PES University, Bengaluru

(Established under Karnataka Act No. 16 of 2013)

UE18CS306B

OCTOBER 2020: IN SEMESTER ASSESSMENT B Tech 5 SEMESTER

TEST – 1

UE18CS306B (2 credit subject) - Python Application Programming

Time: 1 Hr	Answer All Questions	Max Marks: 30

1.	a)	**A Movie Theatre has a loyalty program that notifies all its patrons every time a new movie is released. Using the appropriate design pattern, write a program that emulates this scenario.**
		
		We use the Observer pattern for this instance:
		
		class Theatre:
		def __init__(self):
		self._listOfUsers = []
		self.moviename = None
		def register(self, userObj):
		if userObj not in self._listOfUsers:
		self._listOfUsers.append(userObj)
		def unregister(self, userObj):
		self._listOfUsers.remove(userObj)
		
		def notifyAll(self):
		for objects in self._listOfUsers:
		objects.notify(self.moviename)
		#event handler
		def Newmovie(self, moviename):
		# User writes a movie.
		self.moviename = moviename
		# When submits the movie is published and notification is sent to all
		self.notifyAll()
		
		class Subscriber:
		def __init__(self):
		pass
		def notify(self):
		#OVERRIDE
		pass
		class User1(Subscriber):
		def notify(self,moviename):
		print ('User1 notfied of a new movie: ',moviename)#banner
		
		class User2(Subscriber):
		def notify(self,moviename):
		print ('User2 notfied of a new movie: ',moviename)
			8

```
#client interaction
if __name__ == "__main__":
        Theatre = Theatre()
        user1 = User1()
        user2 = User2()
        #subscription
        Theatre.register(user1)
        Theatre.register(user2)
        #event:new movie : published to all subscribers
        Theatre.Newmovie("Avengers : Infinity War")

        Theatre.unregister(user2)
        print()
        Theatre.Newmovie("Avengers : The End Game")
```

b) **What is the output of the following code snippet?** — 2

```
def test(a, b = 0, c = 1):
        x = b
        while x<a:
                yield x
                x += c
print(list(test(3)))
```

[0,1,2]

2. a) Create an email and password validation program that checks for the following criterion: — 10

i) Email can have any number of : — 4+4
   a. lowercase letters,
   b. dots (.) and
   c. numbers
   d. with a 'gmail.com' extension **[a-z]+\.*[a-z0-9]*@gmail.com**

ii) The password must fulfill the following criterion:
   a. Must contain at least 1 Uppercase Letter **[A-Z]+**
   b. Must contain at least 1 Lowercase letter **[a-z]+**
   c. Must contain at least 1 Number **[0-9]+**
   d. Must contain at least 1 Special character from the set **[@ _ $]**

3. a) Write the output of the following code: — 4

```
import threading, os, random, time
lock = threading.Lock()
def i_am_thread1(n):
        print('I am',threading.current_thread().name)
        lock.acquire()
        while(n):
                print(random.choice(range(0,10,2)))
                n-=1
        lock.release()
def i_am_thread2(n):
        print('I am',threading.current_thread().name)
        lock.acquire()
        while(n):
                print(random.choice(range(1,10,2)))
                n-=1
        lock.release()

if __name__ == '__main__':
```

```
        th1 = threading.Thread(target = i_am_thread1 , args = (2,),name = 'groot')
        th2 = threading.Thread(target = i_am_thread2 , args = (3,),name = 'rocket')
        th1.start()
        th2.start()
        th1.join()
        th2.join()
        print("Current thread count",threading.active_count())\

I am groot
*2 even numbers
I am rocket
*3 odd numbers
```

b) With a program, illustrate the Producer-Consumer Problem and how it can be avoided. 6

```
from threading import Thread, Lock, Condition
from time import sleep
import random

buffer = []
#lock = Lock()
condition = Condition()
class ProducerThread(Thread):#start,run,join
        def run(self):
                nums = range(5)#0,1,2,3,4
                global buffer
                while True:
                        num = random.choice(nums)#selects a number at random from nums
                        condition.acquire()#lock.acquire()
                        buffer.append(num)
                        print("Producer: Value Produced: ",num)
                        condition.notify()#notification signal : it does not release the lock
                        condition.release()#lock.release()
                        sleep(random.random())

class ConsumerThread(Thread):#start,run,join
        def run(self):
                global buffer
                while True:
                        condition.acquire()#lock.acquire()
                        if not buffer:
                                print("Consumer: Empty Buffer, consumer waits")
                                condition.wait()#suspended at this line:until it receives the
notification
                                print("Consumer: Producer added a value and notified the
consumer")
                        num = buffer.pop(0)
                        print("\t\t\tConsumer: value consumed: ",num)
                        condition.release()#lock.release()
                        sleep(random.random())

ProducerThread().start()#=> run()
ConsumerThread().start()
```