PROJECT REPORT

OPERATING SYSTEM (CS-329)

"THE BARBERSHOP PROBLEM"

GROUP MEMBERS:

MISHA AKRAM BAIG (CS-18118)
IQRA IRFAN (CS-18123)

ABSTRACT

In this project we have implemented a Barber Shop classical synchronization problem using semaphores and threads. Barber keep working when there are customers, resting when there are none, and doing so in an orderly manner. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it.

Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves.

SOURCE CODE:

```
import threading
import time
import random

# Haircut start signal
ready = threading.Semaphore(0)
# Haircut end semaphore
finish = threading.Semaphore(0)
# Mutually exclusive semaphore
mutex = threading.Lock()
b = threading.Lock()
# Initial number of haircuts
wait_customer = 0
# Waiting chairs are mutually exclusive
wchair = threading.Lock()
# Barber chairs are mutually exclusive
barber_chair = threading.Lock()
# serial number
counter = 0
# Waiting queue
wchair_list = []
in_cut = 0
served_customer=0

def customer():
    # Customer Process
    global wait_customer, waiting_chair, mutex, wchair, finish, barber_chair, ready, counter, in_cut, served_customer
    time.sleep(4)
    # Enter the critical area, modify the number of people waiting
    mutex.acquire()
    if wait_customer <= waiting_chair:
        wait_customer += 1
        served_customer+=1
        counter += 1
        print("Customer {} is coming".format(counter))
        wchair_list.append("customer"+str(counter))
        print("The current number of customers is:", wait_customer)
        if (waiting_chair>0):
            print("The waiting queue is:", wchair_list)
        mutex.release()
        # Wait for the chair resource to be available
        wchair.acquire()
        # Waiting for haircut
        barber_chair.acquire()
        # You can have a haircut and release the waiting chair resources
        try:
            in_cut = wchair_list.pop(0)
        except:
            pass
```

```
        wchair.release()
        print("{} sat in the barber chair".format(in_cut))
        if (waiting_chair>0):
            print("The remaining waiting queue is:", wchair_list)
        # Send a signal to the barber
        ready.release()
        # Waiting for haircut
        finish.acquire()
        # Release barber chair resources
        print("The {} left the barber chair and walked out of the store happily! ".format(in_cut))
        barber_chair.release()
        # Modify the number of customers
        mutex.acquire()
        wait_customer -= 1
        mutex.release()
    else:
        counter += 1
        print("Customer {} is coming".format(counter))
        balk(counter)


def balk(counter):
    global waiting_chair
    if (waiting_chair>0):
        print("The waiting number is full, there are no seats, the customer {} leaves".format(counter))
    elif (waiting_chair==0):
        print("There are no waiting seats, the customer {} leaves".format(counter))
    else:
        print("There is no barber chair, the customer {} leaves".format(counter))
    mutex.release()


def barber():
    # Barber process
    print("The barber shop is open! ")
    print("-------------------------")
    global wait_customer, waiting_chair, mutex, ready, finish
    while True:
        if wait_customer == 0:
            print("No customers, sleep! ")
            time.sleep(1)
        # Waiting for the customer to make a request
        ready.acquire()
        cut_hair()
        # Haircut finished
        finish.release()
```

```
def cut_hair():
    print("The customer is getting a haircut! ")
    time.sleep(6)
    print("The haircut is over! ")


if __name__ == "__main__":
    total_chairs = int(input("Please enter the number of chairs in the barber shop:"))
    waiting_chair = total_chairs-1
    total_customers = int(input("Please enter the number of customers that will come in the barber shop:"))
    t1 = threading.Thread(target=barber)
    t1.start()
    t1.join(1)

    while True:
        try:
            b.acquire()
            if ( total_customers > 0):
                time.sleep(random.randint(1,2))
                t2 = threading.Thread(target=customer)
                t2.start()
                t2.join(1)
                total_customers -= 1
                b.release()

            elif (total_customers <= 0 and finish._value==0):
                b.release()
                time.sleep(15)
                if (served_customer>0):
                     print("Aaah, all done, served {} customers, going to sleep".format(served_customer))
                else:
                    print("no customers for today, barber is still asleep")
                break
        except:
            print("error! ")
            break
```

TEST CASES:

Test Case 1:

Total number of chairs : 2
Number of waiting chairs : 1
Total customers entering in the shop : 2
Description: Customer 1 came in the barber shop & occupy the waiting chair, he calls the barber & having the hair cut meanwhile customer 2 came as the waiting chair is empty he occupied it & waits for his turn. As none of the customer is waiting so barber going to sleep.
Status : Pass

```
= RESTART: C:\Users\faizr\Desktop\Sixth Semester\BarberShop-classical-synchr
ation-problem\sleep.py
Please enter the number of chairs in the barber shop:2
Please enter the number of customers that will come in the barber shop:2
The barber shop is open!
-------------------------
No customers, sleep!
Customer 1 is coming
The current number of customers is: 1
The waiting queue is: ['customer1']
customer1 sat in the barber chair
The remaining waiting queue is: []
The customer is getting a haircut!
Customer 2 is coming
The current number of customers is: 2
The waiting queue is: ['customer2']
The haircut is over!
The customer1 left the barber chair and walked out of the store happily!
customer2 sat in the barber chair
The remaining waiting queue is: []
The customer is getting a haircut!
The haircut is over!
The customer2 left the barber chair and walked out of the store happily!
Aaah, all done, served 2 customers, going to sleep
>>>
```

Test Case 2:

Total number of chairs : 2
Number of waiting chairs : 1
Total customers entering in the shop : 3
Description: Customer 1 came in the barber shop & occupy the waiting chair, he calls the barber & having the hair cut meanwhile customer 2 came as the waiting chair is empty he occupied it waits for his turn. Now customer 3 came, as the waiting chair is already occupied he left the shop without having the haircut. Now none of the customer is waiting so barber going to sleep.
Status : Pass

```
= RESTART: C:\Users\faizr\Desktop\Sixth Semester\BarberShop-classical-syn
ation-problem\sleep.py
Please enter the number of chairs in the barber shop:2
Please enter the number of customers that will come in the barber shop:3
The barber shop is open!
-------------------------
No customers, sleep!
Customer 1 is coming
The current number of customers is: 1
The waiting queue is: ['customer1']
customer1 sat in the barber chair
The remaining waiting queue is: []
The customer is getting a haircut!
Customer 2 is coming
The current number of customers is: 2
The waiting queue is: ['customer2']
Customer 3 is coming
The waiting number is full, there are no seats, the customer 3 leaves
The haircut is over!
The customer1 left the barber chair and walked out of the store happily!
customer2 sat in the barber chair
The remaining waiting queue is: []
The customer is getting a haircut!
The haircut is over!
The customer2 left the barber chair and walked out of the store happily!
Aaah, all done, served 2 customers, going to sleep
>>>
```

Test Case 3:

Total number of chairs : 4
Number of waiting chairs : 3
Total customers entering in the shop : 0
Description: Barber opens the shop, as no customer enters the shop so he still asleep.
Status : Pass

```
= RESTART: C:\Users\faizr\Desktop\Sixth Semester\BarberShop-classical-synch
ation-problem\sleep.py
Please enter the number of chairs in the barber shop:4
Please enter the number of customers that will come in the barber shop:0
The barber shop is open!
--------------------------
No customers, sleep!
no customers for today, barber is still asleep
>>>
```

Test Case 4:

Total number of chairs : 0
Number of waiting chairs : -1
Total customers entering in the shop : 4
Description: As there are no chairs for the customers to sit so all of the customers came & left the shop without having the haircut.
Status : Pass

```
= RESTART: C:\Users\faizr\Desktop\Sixth Semester\BarberShop-classical-syn
ation-problem\sleep.py
Please enter the number of chairs in the barber shop:0
Please enter the number of customers that will come in the barber shop:4
The barber shop is open!
--------------------------
No customers, sleep!
Customer 1 is coming
There is no barber chair, the customer 1 leaves
Customer 2 is coming
There is no barber chair, the customer 2 leaves
Customer 3 is coming
There is no barber chair, the customer 3 leaves
Customer 4 is coming
There is no barber chair, the customer 4 leaves
no customers for today, barber is still asleep
>>>
```

Test Case 5:

Total number of chairs : 1
Number of waiting chairs : 0
Total customers entering in the shop : 2
Description: Customer 1 came to the shop & having the haircut meanwhile customer 2 came but there are no waiting seats so he left the shop without having the haircut.
Status : Pass

```
ation-problem\sleep.py
Please enter the number of chairs in the barber shop:1
Please enter the number of customers that will come in the barber shop:2
The barber shop is open!
--------------------------
No customers, sleep!
Customer 1 is coming
The current number of customers is: 1
customer1 sat in the barber chair
The customer is getting a haircut!
Customer 2 is coming
There are no waiting seats, the customer 2 leaves
The haircut is over!
The customer1 left the barber chair and walked out of the store happily!
Aaah, all done, served 1 customers, going to sleep
>>>
```