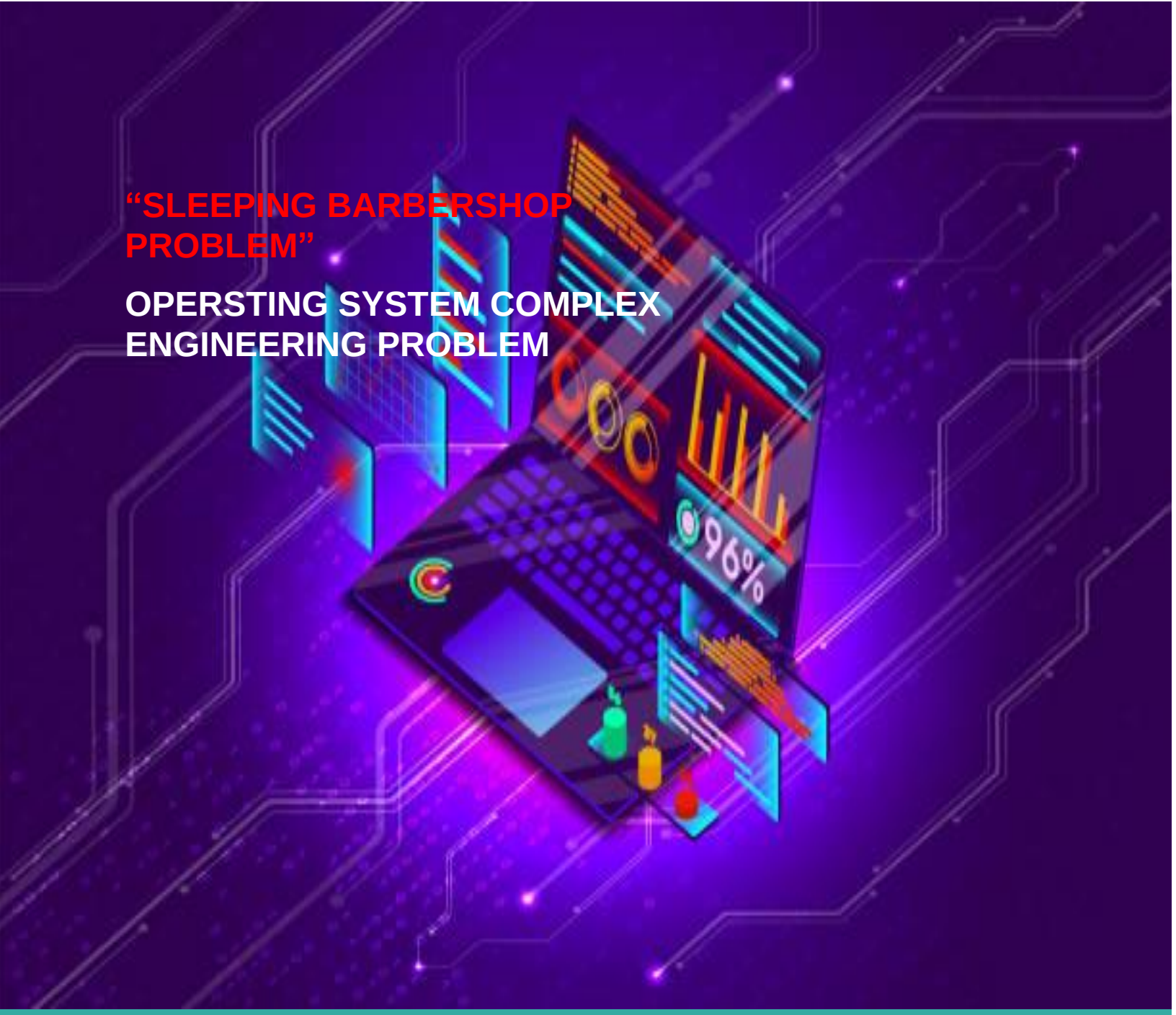


“SLEEPING BARBERSHOP PROBLEM”

OPERSTING SYSTEM COMPLEX ENGINEERING PROBLEM

JULY 23

Name:

- **Khadija**

Roll no:

- **CS-19075**

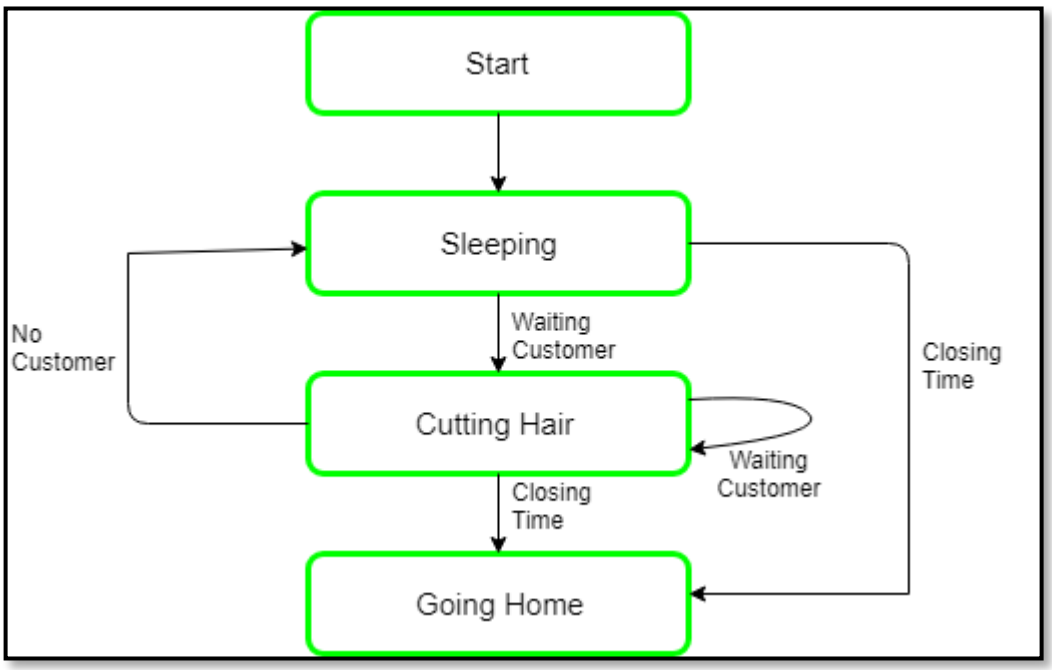
SUBMITTED TO:

Miss Urooj

Problem Statement:

A barbershop consists of a waiting room with n chairs, and a barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

Flow Chart:



Code:

Importing the libraries which I have used to solve this problem. I hard-coded the names of the customer and randomly use them in the procedure.

```
import time
import random
#import Queue
try:
   import queue
except ImportError:
   import Queue as queue
from threading import Thread

NAMES=["Al", "Alex", "Anthony", "Bill", "Bob", "Brad", "Cam", "Cal",
         "Chris", "Charlie", "Dave", "Dan", "Derek", "Devin", "Eric",
         "Elijah", "Frank", "Fred", "Gary", "George", "Hal", "Harry",
         "Isaac", "Ishmael", "Jared", "Jake", "Jeremy", "Kevin", "Kris",
         "Larry", "Louie", "Mark", "Mort", "Nathan", "Norb", "Oscar",
         "Orville", "Peter", "Paul", "Quinn", "Rob", "Rick", "Steve",
         "Tim", "Trevor", "Ulysses", "Victor", "Walter", "Xavier",
         "Yadier", "Zack"]
```

Creating a Class for Barber, make a function as run(), in which the procedures of Barber are performed.

```
class Barber(Thread):

    def __init__(self, waiting_customers):
        super(Barber, self).__init__()
        self.waiting_customers = waiting_customers # the Queue passed in
        self.number_haircuts = 0
        self.tips = 0
        self.days_profit = 0
        self.shop_open = True # Flag to end the main thread
        self.sleeping = True # Flag to have the barber start at sleep until a customer arrives

    def run(self):
        while self.shop_open or not self.waiting_customers.empty(): # continue until Queue is empty or shop is closed
            if not self.waiting_customers.empty():
                self.sleeping = False # if customers awake
                customer = self.waiting_customers.get()
                if customer.hair_type == 'short':
                    customer.in_barber_chair = True
                    print("The barber is cutting %s's hair." % customer.name)
                    print("%s has %s hair, this should take 20 minutes." % (customer.name, customer.hair_type))
                    time.sleep(2)
                elif customer.hair_type == 'medium':
                    customer.in_barber_chair = True
                    print("The barber is cutting %s's hair." % customer.name)
                    print("%s has %s hair, this should take 40 minutes." % (customer.name, customer.hair_type))
                    time.sleep(4)
                else:
                    customer.in_barber_chair = True
                    print("The barber is cutting %s's hair." % customer.name)
                    print("%s has %s hair, this should take 60 minutes." % (customer.name , customer.hair_type))
                    time.sleep(6)
                print("Done cutting %s's hair" % customer.name)
                self.number_haircuts += 1
                self.tips += random.choice(range(6))
                self.days_profit += 15
            elif not self.sleeping: # elif no customers in line and not already sleeping go to sleep
                self.sleeping = True
                print("No one in the waiting room, the barber is going to nap in his chair")
                print("The barber gave %d haircuts and made $%d." % (self.number_haircuts, sum([self.tips, self.days_profit])))
```

For the customer, I made another class, and add an extra feature in it that is:

- Giving random time to long, short, and medium haircuts.

```
class Customer(Thread):

    def __init__(self, name, hair_type):
        super(Customer, self).__init__()
        self.name = name
        self.hair_type = hair_type
        self.in_barber_chair = False

    def run(self):
        while not self.in_barber_chair:
            pass
        if customer.hair_type == 'short':
            customer.in_barber_chair = True
            print("The barber is cutting %s's hair." % customer.name)
            print("%s has %s hair, this should take 20 minutes." % (customer.name, customer.hair_type))
            time.sleep(2)
        elif customer.hair_type == 'medium':
            customer.in_barber_chair = True
            print("The barber is cutting %s's hair." % customer.name)
            print("%s has %s hair, this should take 40 minutes." % (customer.name, customer.hair_type))
            time.sleep(4)
        else:
            customer.in_barber_chair = True
            print("%s has %s hair, this should take 60 minutes." % (customer.name , customer.hair_type))
            time.sleep(6)
```

The code starts to execute from this part.

```
if __name__ == '__main__':
    hair_types = ['short', 'medium', 'long'] # additional variable for a haircut length to simulate time
    chair=int(input("Enter no of chairs:"))
    waiting_room = queue.Queue(chair) # a queue for a waiting room with max input chairs
    barber = Barber(waiting_room)
    barber.start()
    cust=int(input("Enter no of cust:"))
    close_time = 48 # end thread after 48 seconds
    shop_open = time.time()
    current_time = shop_open
    while (current_time - shop_open) < close_time: # check if barber thread has been running longer than open time
        i_need_a_haircut = random.choice(NAMES) # pick a random name
        if cust<=chair:
           if not waiting_room.full(): # if the Queue isn't full grab a seat
            length = random.choice(hair_types)
            customer = Customer(i_need_a_haircut, length)
            waiting_room.put(customer)
            print("%s sat down in the waiting room" % i_need_a_haircut)
            customer.start()
            chair=chair-1

        elif chair!=0:
           if not waiting_room.full(): # if the Queue isn't full grab a seat
            length = random.choice(hair_types)
            customer = Customer(i_need_a_haircut, length)
            waiting_room.put(customer)
            print("%s sat down in the waiting room" % i_need_a_haircut)
            customer.start()
            chair=chair-1
        else:
           print("Sorry, %s too full, try coming back when its not so busy" % i_need_a_haircut)
        stagger = random.choice([1, 5, 10, 20]) # stagger time each customer thread is created
        time.sleep(stagger)
        current_time = time.time()
    barber.shop_open = False # close shop/barber will finish off remaining threads in queue.
```

Output:

```
========== RESTART: E:\6th semester\Operati
Enter no of chairs:1
Enter no of cust:2
Yadier sat down in the waiting roomThe barber is cutting Yadier's hair.

The barber is cutting Yadier's hair.Yadier has medium hair, this should take 40 minutes.

Yadier has medium hair, this should take 40 minutes.
Sorry, Alex too full, try coming back when its not so busy
Done cutting Yadier's hair
No one in the waiting room, the barber is going to nap in his chair
The barber gave 1 haircuts and made $19.
Sorry, Paul too full, try coming back when its not so busy
```

```
Enter no of chairs:2
Enter no of cust:1
Kevin sat down in the waiting roomThe barber is cutting Kevin's hair.

Kevin has long hair, this should take 60 minutes.Kevin has long hair, this should take 60 minutes.

George sat down in the waiting room
Sorry, Xavier too full, try coming back when its not so busy
Done cutting Kevin's hair
The barber is cutting George's hair.The barber is cutting George's hair.

George has short hair, this should take 20 minutes.George has short hair, this should take 20 minutes.

Done cutting George's hair
No one in the waiting room, the barber is going to nap in his chair
The barber gave 2 haircuts and made $37.
Sorry, Louie too full, try coming back when its not so busy
Sorry, Yadier too full, try coming back when its not so busy
```

```
=============== RESTART: E:\6th semester\Operating System\new.py ===============
Enter no of chairs:2
Enter no of cust:2
The barber is cutting Jake's hair.Jake sat down in the waiting room

Jake has short hair, this should take 20 minutes.The barber is cutting Jake's hair.

Jake has short hair, this should take 20 minutes.
Done cutting Jake's hair
No one in the waiting room, the barber is going to nap in his chair
The barber gave 1 haircuts and made $19.
The barber is cutting Mark's hair.Mark sat down in the waiting room

Mark has medium hair, this should take 40 minutes.The barber is cutting Mark's hair.

Mark has medium hair, this should take 40 minutes.
Done cutting Mark's hair
No one in the waiting room, the barber is going to nap in his chair
The barber gave 2 haircuts and made $35.
Sorry, Gary too full, try coming back when its not so busy
```

```
Enter no of chairs:0
Enter no of cust:1
Sorry, Kevin too full, try coming back when its not so busy
```

Test Cases

Test Case1:

Total no of chairs	Used chairs	Expected output	Actual output	Status
1	2	Only one customer can serve and others have to go back	Barber gave only 1 haircut	**Passed**

Test Case2:

Total no of chairs	Used chairs	Expected output	Actual output	Status
2	1	One customer come and get a haircut, but the barber has 2 chairs, afterward another customer come and got a haircut	Barber gave only 2 haircut	**Passed**

Test Case3:

Total no of chairs	Used chairs	Expected output	Actual output	Status
2	2	Customers come and get haircuts one by one both customers get a haircut	Barber gave 2 haircuts	**Passed**

Test Case4:

Total no of chairs	Used chairs	Expected output	Actual output	Status
0	0	Generate an error message that the barber is so much busy	Comeback when is not too busy	**Passed**