1. **Demonstrate Bell-man equation functionality in Reinforcement Learning using Python Programming through 3 X 3 grid**

**Solution:**

#author@Dr.M.Prakash

#Reinforcement Learning

import numpy as np

# Define the grid world

grid\_world = np.zeros((3, 3))

# Define the state transition function (up, down, left, right)

actions = [(0, -1), (0, 1), (-1, 0), (1, 0)]

# Define the reward for each state

rewards = {

(0, 2): 10, # Goal state

(1, 2): -10, # Penalty state

}

# Define the discount factor

gamma = 0.9

# Perform the Bellman update for state values

num\_iterations = 100

for \_ in range(num\_iterations):

new\_grid\_world = np.copy(grid\_world)

for i in range(3):

for j in range(3):

if (i, j) not in rewards:

new\_value = 0

for action in actions:

next\_i, next\_j = i + action[0], j + action[1]

if 0 <= next\_i < 3 and 0 <= next\_j < 3:

new\_value += (grid\_world[next\_i, next\_j] + rewards.get((next\_i, next\_j), 0)) \* gamma / 4

new\_grid\_world[i, j] = new\_value

grid\_world = new\_grid\_world

# Print the final state values

print("State Values:")

print(grid\_world)

**Example Output:**

State Values:

[[ 0.29260626 1.81735407 0. ]

[-0.51688182 -2.21547706 0. ]

[-0.37438173 -1.14703698 -2.50808332]]