

Name: Chen Hao Cheng
AI, CSCI3202, PS2

1)The SIM game

The approach I have is I made two players, one AI, one the real player. For the user part, I check if the move is free, meaning that doesn't get the duplicate move that has already been in the list, and then check if that move will make the user lose the game. For AI part, I made the AI not to lose out of choosing the move, if it still chooses the move that makes AI lose, then it really loses the game.

```
#!/bin/env python
import random
```

```
def whoGoFirst():
    turn = raw_input("Are you playing Red? (Y/N) \n").upper()

    if turn == 'Y':
        return 'Player'
    else:
        return 'Computer'

def getPlayerMove():
    move1, move2 = raw_input("RED move: (with a comma)\n").upper().split(",")
    if move1 not in nodeList or move2 not in nodeList:
        print ("invalid input!")
        getPlayerMove()
    else:
        return move1, move2

def connectMove(move1, move2):
    line.append([move1, move2])
    print line

def isEdgeFree(move1, move2):
    if [move1, move2] not in line and [move2, move1] not in line:
        connectMove(move1, move2)
        return True
    else:
        print ("This move is not valid")
        return False
```

```

def getComputerMove():
    randomNode1, randomNode2 = random.sample(xrange(0,7), 2)
    if randomNode1 == 0:
        randomNode1 = 'A'
    if randomNode1 == 1:
        randomNode1 = 'B'
    if randomNode1 == 2:
        randomNode1 = 'C'
    if randomNode1 == 3:
        randomNode1 = 'D'
    if randomNode1 == 4:
        randomNode1 = 'E'
    if randomNode1 == 5:
        randomNode1 = 'F'
    if randomNode1 == 6:
        randomNode1 = 'G'
    if randomNode1 == 7:
        randomNode1 = 'H'

    if randomNode2 == 0:
        randomNode2 = 'A'
    if randomNode2 == 1:
        randomNode2 = 'B'
    if randomNode2 == 2:
        randomNode2 = 'C'
    if randomNode2 == 3:
        randomNode2 = 'D'
    if randomNode2 == 4:
        randomNode2 = 'E'
    if randomNode2 == 5:
        randomNode2 = 'F'
    if randomNode2 == 6:
        randomNode2 = 'G'
    if randomNode2 == 7:
        randomNode2 = 'H'

    return randomNode1, randomNode2

def isLoser(loserMove1, loserMove2):
    for key1 in line:

```

```

if key1[0] == loserMove1:
    for key2 in line:
        if key2[0] == key1[1]:
            if key2[1] == loserMove2:
                return True
            print (loserMove1, loserMove2)
        else:
            return False

```

```

def playAgain():
    return raw_input("Play again? (y/n)").lower().startswith('y')

```

```

# dictionary
nodeList = {"A": 0, "B": 1, "C": 2, "D": 3, "E": 4, "F": 5, "G": 6, "H": 7 }

```

```

startGame = True
isEdgeSafe = False
line = []
print ("Welcome to the SIM game!")
while True:
    turn = whoGoFirst()
    print (turn + " go first")

```

```

while startGame:
    if turn == 'Player':
        playerMove1, playerMove2 = getPlayerMove()
        print ("RED move: " + playerMove1 + playerMove2)
        isEdgeSafe = True

```

#check if the edge is free first, then check if the chosen moves

would make user self lose

```

if isEdgeFree(playerMove1, playerMove2):
    print "player select right edge"
    if isLoser(playerMove1, playerMove2):
        print ("Player loses!")
        startGame = False

```

```

else:
    turn = 'Computer'

```

```

else:

```

#Check if the AI's moves makes AI lose the game first, if not, check

#if the edge is available. If moves make AI lose, then game over

computerMove1, computerMove2 = getComputerMove()

print ("BLUE move: " + computerMove1 + computerMove2)

if not isLoser(computerMove1, computerMove2):

 if isEdgeFree(computerMove1, computerMove2):

 print "Computer select right edge"

 turn = 'Player'

elif isLoser(computerMove1, computerMove2):

 print ("Computer loses!")

 startGame = False

else:

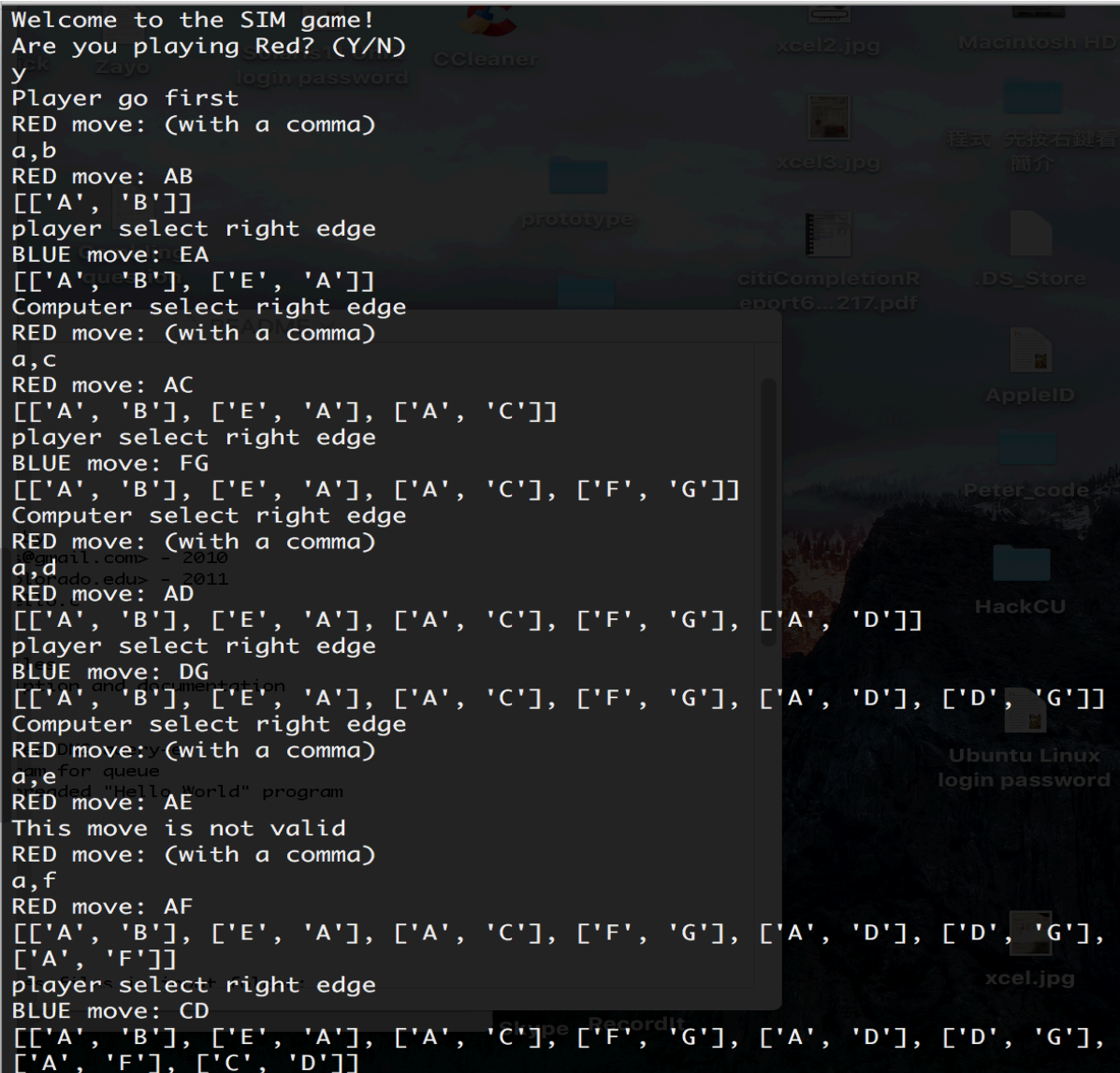
 #This makes AI choose new moves again if the moves it chose makes

 # Ai self lose

 turn = 'Computer'

if not playAgain():

 break



```
Welcome to the SIM game!
Are you playing Red? (Y/N)
y
Player go first
RED move: (with a comma)
a,b
RED move: AB
[['A', 'B']]
player select right edge
BLUE move: EA
[['A', 'B'], ['E', 'A']]
Computer select right edge
RED move: (with a comma)
a,c
RED move: AC
[['A', 'B'], ['E', 'A'], ['A', 'C']]
player select right edge
BLUE move: FG
[['A', 'B'], ['E', 'A'], ['A', 'C'], ['F', 'G']]
Computer select right edge
RED move: (with a comma)
a,d
RED move: AD
[['A', 'B'], ['E', 'A'], ['A', 'C'], ['F', 'G'], ['A', 'D']]
player select right edge
BLUE move: DG
[['A', 'B'], ['E', 'A'], ['A', 'C'], ['F', 'G'], ['A', 'D'], ['D', 'G']]
Computer select right edge
RED move: (with a comma)
a,e
RED move: AE
This move is not valid
RED move: (with a comma)
a,f
RED move: AF
[['A', 'B'], ['E', 'A'], ['A', 'C'], ['F', 'G'], ['A', 'D'], ['D', 'G'],
['A', 'F']]
player select right edge
BLUE move: CD
[['A', 'B'], ['E', 'A'], ['A', 'C'], ['F', 'G'], ['A', 'D'], ['D', 'G'],
['A', 'F'], ['C', 'D']]
ended "Hello World" program
```

4) MasterMind Game

```
=====2
Random puzzle is [1, 2, 1]
=====1
([0, 0, 3], 0, 0)
([2, 2, 2], 1, 0)
([1, 1, 2], 1, 0)
([3, 3, 2], 0, 0)
([2, 1, 0], 0, 0)
Your answer is:
[1, 2, 1]
[Mac-632:PS2 user$ python CorrectMasterMind.py]
=====2
Random puzzle is [2, 1, 0]
=====1
([0, 1, 1], 1, 0)
([2, 3, 1], 1, 0)
([1, 0, 1], 0, 0)
([0, 3, 2], 0, 0)
Your answer is:
[2, 1, 0]
[Mac-632:PS2 user$ python CorrectMasterMind.py]
=====2
Random puzzle is [3, 2, 3]
=====1
([2, 0, 1], 0, 0)
([1, 2, 0], 1, 0)
([1, 1, 2], 0, 0)
([0, 2, 3], 2, 0)
Your answer is:
[3, 2, 3]
[Mac-632:PS2 user$ python CorrectMasterMind.py]
=====2
Random puzzle is [1, 2, 2]
=====1
([1, 2, 0], 2, 0)
([2, 2, 0], 1, 0)
([1, 1, 0], 1, 0)
([1, 2, 1], 2, 0)
Your answer is:
[1, 2, 2]
[Mac-632:PS2 user$ ]
```

```
#!/bin/env python
```

```
import random
```

```
ColorList = {'R': 0, 'B': 1, 'O': 2, 'W': 3}
```

```
puzzle = ['R', 'R', 'W']
```

```
numberColors = 4
```

```
numberPostions = len(puzzle)
```

```
def check(guess, checkPuzzle):
```

```
    rightColor = 0
```

```
    #right color and right position
```

```
    rightPosition = 0
```

```
    # initialize the Matrices all 0
```

```
    MatrixPuzzleColor = [0 for i in range(numberColors)]
```

```
    MatrixGuessColor = [0 for j in range(numberColors)]
```

```
    #print MatrixPuzzleColor
```

```
    for e in checkPuzzle:
```

```
        MatrixPuzzleColor[e] = MatrixPuzzleColor[e] + 1
```

```
        #print ("MatrixPuzzleColor is " + str(MatrixPuzzleColor[e]))
```

```
    for f in guess:
```

```
        MatrixGuessColor[f] = MatrixGuessColor[f] + 1
```

```
        #print ("MatrixGuessColor is " + str(MatrixGuessColor[f]))
```

```
    for g in range(len(checkPuzzle)):
```

```
        colorCurrent = checkPuzzle[g]
```

```
        #print colorCurrent
```

```

    if colorCurrent == guess[g]:
        rightPosition = rightPosition + 1

    if MatrixGuessColor[colorCurrent] > 0:
        rightColor = rightColor + 1
        MatrixGuessColor[colorCurrent] =
MatrixGuessColor[colorCurrent] - 1

    rightColor = rightColor - rightPosition
    if(rightColor < 0):
        rightColor = 0

    return (rightPosition, rightColor)

def guess(guessPuzzle, guessPossibleWays):

    # randomly choose one from the 3D array
    TryGuessing = guessPossibleWays[random.randint(0,
len(guessPossibleWays) - 1)]
    #print TryGuessing

    guessX, guessO = check(TryGuessing, guessPuzzle)
    getRidOf = []

    for position in guessPossibleWays:
        checkX, checkO = check(position, TryGuessing)
        # 'or' for string, '|' for number
        if(checkX != guessX) | (checkO != guessO):
            # get rid of any possibility that contains a different answer
            getRidOf.append(position)

```

```
# using remove function to help me  
for i in getRidOf:  
    guessPossibleWays.remove(i)
```

```
if (guessX, guessO) == (numberPostions, 0):  
    print ("Your answer is: ")  
    print (TryGuessing)  
else:  
    print (TryGuessing, guessX, guessO)  
    guess(guessPuzzle, guessPossibleWays)
```

```
print ("=====2")
```

```
# 3D arrays  
possibilities = []  
for a in range(numberColors):  
    for b in range(numberColors):  
        for c in range(numberColors):  
            possibilities.append([a, b, c])
```

```
puzzle = [random.randint(0, 4), random.randint(0,4),  
random.randint(0,4)]  
print ("Random puzzle is " + str(puzzle))
```

```
print ("=====1")  
guess(puzzle, possibilities)
```

```
.....  
mastermind is brute force program.
```

it starts with a list of all possible solutions and it guesses one at random and gets (x,o) feedback. It will look at all the possible solutions, and checks the guess against each of them. Any solution that gives the correct feedback is still a possibility and Any solution that gives different feedback is thrown out.