fail: move(state(atwindow,onbox,atwindow,hasnot),\_165,\_193) ?

5 3 Fail: canget(state(atwindow,onbox,atwindow,hasnot)) ?

4 3 Redo: move(state(atwindow,onfloor,atwindow,hasnot),climb,state(atwindow,onbox,atwindow,hasnot)) ?

4 3 Exit: move(state(atwindow,onfloor,atwindow,hasnot),drag(atwindow,\_138),state(\_138,onfloor,\_138,hasnot)) ?

5 3 Call: canget(state(\_138,onfloor,\_138,hasnot)) ?

6 4 Call: move(state(\_138,onfloor,\_138,hasnot),\_168,\_208) ?

6 4 Exit: move(state(\_138,onfloor,\_138,hasnot),climb,state(\_138,onbox,\_138,hasnot)) ?

7 4 Call: canget(state(\_138,onbox,\_138,hasnot)) ?

8 5 Call: move(state(\_138,onbox,\_138,hasnot),\_223,\_263) ?

8 5 Exit: move(state(middle,onbox,middle,hasnot),grasp,state(middle,onbox,middle,has)) ?

9 5 Call: canget(state(middle,onbox,middle,has)) ?

9 5 Exit: canget(state(middle,onbox,middle,has)) ?

7 4 Exit: canget(state(middle,onbox,middle,hasnot)) ?

5 3 Exit: canget(state(middle,onfloor,middle,hasnot)) ?

3 2 Exit: canget(state(atwindow,onfloor,atwindow,hasnot)) ?

1 1 Exit: canget(state(atdoor,onfloor,atwindow,hasnot)) ?

true ?

yes

# Question 9 : Write a pyhton program to implement Simple Calculator program.

# Program make a simple calculator

# This function adds two numbers

def add(x, y):

return x + y

# This function subtracts two numbers

def subtract(x, y):

return x - y

# This function multiplies two numbers

def multiply(x, y):

return x \* y

# This function divides two numbers

def divide(x, y):

return x / y

print("Select operation.")

print("1.Add")

print("2.Subtract")

print("3.Multiply")

print("4.Divide")

while True:

# take input from the user

choice = input("Enter choice(1/2/3/4): ")

# check if choice is one of the four options

if choice in ('1', '2', '3', '4'):

num1 = float(input("Enter first number: "))

num2 = float(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))

# check if user wants another calculation

# break the while loop if answer is no

next\_calculation = input("Let's do next calculation? (yes/no): ")

if next\_calculation == "no":

break

else:

print("Invalid Input")

OUTPUT:

Select operation.

1.Add

2.Subtract

3.Multiply

4.Divide

Enter choice(1/2/3/4): 4

Enter first number: 34

Enter second number: 4

34.0 / 4.0 = 8.5

Let's do next calculation? (yes/no): yes

Enter choice(1/2/3/4): 4

Enter first number: 8

Enter second number: 4

8.0 / 4.0 = 2.0

Let's do next calculation? (yes/no): yes

Enter choice(1/2/3/4): 4657839283829

Invalid Input

Enter choice(1/2/3/4): 6677777

Invalid Input

Enter choice(1/2/3/4): 7777

Invalid Input

Enter choice(1/2/3/4): 4

Enter first number: 77777777

Enter second number: 7777

77777777.0 / 7777.0 = 10001.0

# Question 10 : Write a python program to POS(Part of speech) tagging for the given sentence using NLTK.

Import nltk

From nltk.corpus import stopwords

From nltk.tokenize import word\_tokenize, sent\_tokenize

Stop\_words = set(stopwords.words(‘english’))

// Dummy text

Txt = “Sukanya, Rajib and Naba are my good friends. “ “Sukanya is getting married next year. “ “Marriage is a big step in ones life.” “It is both exciting and frightening. “ “But friendship is a sacred bond between people.” “It is a special kind of love between us. “ “Many of you must have tried searching for a friend “ “but never found the right one.”

# sent\_tokenize is one of instances of

# PunktSentenceTokenizer from the nltk.tokenize.punkt module

Tokenized = sent\_tokenize(txt)

For I in tokenized:

# Word tokenizers is used to find the words

# and punctuation in a string

wordsList = nltk.word\_tokenize(i)

# removing stop words from wordlist

wordsList = [w for w in wordsList if not w in stop\_words]

# Using a Tagger. Which is part-of-speech

# tagger or POS-tagger.

Tagged = nltk.pos\_tag(wordsList)

Print(tagged)

**OUTPUT:**

[(‘Sukanya’, ‘NNP’), (‘Rajib’, ‘NNP’), (‘Naba’, ‘NNP’), (‘good’, ‘JJ’), (‘friends’, ‘NNS’)]

[(‘Sukanya’, ‘NNP’), (‘getting’, ‘VBG’), (‘married’, ‘VBN’), (‘next’, ‘JJ’), (‘year’, ‘NN’)]

[(‘Marriage’, ‘NN’), (‘big’, ‘JJ’), (‘step’, ‘NN’), (‘one’, ‘CD’), (‘’, ‘NN’), (‘life’, ‘NN’)]

[(‘It’, ‘PRP’), (‘exciting’, ‘VBG’), (‘frightening’, ‘VBG’)]

[(‘But’, ‘CC’), (‘friendship’, ‘NN’), (‘sacred’, ‘VBD’), (‘bond’, ‘NN’), (‘people’, ‘NNS’)]

[(‘It’, ‘PRP’), (‘special’, ‘JJ’), (‘kind’, ‘NN’), (‘love’, ‘VB’), (‘us’, ‘PRP’)]

[(‘Many’, ‘JJ’), (‘must’, ‘MD’), (‘tried’, ‘VB’), (‘searching’, ‘VBG’), (‘friend’, ‘NN’),

(‘never’, ‘RB’), (‘found’, ‘VBD’), (‘right’, ‘RB’), (‘one’, ‘CD’)]

# Question 11 : Solve 8-Puzzle problem using best first search

#SOLVE 8 PUZZLE PROBLEM USING BEST FIRST SEARCH

import sys

import numpy as np

class Node:

def \_\_init\_\_(self, state, parent, action):

self.state = state

self.parent = parent

self.action = action

class StackFrontier:

def \_\_init\_\_(self):

self.frontier = []

def add(self, node):

self.frontier.append(node)

def contains\_state(self, state):

return any((node.state[0] == state[0]).all() for node in

self.frontier)

def empty(self):

return len(self.frontier) == 0

def remove(self):

if self.empty():

raise Exception("Empty Frontier")

else:

node = self.frontier[-1]

self.frontier = self.frontier[:-1]

return node

class QueueFrontier(StackFrontier):

def remove(self):

if self.empty():

raise Exception("Empty Frontier")

else:

node = self.frontier[0]

self.frontier = self.frontier[1:]

return node

class Puzzle:

def \_\_init\_\_(self, start, startIndex, goal, goalIndex):

self.start = [start, startIndex]

self.goal = [goal, goalIndex]

self.solution = None

def neighbors(self, state):

mat, (row, col) = state

results = []

if row > 0:

mat1 = np.copy(mat)

mat1[row][col] = mat1[row - 1][col]

mat1[row - 1][col] = 0

results.append(('up', [mat1, (row - 1, col)]))

if col > 0:

mat1 = np.copy(mat)

mat1[row][col] = mat1[row][col - 1]

mat1[row][col - 1] = 0

results.append(('left', [mat1, (row, col - 1)]))

if row < 2:

mat1 = np.copy(mat)

mat1[row][col] = mat1[row + 1][col]

mat1[row + 1][col] = 0

results.append(('down', [mat1, (row + 1, col)]))

if col < 2:

mat1 = np.copy(mat)

mat1[row][col] = mat1[row][col + 1]

mat1[row][col + 1] = 0

results.append(('right', [mat1, (row, col + 1)]))

return results

def print(self):

solution = self.solution if self.solution is not None else None

print("Start State:\n", self.start[0], "\n")

print("Goal State:\n", self.goal[0], "\n")

print("\nStates Explored: ", self.num\_explored, "\n")

print("Solution:\n ")

for action, cell in zip(solution[0], solution[1]):

print("action: ", action, "\n", cell[0], "\n")

print("Goal Reached!!")

def does\_not\_contain\_state(self, state):

for st in self.explored:

if (st[0] == state[0]).all():

return False

return True

def solve(self):

self.num\_explored = 0

start = Node(state=self.start, parent=None, action=None)

frontier = QueueFrontier()

frontier.add(start)

self.explored = []

while True:

if frontier.empty():

raise Exception("No solution")

node = frontier.remove()

self.num\_explored += 1

if (node.state[0] == self.goal[0]).all():

actions = []

cells = []

while node.parent is not None:

actions.append(node.action)

cells.append(node.state)

node = node.parent

actions.reverse()

cells.reverse()

self.solution = (actions, cells)

return

self.explored.append(node.state)

for action, state in self.neighbors(node.state):

if not frontier.contains\_state(state) and self.does\_not\_contain\_state(state):

child = Node(state=state, parent=node, action=action)

frontier.add(child)

start = np.array([[1, 2, 3], [8, 0, 4], [7, 6, 5]])

goal = np.array([[2, 8, 1], [0, 4, 3], [7, 6, 5]])

startIndex = (1, 1)

goalIndex = (1, 0)

p = Puzzle(start, startIndex, goal, goalIndex)

p.solve()

p.print()

OUTPUT:

Start State:

[[1 2 3]

[8 0 4]

[7 6 5]]

Goal State:

[[2 8 1]

[0 4 3]

[7 6 5]]

States Explored: 358

Solution:

action: up

[[1 0 3]

[8 2 4]

[7 6 5]]

action: left

[[0 1 3]

[8 2 4]

[7 6 5]]

action: down

[[8 1 3]

[0 2 4]

[7 6 5]]

action: right

[[8 1 3]

[2 0 4]

[7 6 5]]

action: right

[[8 1 3]

[2 4 0]

[7 6 5]]

action: up

[[8 1 0]

[2 4 3]

[7 6 5]]

action: left

[[8 0 1]

[2 4 3]

[7 6 5]]

action: left

[[0 8 1]

[2 4 3]

[7 6 5]]

action: down

[[2 8 1]

[0 4 3]

[7 6 5]]

Goal Reached!!

# Question 12 : Solve robot (traversal) problem using mean end analysis.

# Python3 program to implement traveling salesman

# problem using naive approach.

from sys import maxsize

from itertools import permutations

V = 4

# implementation of traveling Salesman Problem

def travellingSalesmanProblem(graph, s):

# store all vertex apart from source vertex

vertex = []

for i in range(V):

if i != s:

vertex.append(i)

# store minimum weight Hamiltonian Cycle

min\_path = maxsize

next\_permutation=permutations(vertex)

for i in next\_permutation:

# store current Path weight(cost)

current\_pathweight = 0

# compute current path weight

k = s

for j in i:

current\_pathweight += graph[k][j]

k = j

current\_pathweight += graph[k][s]

# update minimum

min\_path = min(min\_path, current\_pathweight)

return min\_path

# Driver Code

if \_\_name\_\_ == "\_\_main\_\_":

# matrix representation of graph

graph = [[0, 10, 15, 20], [10, 0, 35, 25],

[15, 35, 0, 30], [20, 25, 30, 0]]

s = 0

print(travellingSalesmanProblem(graph, s))

OUTPUT:

80

80

# Question 13 : Implementation of image features Processing using OPENCV and OPENVINO

# Importing the libraries

import cv2

import numpy as np

# Reading the image and converting into B?W

image = cv2.imread("book.png")

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Applying the function

corners = cv2.goodFeaturesToTrack(

gray\_image, maxCorners=50, qualityLevel=0.02, minDistance=20)

corners = np.float32(corners)

for item in corners:

x, y = item[0]

x = int(x)

y = int(y)

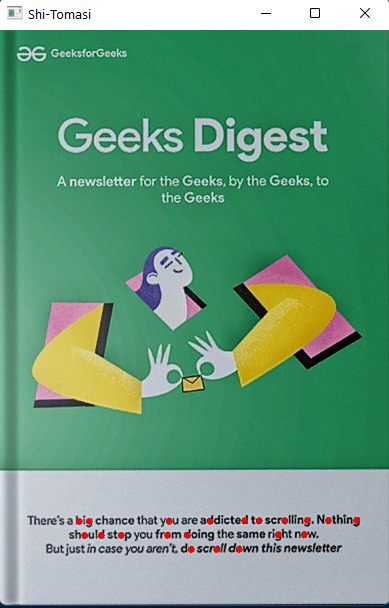
cv2.circle(image, (x, y), 6, (0, 255, 0), -1)

# Showing the image

cv2.imshow('good\_features', image)

cv2.waitKey()

**OUTPUT**



# Question 14 : Write a program to implement Naïve Bayes Algorithm

# load the iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# store the feature matrix (X) and response vector (y)

X = iris.data

y = iris.target

# splitting X and y into training and testing sets

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=1)

# training the model on training set

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

# making predictions on the testing set

y\_pred = gnb.predict(X\_test)

# comparing actual response values (y\_test) with predicted response values (y\_pred)

from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy\_score(y\_test, y\_pred)\*100)

OUTPUT:

Gaussian Naive Bayes model accuracy(in %):95.0

# Question 15 : Write a Program to implememnt alpha-bera Purning.

class MinimaxABAgent:

""Minimax agent ""

def \_\_init\_\_(self, max\_depth, player\_color):

"""

Initiation

Parameters

----------

max\_depth : int

The max depth of the tree

player\_color : int

The player's index as MAX in minimax algorithm

self.max\_depth = max\_depth

self.player\_color = player\_color

self.node\_expanded = 0

def choose\_action(self, state):

"""

Predict the move using minimax algorithm

Parameters

----------

state : State

Returns

-------

float, str:

The evaluation or utility and the action key name

"""

self.node\_expanded = 0

start\_time = time.time()

print("MINIMAX AB : Wait AI is choosing")

list\_action = AIElements.get\_possible\_action(state)

eval\_score, selected\_key\_action = self.\_minimax(0,state,True,float('-inf'),float('inf'))

print("MINIMAX : Done, eval = %d, expanded %d" % (eval\_score, self.node\_expanded))

print("--- %s seconds ---" % (time.time() - start\_time))

return (selected\_key\_action,list\_action[selected\_key\_action])

def \_minimax(self, current\_depth, state, is\_max\_turn, alpha, beta):

if current\_depth == self.max\_depth or state.is\_terminal():

return AIElements.evaluation\_function(state, self.player\_color), ""

self.node\_expanded += 1

possible\_action = AIElements.get\_possible\_action(state)

key\_of\_actions = list(possible\_action.keys())

shuffle(key\_of\_actions) #randomness

best\_value = float('-inf') if is\_max\_turn else float('inf')

action\_target = ""

for action\_key in key\_of\_actions:

new\_state = AIElements.result\_function(state,possible\_action[action\_key])

eval\_child, action\_child = self.\_minimax(current\_depth+1,new\_state,not is\_max\_turn, alpha, beta)

if is\_max\_turn and best\_value < eval\_child:

best\_value = eval\_child

action\_target = action\_key

alpha = max(alpha, best\_value)

if beta <= alpha:

break

elif (not is\_max\_turn) and best\_value > eval\_child:

best\_value = eval\_child

action\_target = action\_key

beta = min(beta, best\_value)

if beta <= alpha:

break

return best\_value, action\_target