```
#River Crossing Problem Part2
        #Coded By Parsa Yousefi Nejad
                #Changes:   //pathShow(), isValid(), isGoal, GenerateAllValidStates()//
Added to the Code

#Importing Necessary libraries
from os import system, name     #To clear() method
from copy import copy           #To shallow copy of an object
from time import sleep          #For implementing pause in pathShow()

#Display:   Shows one Record Graphically
def Display(Record):
    listOfChars =
["POLICE","THIEF","FATHER","MOTHER","DAUGHTER_1","DAUGHTER_2","SON_1","SON_2",]
    shore = ('\x1b[1;32;42m'+"⥯"+'\x1b[0m') * 10
    plainText = '\033[2m'+"❖"+'\x1b[0m'+" {}
"+'\x1b[4;35;43m'+"|"+'\x1b[0m'+'\x1b[1;33;34m' + \
        "~~~~~~~~~~~~~~~~~"+'\x1b[0m'+'\x1b[4;35;43m'+"|" + \
        '\x1b[0m'+" {} "+('\033[2m'+"❖"+'\x1b[0m')

    print(('\033[2m'+"✵"+'\x1b[0m') * 44)
    print(plainText.format(shore, shore))

    for i in range(0, 8):
        characterName = listOfChars[i] + \
            ('\x1b[1;32;42m'+"⥯"+'\x1b[0m') * (10 - len(listOfChars[i]))
        if Record[i] == 0:
            print(plainText.format('\x1b[7;35;46m'+characterName+'\x1b[0m', shore))
        else:
            print(plainText.format(shore, '\x1b[7;35;46m'+characterName+'\x1b[0m'))
        if i == 3:
            if Record[8] == 1:
                print('\033[2m'+"❖"+'\x1b[0m', shore, '\x1b[4;35;43m'+"|"+'\x1b[0m',
'\x1b[1;33;34m'+"~~~~~~~~~~~~"+'\x1b[0m',
                      '\x1b[1;34;41m'+"⛵"+'\x1b[0m', '\x1b[4;35;43m'+"|"+'\x1b[0m',
shore, '\033[2m'+"❖"+'\x1b[0m')
            else:
                print('\033[2m'+"❖"+'\x1b[0m', shore, '\x1b[4;35;43m'+"|"+'\x1b[0m',
'\x1b[1;34;41m'+"⛵ "+'\x1b[0m',
                      '\x1b[1;33;34m'+"~~~~~~~~~~~~"+'\x1b[0m',
'\x1b[4;35;43m'+"|"+'\x1b[0m', shore, '\033[2m'+"❖"+'\x1b[0m')

    print(plainText.format(shore, shore))
    print(('\033[2m'+"✵"+'\x1b[0m') * 44)

#pathShow:  Shows Multiple States in order
def pathShow(List_States):
    if List_States == None:
        print("\x1B[41;2;35mThere is Nothing To Show\033[0m")
        exit(-1)
    Counter = 1
    for state in List_States:
        if Counter !=1 : sleep(1.6)
        clear()
        print(f"\033[3;46;35mChild State {Counter}\033[0m")
```

```
        Display(state)
        Counter += 1

#clear:     Clears Terminal output
def clear():
    if name == 'nt':
        system('cls')
    else:
        system('clear')

#Assigning values to problem members
POLICE = 0 ;THIEF = 1 ;FATHER = 2 ;MOTHER = 3 ;DAUGHTER_1 = 4 ;DAUGHTER_2 = 5 ;SON_1 = 6
;SON_2 = 7 ;BOAT_Direction = 8

#Checks whether a state is valid
def isValid(state):
    return ((state[DAUGHTER_1] == state[MOTHER] or state[DAUGHTER_1] != state[FATHER])
and (
            state[DAUGHTER_2] == state[MOTHER] or state[DAUGHTER_2] != state[FATHER]))
and ((

            state[SON_1] == state[FATHER] or state[SON_1] != state[MOTHER]) and (
            state[SON_2] == state[FATHER] or state[SON_2] != state[MOTHER] )) and (

            state[POLICE] == state[THIEF] or (state[THIEF] != state[FATHER] and
            state[THIEF] != state[MOTHER] and state[THIEF] != state[DAUGHTER_1] and
            state[THIEF] != state[DAUGHTER_2] and state[THIEF] != state[SON_1] and
            state[THIEF] != state[SON_2]
            ))

#checks if the state is Goal
def isGoal(state):
    return state == [1, 1, 1, 1, 1, 1, 1, 1, 1]

#It genrate all states from a valid state and filters all invalid ones
def generateAllValidStates(state):

    if not isValid(state):
        print('\n'+"\x1B[41;1;35mSorry I can\'t Generate States for an Invalid
State\033[0m")

    else:
        valid_states = []
        for _ in [POLICE, THIEF, FATHER, MOTHER ,DAUGHTER_1 ,DAUGHTER_2 , SON_1, SON_2]:

            if state[_] == state[FATHER] == state[BOAT_Direction]:
                new_State = copy(state)
                new_State[FATHER] = 0 if state[FATHER] == 1 else 1
                new_State[_] = 0 if state[_] == 1 else 1
                new_State[BOAT_Direction] = 0 if state[BOAT_Direction] == 1 else 1

                if isValid(new_State) and new_State not in valid_states:
                    valid_states.append(new_State)

            if state[_] == state[MOTHER] == state[BOAT_Direction]:
                new_State = copy(state)
                new_State[MOTHER] = 0 if state[MOTHER] == 1 else 1
                new_State[_] = 0 if state[_] == 1 else 1
                new_State[BOAT_Direction] = 0 if state[BOAT_Direction] == 1 else 1

                if isValid(new_State) and new_State not in valid_states:
                    valid_states.append(new_State)

            if state[_] == state[POLICE] == state[BOAT_Direction]:
                new_State = copy(state)
                new_State[POLICE] = 0 if state[POLICE] == 1 else 1
                new_State[_] = 0 if state[_] == 1 else 1
                new_State[BOAT_Direction] = 0 if state[BOAT_Direction] == 1 else 1

                if isValid(new_State) and new_State not in valid_states:
                    valid_states.append(new_State)

        return valid_states


#main part of Code,PROMPTING user to enter a new state
# ///////////////MAIN//////////////////
Record = [1] * 9
clear()
print('\n', '\x1b[1;30;47m'+'Enter State(0-1) of Characters in order
of:'+'\x1b[0m'+'\n'+'\x1b[1;33;44m' +
        '1:Police  2:Thief  3:Father  4:Mother  5:Daughter_1  6:Daughter_2  7:Son_1
8:Son_2  9:BOAT_Direction'+'\x1b[0m'+'\n\n'+'\x1b[1;31;40m'+'Enter \'exit\' to
quit.'+'\x1b[0m'+'\n\n')
i = -1
while True:
    i += 1
    if i in range(0, 9):
        print('\x1b[1;30;45m'+(str(i+1))+'\x1b[0m', end=': ')
        userInput = input()
        if userInput == 'exit':
            exit()
        elif userInput == '0' or userInput == '1':
            Record[i] = int(userInput)
        else:
            print('\033[91m'+'Invalid Input, Try Again(0-1):'+'\x1b[0m')
            i -= 1
    else:
        break
clear()
print('\n\n','\x1b[3;4;46m' +'1:Police  2:Thief  3:Father  4:Mother  5:Daughter_1
6:Daughter_2  7:Son_1  8:Son_2  9:BOAT_Direction'+'\x1b[0m')
print("State =",'\x1b[1;7;3m'+str(Record)+'\x1b[0m','\n')
Display(Record)
input('\n'+'\x1b[7;33;66m'+'Press Return Key to Execute '+
'\033[1;42;34mgenerateAllValidStates() and PathShow()\033[0m'+'\x1b[7;33;66m'+ ' for
this state'+'\x1b[0m'+'\n\n')
pathShow((generateAllValidStates(Record)))
```