NED University Of Engineering And Technology

CS-329 - Operating Systems

Complex Engineering Problem

Group Members

- Rohan Ahmed – CS-19091
- Sohaib Ahmed Abbasi – CS-19096

Problem Chosen

Problem 1: Cigarette smokers' problem.

Code Repository

https://github.com/Sohaib-50/Cigarette-Smokers-Problem

Code

```
import threading
import random
import time

# Global variables
INGREDIENTS_NAMES = {'tobacco', 'paper', 'matches'}
ingredients_semaphores = {
    'tobacco_paper': threading.Semaphore(0),
    'tobacco_matches': threading.Semaphore(0),
    'paper_matches': threading.Semaphore(0)
    }  # Semaphores for the possible pairs of ingredients that the agent may produce
agent_semaphore = threading.Semaphore(0)  # Semaphore synchronization between agent and
smokers

class Agent(threading.Thread):

    def run(self):
        '''
        This method runs the agent thread. It simulates the work of the agent.
        '''
        while True:

            # Choose ingredient pair to make available to smokers
            try:
                random_seed = int(input('\nEnter a number for the random seed: ')) * 100
            except ValueError:
                print("Invalid input, setting random seed to 0")
                random_seed = 0
            random.seed(random_seed)
            ingredient_pair = random.choice(list(ingredients_semaphores.keys()))

            # Make the ingredient pair available to smokers
            ingredients_semaphores[ingredient_pair].release()  # call sem signal on
ingredients pair
            print(f"Agent makes {ingredient_pair} available to smokers")

            # Wait for the smoker that has the complementary ingredient to
            # pick up the ingredients, make cigarette, and finish smoking
```

```
            ingredients_semaphores[ingredient_pair].acquire()  # call sem wait on ingredients
pair


class Smoker(threading.Thread):

    def __init__(self, name, ingredient):
        '''
        initialize the smoker thread.
        '''
        super().__init__()
        self.name = name

        # set the required ingredient pair
        if ingredient == 'tobacco':
            self.required_ingredient_pair = 'paper_matches'
        elif ingredient == 'paper':
            self.required_ingredient_pair = 'tobacco_matches'
        elif ingredient == 'matches':
            self.required_ingredient_pair = 'tobacco_paper'

    def run(self):
        '''
        This method runs the smoker thread. It simulates the work of each smoker.
        '''
        while True:

            # Wait for complimentary ingredients to be available
            print(f"{self.name} waits for {self.required_ingredient_pair}")
            ingredients_semaphores[self.required_ingredient_pair].acquire()  # sem wait on
required ingredient pair semaphore

            # Make cigarette and smoke it
            print(f"{self.name} makes a cigarette and smokes it")
            time.sleep(random.uniform(0.5, 1.5))  # simulate smoking for a random duration of
time between 0.5 and 1.5 seconds

            # Signal agent that the ingredients pair is used up
            print(f"{self.name} signals to agent to make next ingredients pair")
            ingredients_semaphores[self.required_ingredient_pair].release()  # sem signal on
required ingredient pair semaphore

def main():
    '''
    Main program, simulates the Cigarette Smokers' Problem
    '''

    # Create agent and smokers
    agent = Agent()
    smoker1 = Smoker('Smoker with tobacco', 'tobacco')
    smoker2 = Smoker('Smoker with paper', 'paper')
    smoker3 = Smoker('Smoker with matches', 'matches')
    smokers = {smoker1, smoker2, smoker3}

    # Start agent and smokers threads (simulate their work)
    for smoker in smokers:
        smoker.start()
    agent.start()
```

```python
    # Wait for threads to finish
    for smoker in smokers:
        smoker.join()
    agent.join()

# Run the main program
if __name__ == '__main__':
    main()
    print('Done')
    exit(0)
```

About the code

The code relies on python's threading library. The Agent and Smoker classes inherit from threading library's Thread class, making those classes act as thread. The run() method of both classes is overridden to the respective required functionality. Moreover the threading library's Semaphore class is also used to get semaphore functionality; the methods .acquire() and .release() are used to perform functionality of sem_wait and sem_signal respectively.

Test Cases

For all test cases, the behavior of agents and smokers should be as expected, i.e. as demanded by the problem statement. The individual test cases then are based on different types of inputs the user may give. Regardless of input the behavior of agents and smokers should be correct and program shouldn't crash.

Test Case 1

Input: User enters the same valid integer value for random seed every time

Expectation: Agent should produce same pair of ingredients each time and the same smoker should pick them up each time.

Execution:

```
C:\Google Drive\NED\Semester 6\OS\CEP>python cigarette_smokers.py
Smoker with matches waits for tobacco_paper
Smoker with tobacco waits for paper_matches
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 4
Agent makes tobacco_matches available to smokers
Smoker with paper makes a cigarette and smokes it
Smoker with paper signals to agent to make next ingredients pair
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 4
Agent makes tobacco_matches available to smokers
Smoker with paper makes a cigarette and smokes it
Smoker with paper signals to agent to make next ingredients pair
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 4
Agent makes tobacco_matches available to smokers
Smoker with paper makes a cigarette and smokes it
Smoker with paper signals to agent to make next ingredients pair
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 4
Agent makes tobacco_matches available to smokers
Smoker with paper makes a cigarette and smokes it
Smoker with paper signals to agent to make next ingredients pair
Smoker with paper waits for tobacco_matches
```

Output: On user entering 4 each time, agent generates tobacco and matches each time and smoker with paper picks the ingredients up each time and smokes a cigarette.

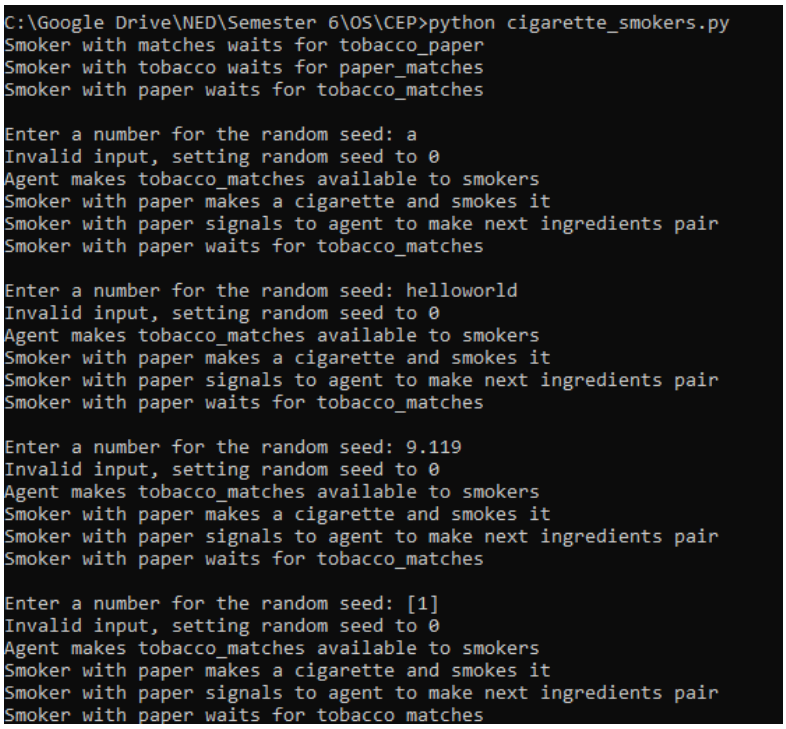
Result: **Test Case Passed.**

Test Case 2

Input: User enters invalid, i.e., non-integer values (characters, strings, floats, etc.)

Expectation: Program should handle invalid inputs and use the default value of 0 for random seed on each wrong input leading to similar kind of agent and smoker behavior as in Test Case 1 as seed would be same each time.

Execution:



Output: As expected.

Result: **Test Case Passed.**

Test Case 3

Input: User enters different but valid positive integer values each time for random seed.

Expectation: Agent should generate different pairs of ingredients some times but may produce same pairs for some different inputs (as there are just 3 possible pairs). The correct agent should pick up the ingredient pair each time.

Execution:

```
C:\Google Drive\NED\Semester 6\OS\CEP>python cigarette_smokers.py
Smoker with matches waits for tobacco_paper
Smoker with tobacco waits for paper_matches
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 3
Agent makes paper_matches available to smokers
Smoker with tobacco makes a cigarette and smokes it
Smoker with tobacco signals to agent to make next ingredients pair
Smoker with tobacco waits for paper_matches

Enter a number for the random seed: 6
Agent makes tobacco_matches available to smokers
Smoker with paper makes a cigarette and smokes it
Smoker with paper signals to agent to make next ingredients pair
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 1993
Agent makes tobacco_paper available to smokers
Smoker with matches makes a cigarette and smokes it
Smoker with matches signals to agent to make next ingredients pair
Smoker with matches waits for tobacco_paper

Enter a number for the random seed: 2
Agent makes tobacco_paper available to smokers
Smoker with matches makes a cigarette and smokes it
Smoker with matches signals to agent to make next ingredients pair
Smoker with matches waits for tobacco_paper

Enter a number for the random seed: 1
Agent makes tobacco_paper available to smokers
Smoker with matches makes a cigarette and smokes it
Smoker with matches signals to agent to make next ingredients pair
Smoker with matches waits for tobacco_paper
```

Output: Same as expected.
Result: **Test Case Passed.**

Test Case 4

Input: User enters different but valid positive and negative integer values each time for random seed.
Expectation: Same as Test Case 3, as negative integers are allowed.
Execution:

```
C:\Google Drive\NED\Semester 6\OS\CEP>python cigarette_smokers.py
Smoker with matches waits for tobacco_paper
Smoker with tobacco waits for paper_matches
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: -13
Agent makes tobacco_matches available to smokers
Smoker with paper makes a cigarette and smokes it
Smoker with paper signals to agent to make next ingredients pair
Smoker with paper waits for tobacco_matches

Enter a number for the random seed: 2
Agent makes tobacco_paper available to smokers
Smoker with matches makes a cigarette and smokes it
Smoker with matches signals to agent to make next ingredients pair
Smoker with matches waits for tobacco_paper

Enter a number for the random seed: -99
Agent makes paper_matches available to smokers
Smoker with tobacco makes a cigarette and smokes it
Smoker with tobacco signals to agent to make next ingredients pair
Smoker with tobacco waits for paper_matches
```

Output: As expected.
Result: **Test Case Passed.**