Shivam Tiwari
5117060
BE COMPS.

Experiment 3.

- Aim:- Write a program to Implement Wumpus World Problem

- Theory:-

The Wumpus world is a cave consisting of rooms connected by passage ways, lurking somewhere in the caves consist a terrible wumpus, a beast that eats anyone who enters the room. The wumpus can be shot by an agent, but the agent has only one arrow. Some room contains bottomless pits. that will trap anyone who wander into these rooms, The only mitigating feature of this bleak environment is the possibility of finding a heap of gold. Although the wumpus world is rather tame by modern computer game standards.

- PEAS description:-

- Performance measure: +1000 for climbing out of the cave with gold -1000 for falling in pit or being eaten by wumpus. -1 for each action taken -10 for using arrows. The game ends either when the agent dies or when agent climbs out of cave.

- Environment:- A 4X4 grid of rooms. The agent always starts in the square labelled [1,1] facing to the right the locations of the gold & the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start squares. In addition, each square other than the start can be a pit with probability 0.2.

stench		Breeze	Pit
	Breeze gold stench	Pit	Breeze
stench		Breeze	
start	Breeze	Pit	Breeze

- Actuators :- The agent can move forward, turn left (90°), right (90°)
If dies if it enters the wumpus room or by entering the square with pits
If the agent tries to move forward onto a wall it bumps & does not move

Action Grab → Pick gold if same square as agent

Action Shoot → The arrow straight in line where the agent is facing
Agent has only 1 arrow

Action Climb → Used to climb out of the cave.

- Sensors :-
Agent has 5 sensors
- If agent is in the adjacent square of wumpus it will percieve stench
- If agent is in the adjacent square of pits it will percieve breeze
- If agent is in the square of gold, the agent will percieve glitter
- If agent walks into a wall, it will percieve bump.
- When wumpus dies it emits a scream that can be percieved throughout cave.

Percept will be given as 5 symbols

eg:- Consider only stench an breeze ; the agent program will get
[Stench, Breeze, None, None, None]

- Conclusion:-
Thus we have understood & successfully implemented wumpus world problem while meeting all criterias.

Roll No: 5117060

WUMPUS WORLD

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):
       agi,agj=learnagent(world,agvi,agvj)            #if input valid calling learn
agent function
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:
       print("\nYou have unlocked next level move back to your initial position")
       #acquired gold


       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.")
```

Output:

```
== RESTART: C:/Users/Shivam/AppData/Local/Programs/Python/Python38-32/Wumpus.py =
initially agent is at 0,0

you can go at       1      0
you can go at       0      1
Enter input for row => 1
Enter input for column => 0

Now the agent is at 1,0
You came across a stench

you can go at       2      0
you can go at       1      1

Enter input for row => 0
Enter input for column => 0

Not valid

you can go at       2      0
you can go at       1      1

Enter input for row => 1
Enter input for column => 1

Now the agent is at 1,1

you can go at       1      2
you can go at       1      0
you can go at       2      1

Enter input for row => 1
```

```
Enter input for column => 2

Now the agent is at 1,2
You came across a glitter

you can go at       1      3
you can go at       1      1
you can go at       2      2

Enter input for row => 1
Enter input for column => 1

Now the agent is at 1,1

you can go at       1      2
you can go at       1      0
you can go at       2      1

Enter input for row => 2
Enter input for column => 1

Now the agent is at 2,1
You found gold

You have unlocked next level move back to your initial position

you can go at       1      1
you can go at       2      0
you can go at       2      2

Enter input for row => 1
Enter input for column => 1

Now the agent is at 1,1

you can go at       0      1
you can go at       1      0
you can go at       1      2
```

```
Enter input for row => 0
Enter input for column => 1

Now the agent is at 0,1
You feel breeze
you can go at       0      2
you can go at       0      0

Enter input for row => 0
Enter input for column => 0

Hurray You won!!!!! Three cheers.
>>>
```