Name: Saurabh Jawahar Kakade

Course: CS570/Spring 2021/Advanced Intelligence Systems

Assignment Title: Fred Flintstone Problem Solving / Program 01 / Part 01

Date: 01/29/2021

Output:

```
File  Edit  Selection  View  Go  Run  Terminal  Help                    project1_test1.py - Assign
project1_funs.py | project1_test1.py × | interpreter.py | code backup.py | twl06.txt | board.txt
AI_a1_Final > project1_test1.py > ...
from project1_funs import *
myBoard = loadBoard("board.txt")
printBoard(myBoard)
possibleMoves((3,3),myBoard)
possibleMoves((2,1),myBoard)
legalMoves( possibleMoves((1,2), myBoard), ( (1,0),(2,0),(2,1),(2,2) ))
legalMoves( possibleMoves((2,2), myBoard), ( (1,1),(1,2),(1,3),(2,3),(3,2) ) )
myDict = frozenset(word.strip() for word in open("twl06.txt"))
examineState(myBoard,(0,3),( (1,1), (0,1),(0,2) ) ,myDict)
examineState(myBoard,(0,0),( (3,3), (2,2), (1,1) ) ,myDict)
examineState(myBoard,(3,3),( (2,2),(2,1),(2,0),(3,0),(3,1),(3,2) ) ,myDict)

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL

PS N:\Spring 2021\Subjects\AI\Assignments\Assignment 01 Game\AI_a1_Final> python interpreter.py project1_test1.py
>>

>>>

>>> from project1_funs import *

>>> myBoard = loadBoard("board.txt")

>>> printBoard(myBoard)
J O P Y
C M P V
X F E G
P G V U

>>> possibleMoves((3,3),myBoard)
[(2, 2), (2, 3), (3, 2)]

>>> possibleMoves((2,1),myBoard)
[(1, 0), (1, 1), (1, 2), (2, 0), (2, 2), (3, 0), (3, 1), (3, 2)]

>>> legalMoves( possibleMoves((1,2), myBoard), ( (1,0),(2,0),(2,1),(2,2) ))
{(0, 1), (2, 3), (0, 2), (1, 1), (0, 3), (1, 3)}

>>> legalMoves( possibleMoves((2,2), myBoard), ( (1,1),(1,2),(1,3),(2,3),(3,2) ) )
{(3, 1), (3, 3), (2, 1)}

>>> myDict = frozenset(word.strip() for word in open("twl06.txt"))

>>> examineState(myBoard,(0,3),( (1,1), (0,1),(0,2) ) ,myDict)
('mopy', 'Yes')

>>> examineState(myBoard,(0,0),( (3,3), (2,2), (1,1) ) ,myDict)
('uemj', 'No')

>>> examineState(myBoard,(3,3),( (2,2),(2,1),(2,0),(3,0),(3,1),(3,2) ) ,myDict)
('efxpgvu', 'No')

PS N:\Spring 2021\Subjects\AI\Assignments\Assignment 01 Game\AI_a1_Final>
Python 3.8.5 64-bit (conda)   ⊗ 0 ⚠ 1
```

Source Code:

AI_a1_Final > project1_funs.py > ...

```python
#AIS Programming assignment: 01 (Part 01)
#Programmer: Saurabh Jawahar Kakade
#NAU Email ID: sk2354@nau.edu

import string

#load board function
def loadBoard(newboard):
    board = open(newboard,"r")
    new_board = []
    for i in board:
        new_board.append(i.split())
    return new_board

#print board function
def printBoard(newboard):
    for i in newboard:
        print(' '.join(map(str,i)))

#possible move function
def possibleMoves(moves,newboard):
    newpmoves = list() #possible moves list
    x = moves[0] #row
    y = moves[1] #column

    l = len(newboard)

    #top coordinates
    if (x-(x-1)==1) & (y-(y-1)==1):
        if (l>(x-1)>-1) & (l>(y-1)>-1):
            newpmoves = newpmoves + [(x-1,y-1)]

    if (x-(x-1)==1):
        if (l>(x-1)>-1) & (l>(y)>-1):
            newpmoves = newpmoves + [(x-1,y)]

    if (x-(x-1)==1) & (((y+1)-y)==1):
        if (l>(x-1)>-1) & (l>(y+1)>-1):
```

AI_a1_Final > project1_funs.py > ...

```python

    if (x-(x-1)==1) & (((y+1)-y)==1):
        if (l>(x-1)>-1) & (l>(y+1)>-1):
            newpmoves = newpmoves + [(x-1,y+1)]

    #left coordinates
    if (y-(y-1)==1):
        if (l>(x)>-1) & (l>(y-1)>-1):
            newpmoves = newpmoves + [(x,y-1)]

    #right coordinates
    if (((y+1)-y)==1):
        if (l>(x)>-1) & (l>(y+1)>-1):
            newpmoves = newpmoves + [(x,y+1)]

    #bottom coordinates
    if ((x+1)-x==1) & (y-(y-1)==1):
        if (l>(x+1)>-1) & (l>(y-1)>-1):
            newpmoves = newpmoves + [(x+1,y-1)]

    if ((x+1)-x==1):
        if (l>(x+1)>-1) & (l>(y)>-1):
            newpmoves = newpmoves + [(x+1,y)]

    if ((x+1)-x==1) & (((y+1)-y)==1):
        if (l>(x+1)>-1) & (l>(y+1)>-1):
            newpmoves = newpmoves + [(x+1,y+1)]

    else:
        print("error")

    return newpmoves

#Legal moves function
def legalMoves(m0,m1):  #m0 is all possible moves and m1 is visited moves
    l1 = set(m1)
    l2 = set(m0)
    lm = set(l2 - l1) #legal moves set
    # print(lm)
    return lm
```

AI_a1_Final > project1_funs.py > possibleMoves

```python
    l2 = set(m0)
    lm = set(l2 - l1) #legal moves set
    # print(lm)
    return lm

#examine state function
def examineState(myboard,m1,m2,myDict):
    #m1=current position
    #m2=current path
    es = list() #examine state list variable


    for i in m2:
        x, y = i    #x=row and y=column
        es.append(myboard[x][y])

    x, y = m1

    es.append(myboard[x][y])

    #fes variable = final examine state
    fes = "".join(map(str,es))


    if fes.lower() in myDict:
        print((fes.lower(),"Yes"))
    else:
        print((fes.lower(),"No"))


    #end of code

```