```
import numpy as np

# Define the warehouse grid (4x4 example)
# S = start, G = goal, O = obstacle, I = item
grid = [
    ['S', '.', '.', 'I'],
    ['.', 'O', '.', '.'],
    ['.', '.', 'O', '.'],
    ['.', '.', '.', 'G']
]

# Parameters
gamma = 0.9        # Discount factor
theta = 1e-4       # Convergence threshold
actions = ['up', 'down', 'left', 'right']

rows, cols = len(grid), len(grid[0])
V = np.zeros((rows, cols))  # Initialize value function

# Policy: move right if possible, else move down
def policy(state):
    r, c = state
    if c < cols - 1:
        return 'right'
    return 'down'

# Transition model
def step(state, action):
    r, c = state
    if grid[r][c] == 'O':  # obstacle state
        return state, -2
    if grid[r][c] == 'I':  # item state
        reward = 2
    elif grid[r][c] == 'G':  # goal state
        reward = 5
    else:
        reward = 0

    # Movement
    if action == 'up':    r = max(r - 1, 0)
    elif action == 'down': r = min(r + 1, rows - 1)
    elif action == 'left': c = max(c - 1, 0)
    elif action == 'right': c = min(c + 1, cols - 1)

    # Hitting obstacle
    if grid[r][c] == 'O':
        return (r, c), -2

    return (r, c), reward

# Policy Evaluation
def policy_evaluation():
    while True:
        delta = 0
        for r in range(rows):
            for c in range(cols):
                state = (r, c)
                v = V[r][c]
                action = policy(state)
                next_state, reward = step(state, action)
                nr, nc = next_state
                V[r][c] = reward + gamma * V[nr][nc]
                delta = max(delta, abs(v - V[r][c]))
        if delta < theta:
            break
    return V

# Run evaluation
value_function = policy_evaluation()
print("Value Function:")
print(np.round(value_function, 2))
```

```
Value Function:
[[ 28.03  31.14  34.6   38.45]
 [-20.   -20.    36.45  40.5 ]
 [-18.   -20.   -20.    45.  ]
 [ 36.45  40.5   45.    50.  ]]
```