Name: Shaikh Mohd Raza

Roll No: 20CO53

**Experiment No : 5**

**AIM:**

To implement WUMPUS world.

**THEORY :**

The Wumpus world is a simple world example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game Hunt the Wumpus by Gregory Yob in 1973.

**Problem statement :**

The WUMPUS world is a cave which has 4/4 rooms connected with passageways. So there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus, who eats anyone who enters the room. The Wumpus can be shot by the agent, but the agent has a single arrow. In the Wumpus world, there are some Pits rooms which are bottomless, and if agent falls in Pits, then he will be stuck there forever. The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold. So the agent goal is to find the gold and climb out the cave without fallen into Pits or eaten by Wumpus. The agent will get a reward if he comes out with gold, and he will get a penalty if eaten by Wumpus or falls in the pit.

Following is a sample diagram for representing the Wumpus world. It is showing some rooms with Pits, one room with Wumpus and one agent at (1, 1) square location of the world.

PEAS represents Performance Measures, Environment, Actuators, and Sensors. The PEAS description helps in grouping the agents.

**PEAS Description for the Wumpus World problem:**

* Performance measures:
* Agent gets the gold and return back safe = +1000 points
* Agent dies = -1000 points
* Each move of the agent = -1 point
* Agent uses the arrow = -10 points

**CODE :-**

def learnagent(world, i, j):

'''Function for an agent to know what poisitin contains which environment objects'''

if (world[i][j] == 9):

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

print("You came across a stench")

return agi, agj

elif (world[i][j] == 8):

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

print("You came across a glitter")

return agi, agj

elif (world[i][j] == 7):

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

print("You came across a pit")

return -5, -5

elif (world[i][j] == 6):

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

print("You found gold")

return -4, -4

elif (world[i][j] == 5):

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

print("You feel breeze")

return agi, agj

elif (world[i][j] == -1):

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

print("You met wumpus")

return -5, -5

else: # if world environment was empty

agi, agj = i, j

print("\nNow the agent is at "+str(agi)+","+str(agj))

return agi, agj

def checkinp(agi, agj):

'''Function for checking input going in forward direction to get gold'''

if(agi == 0 and agj == 0):

print("\nyou can go at "+str(agi+1)+" "+str(agj)) # can move upward

print("you can go at "+str(agi)+" "+str(agj+1)) # can move right

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi+1 and agvj == agj or agvi == agi and agvj == agj+1):

return agvi, agvj

else:

return -5

elif(agi == 3 and agj == 0):

print("\nyou can go at "+str(agi-1)+" "+str(agj)) # can go left

print("you can go at "+str(agi)+" "+str(agj+1)) # can go right

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi-1 and agvj == agj or agvi == agi and agvj == agj+1):

return agvi, agvj

else:

return -5

elif(agi == 3 and agj == 3):

print("\nyou can go at "+str(agi-1)+" "+str(agj)) # can go down

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi-1 and agvj == agj or agvi == agi and agvj == agj-1):

return agvi, agvj

else:

return -5

elif(agi == 0 and agj == 3):

print("\nyou can go at "+str(agi+1)+" "+str(agj)) # can go upward

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi+1 and agvj == agj or agvi == agi and agvj == agj-1):

return agvi, agvj

else:

return -5, -5

elif(agi == 1 and agj == 0 or agi == 2 and agj == 0 or agi == 3 and agj == 0):

print("\nyou can go at "+str(agi+1)+" "+str(agj)) # can go upward

print("you can go at "+str(agi)+" "+str(agj+1)) # can move right

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi+1 and agvj == agj or agvi == agi and agvj == agj+1):

return agvi, agvj

else:

return -5, -5

elif(agi == 0 and agj == 3 or agi == 1 and agj == 3 or agi == 2 and agj == 3 or agi == 3 and agj == 3):

print("you can go at "+str(agi+1)+" "+str(agj)) # can go upward

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

agvi = int(input("Enter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi+1 and agvj == agj or agvi == agi and agvj == agj-1):

return agvi, agvj

else:

return -5, -5

elif(agi == 3 and agj == 1 or agi == 3 and agj == 2 or agi == 3 and agj == 3):

print("\nyou can go at "+str(agi)+" "+str(agj+1)) # can go right

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

print("you can go at "+str(agi-1)+" "+str(agj)) # can move downward

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi and agvj == agj+1 or agvi == agi and agvj == agj-1 or agvi == agi-1 and agvj == agj):

return agvi, agvj

else:

return -5, -5

else:

print("\nyou can go at "+str(agi)+" "+str(agj+1)) # can go right

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

print("you can go at "+str(agi+1)+" "+str(agj)) # can move upward

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi and agvj == agj+1 or agvi == agi and agvj == agj-1 or agvi == agi+1 and agvj == agj):

return agvi, agvj

else:

return -5, -5

def checkinpreverse(agi, agj):

'''Function for checking input going in reverse direction to get back to original position'''

if(agi == 0 and agj == 3):

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi and agvj == agj-1):

return agvi, agvj

else:

return -5, -5

elif(agi == 0 and agj == 2 or agi == 0 and agj == 1):

print("you can go at "+str(agi)+" "+str(agj+1)) # can go right

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi and agvj == agj-1 or agvi == agi and agvj == agj+1):

return agvi, agvj

else:

return -5, -5

elif(agi == 1 and agj == 0 or agi == 2 and agj == 0):

print("\nyou can go at "+str(agi-1)+" "+str(agj)) # can go downward

print("you can go at "+str(agi)+" "+str(agj+1)) # can move right

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi-1 and agvj == agj or agvi == agi and agvj == agj+1):

return agvi, agvj

else:

return -5, -5

elif(agi == 1 and agj == 3 or agi == 2 and agj == 3):

print("you can go at "+str(agi-1)+" "+str(agj)) # can go downward

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

agvi = int(input("Enter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi-1 and agvj == agj or agvi == agi and agvj == agj-1):

return agvi, agvj

else:

return -5, -5

else:

print("\nyou can go at "+str(agi-1)+" "+str(agj)) # can go downward

print("you can go at "+str(agi)+" "+str(agj-1)) # can go left

print("you can go at "+str(agi)+" "+str(agj+1)) # can go right

agvi = int(input("\nEnter input for row => "))

agvj = int(input("Enter input for column => "))

if(agvi == agi-1 and agvj == agj or agvi == agi and agvj == agj-1 or agvi == agi and agvj == agj+1):

return agvi, agvj

else:

return -5, -5

world = [ [0, 5, 7, 5],

[9, 0, 8, 0],

[-1, 6, 7, 8],

[9, 0, 8, 7] ] # declaration of a world

agi, agj = 0, 0 # initial agent position

print("\n\n\ninitially agent is at "+str(agi)+","+str(agj))

print("\nyou can go at "+str(agi+1)+" "+str(agj))

print("you can go at "+str(agi)+" "+str(agj+1))

agvi = int(input("Enter input for row => "))

agvj = int(input("Enter input for column => ")) # taking row and column values

if(agvi == 1 and agvj == 0 or agvi == 0 and agvj == 1):

# if input valid calling learn agent function

agi, agj = learnagent(world, agvi, agvj)

else:

print("Not valid")

while(agi >= 0):

agvi, agvj = checkinp(agi, agj)

if(agvi != -5 and agvj != -5):

agi, agj = learnagent(world, agvi, agvj)

else:

print("\nNot valid")

if(agi == -5):

print("\nGame over Sorry try next time!!!")

else:

# acquired gold

print("\nYou have unlocked next level move back to your initial position")

agi, agj = 2, 1 # implementation of reverse logic

while(agi >= 0):

agvi, agvj = checkinpreverse(agi, agj)

if(agvi == 0 and agvj == 0):

agi, agj = -4, -4

elif(agvi != -5 and agvj != -5):

agi, agj = learnagent(world, agvi, agvj)

else:

print("\nNot valid")

if(agi == -5):

print("\nYou were really close but unfortunately you failed!!! Try next time")

else:

print("\nHurray You won!!!!! Three cheers.")

**CONCLUSION:**

In this experiment we learned about the WUMPUS world, its characteristics, peas description and also learned about the exploring how agent will traverse and find the result. And implemented it using python.