**NAME:ROHAN NYATI**

**SAP ID:500075940**

**ROLL NO. : R177219148**

**BATCH-5(AI&ML)**

**Solving River Crossing Problem using python**

**Code:--**

"""

Wolf, Goat and Cabbage Puzzle in Python

By Robin Andrews - https://compucademy.co.uk/

"""

import os  # Only needed in console version

import time # Only needed in console version

names = {"F": "Farmer",

         "W": "Wolf",

         "G": "Goat",

         "C": "Cabbage"}

forbidden\_states = [{"W", "G"}, {"G", "C"}, {"G", "C", "W"}]

def print\_story():

    print("""

#### WOLF, GOAT and CABBAGE PROBLEM ####

Once upon a time a farmer went to a market and purchased a wolf, a goat, and a cabbage. On his way home, the farmer came

to the bank of a river and rented a boat. But crossing the river by boat, the farmer could carry only himself and a single

one of his purchases: the wolf, the goat, or the cabbage.

If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage.

The farmer's challenge was to carry himself and his purchases to the far bank of the river, leaving each purchase intact.

How did he do it?

""")

    input("Press enter to continue.")

"""

Wolf, Goat and Cabbage Puzzle in Python

"""

names = {"F": "Farmer",

         "W": "Wolf",

         "G": "Goat",

         "C": "Cabbage"}

forbidden\_states = [{"W", "G"}, {"G", "C"}, {"G", "C", "W"}]

def print\_story():

    print("""

#### WOLF, GOAT and CABBAGE PROBLEM ####

Once upon a time a farmer went to a market and purchased a wolf, a goat, and a cabbage. On his way home, the farmer came

to the bank of a river and rented a boat. But crossing the river by boat, the farmer could carry only himself and a single

one of his purchases: the wolf, the goat, or the cabbage.

If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage.

The farmer's challenge was to carry himself and his purchases to the far bank of the river, leaving each purchase intact.

How did he do it?

""")

    input("Press enter to continue.")

def section\_break():

    print("\*" \* 50)

def print\_state(state):

    left\_bank, right\_bank = state

    print("#### CURRENT STATE OF PUZZLE ####")

    print("")

    left\_bank\_display = [names[item] for item in left\_bank]

    right\_bank\_display = [names[item] for item in right\_bank]

    print (left\_bank\_display, "|", right\_bank\_display if right\_bank else "[]") # Add brackets for Python 3

    print("")

def get\_move():

    print("Which item do you wish to take across the river?")

    answer = ""

    while answer.upper() not in ["F", "W", "G", "C"]:

        answer = input("Just Farmer (f), Wolf (w), Goat (g) or Cabbage (c)? ")

    return answer.upper()

def process\_move(move, state):

    # We need to "think ahead" to see if move is illegal.

    temp\_state = [state[0].copy(), state[1].copy()]

    containing\_set = 0 if move in state[0] else 1

    if "F" not in state[containing\_set]:

        print("Not allowed - the farmer must accompany the item.")

        print("")

        return state

    if containing\_set == 0:

        temp\_state[0].difference\_update({move, "F"})

        temp\_state[1].update([move, "F"])

    elif containing\_set == 1:

        temp\_state[1].difference\_update({move, "F"})

        temp\_state[0].update([move, "F"])

    if temp\_state[0] not in forbidden\_states and temp\_state[1] not in forbidden\_states:

        state = [temp\_state[0].copy(), temp\_state[1].copy()]

    else:

        print("Not allowed - one of your items would be eaten!")

    print("")

    return state

def is\_win(state):

    return state[1] == {"F", "W", "G", "C"}

def main():

    left\_bank = {"F", "W", "G", "C"}

    right\_bank = set()

    state = [left\_bank, right\_bank]

    print\_story()

    while not is\_win(state):

        section\_break()

        print\_state(state)

        move = get\_move()

        state = process\_move(move, state)

    print("Well done - you solved the puzzle!")

main()

**Output:--**

****

