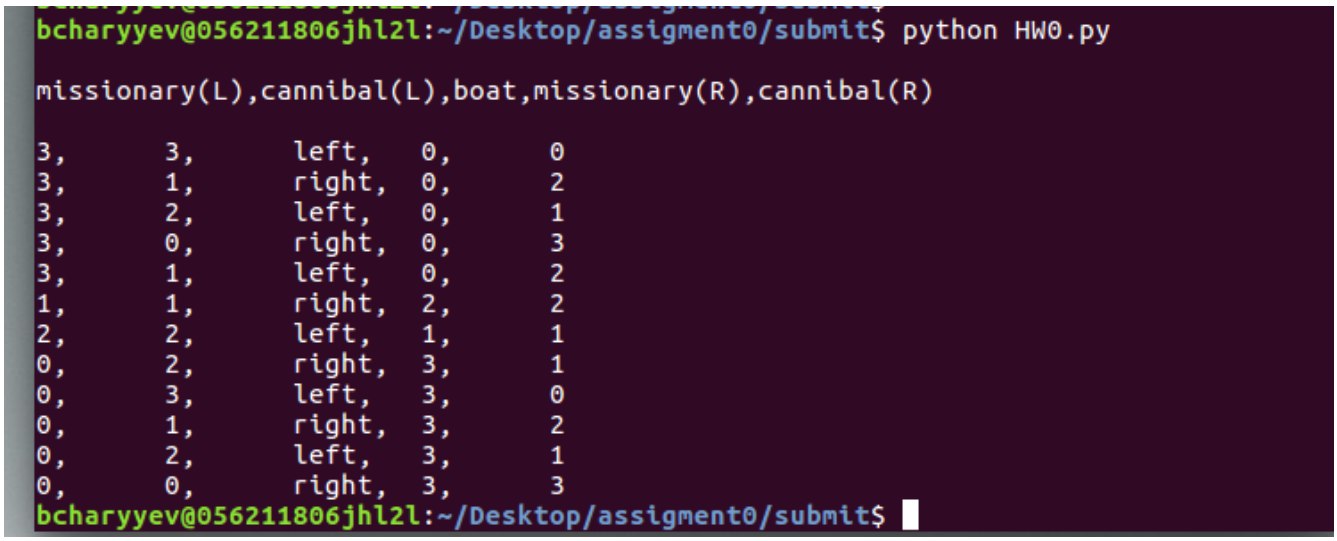
CS 776: Evolutionary Computing -Assignment 0 Fall 2018

Full Name: Batyr Charyyev
Email address: bcharyyev@nevada.unr.edu

Source code: HW0.py
How to run: python HW0.py

Example output:



I used breadth first search (BFS) because depth first search (DFS) is not guaranteed to find solution moreover it is not optimal.

1st column: number of missionary on the left side of the river
2nd column: number of cannibals on the left side of the river

3rd column: shows in which side boat is located

4th column: number of missionary on the right side of the river
5th column: number of cannibals on the right side of the river

```
import Queue

def check(state):
	m=state[0]
	c=state[1]
	b=state[2]
	if m==0 and c==0:
		return "Win"

	if m >= 0 and (3-c) >= 0 \
	    and c >= 0 and (3-c)>= 0 \
	    and (m == 0 or m >= c) \
	    and ((3-m) == 0 or (3-m) >= (3-c)):
	    return "continue"
	return "Lose"

def generateNextStates(state):
	m=state[0]
	c=state[1]
	b=state[2]
	nextSteps=[]

	steps=[]
	if b==1: #if boat is on this side
		#steps.append((m,c,0))
		steps.append((m-1,c,0))
		steps.append((m-1,c-1,0))
		steps.append((m-2,c,0))
		steps.append((m,c-1,0))
		steps.append((m,c-2,0))
	else:
		#steps.append((m,c,1))
		steps.append((m+1,c,1))
		steps.append((m+1,c+1,1))
		steps.append((m+2,c,1))
		steps.append((m,c+1,1))
		steps.append((m,c+2,1))

	for s in steps:
		if (s[0]<0) or (s[1]<0):
			continue

		nextSteps.append(s)

	return nextSteps
```

```
dic={}


def pathPrinter(S):

        level=S[1]
        state=S[0]
        path=[]

        while level!=0:
                path.append(state)
                parent=dic.get((state,level))
                level=parent[1]
                state=parent[0]

        path.append(state)

        print ""
        print "missionary(L),cannibal(L),boat,missionary(R),cannibal(R)"

        print ""

        for p in reversed(path):
                ml=str(p[0])
                cl=str(p[1])
                b="left"
                if p[2]==0:
                        b="right"
                mr=str(3-p[0])
                cr=str(3-p[1])

                print ml+",\t"+cl+",\t"+b+",\t"+mr+",\t"+cr

def search(state):

        L = Queue.Queue()

        L.put([state,0])

        while L.empty() == False:
                S=L.get()
                currentState=S[0]
                currentlevel=S[1]

#               print currentState

                if "Win" == check(currentState):
                        pathPrinter(S)
```

```
                    break

                elif "Lose" == check(currentState):
                    continue
                else:
                    nextStates=generateNextStates(currentState)
                    for s in nextStates:

                        dic[(s,currentlevel+1)]=[currentState,currentlevel]
                        L.put((s,currentlevel+1))


if __name__ == '__main__':

        search((3,3,1)) #MCB
```