Wumpus World

Pranjal Mathur	Prerna.	Saketh Vallakatla
1410110296	1410110306	1410110352

Introduction:

The environment consists of a *n*n* square grid consisting of an agent, a crate of gold, a Wumpus and *k* number of pits. The aim of the quest is for the agent to pick up the gold and come back to his starting position. Entering the same block as occupied by the Wumpus or the pit kills the agent. Each action the agent takes has cost (listed in table 1) and the goal is to maximize the score by the end of the quest.

The agent also has two tools to assist him in his gold hunt, both of which can be used only once:

- An arrow which can kill the Wumpus with the probability of 0.8, i.e. there are chances that the Wumpus can withstand the arrow attack and will remain alive with the probability of 0.2.
- A drone to assist him in the quest and respond back with a signal. The signal depends upon the occupant of the block. The drone can be used as a look ahead into a block.

There is an additional incentive for taking a different path to come back to the start point from the path that was taken to reach the gold. The number of points allotted for the returning path is inversely proportional to number of deviating blocks from the original path take.

Wumpus falls asleep after every two steps of the agent. That is, the agent can enter the block of Wumpus in his third, sixth, etc. steps without the threat of being killed.

Environment:

Some implicit details are:

- Agent starts from block [1,1].
- The gold crate is present in only one block.
- The Wumpus is stationary and doesn't shift blocks.
- The block can either contain gold, a pit or a Wumpus or nothing. More than one entity can't exist in a block.

Action	Points
Picking up the gold and coming back	+1000
Dying due to Monster or Pit	-1000
Each movement of the agent	-1
Using arrow	-10
Using the drone	-10
Coming back to the starting point from a different path	+2* inv(#non-coinciding blocks)

Table 1

Sensors:

The agent can sense the following:

Source	Indicator
Wumpus	Stench (S)
Pit	Breeze (B)
Wumpus	Shout
Gold	Glitter
Drone	S (if Wumpus), BT (if Pit)

- **Assumptions**
  - The indicators around the source are in all 4 possible directions. (top-right-bottom-left), but not diagonally.
  - There can be an overlap of various indicators.
  - Gold can be detected by its glitter only at the block of its presence.

Actuators:

Turn	Within a block, change direction by 90 degrees
Walk	Walk across the block boundaries
Grab	Get the gold
Shoot	Shoot arrow to kill Wumpus in a particular direction
Throw	Send drone to the immediate neighboring block in a particular direction

Code:

```
class _NOTATIONS:
    EMPTY = 0
    HERO = 1
    GOLD = 2
    WUMPUS = 3
    PIT = 4
    GLITTER = 5
    STENCH = 6
    BREEZE = 7


def makeBoard(N):
    board = []
    for i in range(0, N):
            board.append([])
    for item in board:
        for i in range(0, N):
            item.append(0)
    return board
```

```
def genPits(self, NPITS):
    emptyCells = 0
    for i in range(self._EDGE):
        emptyCells += self.board[i].count(0)
    if NPITS > emptyCells:
        for i in range(self._EDGE):
            for j in range(self._EDGE):
                if self.board[i][j] == 0:
                    self.board[i][j] = _NOTATIONS.PIT
                    self.genBreeze(i, j)
        # return True
    else:
        for i in range(NPITS):
            x = randint(0, self._EDGE - 1)
            y = randint(0, self._EDGE - 1)
            while self.board[x][y] != 0:
                x = randint(0, self._EDGE)
                y = randint(0, self._EDGE)
            self.board[x][y] = _NOTATIONS.PIT
            self.genBreeze(x, y)
    # return True

def genBreeze(self, x, y):
class Arena(object):
    def __init__(self, N):
        self._EDGE = N
        self.board = makeBoard(N)
        self.stenchBoard = makeBoard(N)
        self.breezeBoard = makeBoard(N)

        print("SIZE = ", len(self.board), ", ", len(self.board[0]))

        # Generating Locations
        # Hero
        self.heroX = randint(0, N - 1)
        self.heroY = randint(0, N - 1)

        print("Hero: ", self.heroX, ", ", self.heroY)
        self.board[self.heroX][self.heroY] = _NOTATIONS.HERO

        # GOLD
        self.goldX = randint(0, N - 1)
        self.goldY = randint(0, N - 1)
        while self.board[self.goldX][self.goldY]:
            self.goldX = randint(0, N - 1)
            self.goldY = randint(0, N - 1)

        print("GOLD: ", self.goldX, ", ", self.goldY)
        self.board[self.goldX][self.goldY] = _NOTATIONS.GOLD
        # Not Generating glitter in this version

        # WUMPUS
        self.wumpusX = randint(0, N - 1)
        self.wumpusY = randint(0, N - 1)
```

```
def moveSelf(self, direction):
    if direction == 'left':
        if self.xCoord - 1 < 0:
            return False
        self.xCoord -= 1
    elif direction == 'right':
        if self.xCoord + 1 > self.board.xEdge:
            return False
        self.xCoord += 1
    elif direction == 'up':
        if self.yCoord - 1 < 0:
            return False
        self.yCoord -= 1
    elif direction == 'down':
        if self.yCoord + 1 > self.board.yEdge:
            return False
        self.yCoord += 1
    return True

def moveDrone(self, direction):
    drone = self.drone
    if direction == 'left':
        if drone.xCoord - 1 < 0:
            return "Invalid"
        drone.xCoord -= 1
    elif direction == 'right':
        if drone.xCoord + 1 > drone.board.xEdge:
            return False
        drone.xCoord += 1
    elif direction == 'up':
        if drone.yCoord - 1 < 0:
            return False
        drone.yCoord -= 1
```

Note: Full code in the zip folder