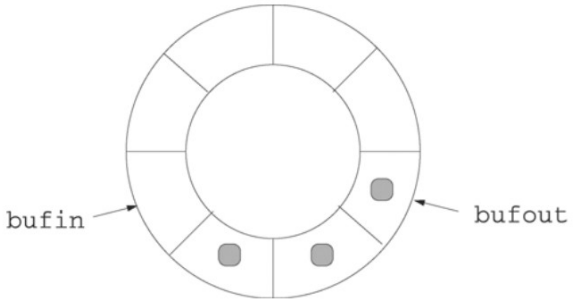
PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERIA
Laboratorio de Sistemas Operativos

CONCURRENCIA EN PYTHON

1) (10 puntos) (nombre del archivo a entregar: ***prodcon.py***) A continuación se tiene el código del productor y consumidor sin sincronizar. En este código hay 4 productores y 4 consumidores. Cada uno de ellos realiza 3 veces su labor, según corresponda.

```
 1  import threading as th
 2  import time
 3  import random
 4
 5  Buffer=[0]*8
 6  bufout = 0
 7  bufin = 0
 8
 9  def getitem(index):
10      global bufout
11
12      item = Buffer[bufout]
13      print(f"c{index} {item} {bufout}")
14      bufout = (bufout + 1) % 8
15      return item
16
17  def putitem(index, item):
18      global bufin
19
20      Buffer[bufin] = item
21      print(f"p{index} {item} {bufin}")
22      bufin = (bufin + 1) % 8
23
24  def consumer(index):
25      for _ in range(3):
26          nextitem = getitem(index)
27
28  def producer(index):
29      for _ in range(3):
30          random.seed(time.time())
31          item = random.randint(10,100)
32          putitem(index,item)
33
34  def main():
35      lconsumer = []
36      lproducer = []
37      for i in range(4):
38          h = th.Thread(target=consumer, args=(i,))
39          lconsumer.append(h)
40          h = th.Thread(target=producer, args=(i,))
41          lproducer.append(h)
42
43      for i in range(4):
44          lconsumer[i].start()
45          lproducer[i].start()
46
47  if __name__ == "__main__":
48      main()
49
```

En este caso se ha empleado un *buffer* circular de 8 elementos. La variable *bufin* apunta a la siguiente posición libre del *buffer*, mientras que la variable *bufout* apunta a la primera posición llena del buffer, tal como se muestra en la figura.



Se le solicita sincronizar el programa para que se cumpla las siguientes restricciones:

a) Los productores deben de abstenerse de producir cuando el *buffer* está lleno.
b) Los consumidores deben de abstenerse de consumir cuando el *buffer* está vacío.
c) Ningún productor puede sobre-escribir un elemento que no ha sido removido del *buffer*.
d) Ningún consumidor remueve un elemento que ya ha sido removido del *buffer*.
e) Ningún consumidor remueve un elemento que un productor está en proceso de insertar en el *buffer*.

Para lograr su propósito solo puede emplear una variable del tipo *Conditional Objects*. No puede usar otras variables locales, ni globales. No puede eliminar líneas del programa.

2) (10 puntos) (nombre del archivo a entregar: ***asesoria.py***) Usted ha sido contratado por la empresa CS Synchronizer para escribir un programa en Python que ayude a sincronizar un profesor y sus alumnos durante las horas de oficina. El profesor, por supuesto, desea tomar una siesta si no hay algún estudiante que quiera preguntar; si hay estudiantes que desean realizar preguntas, ellos deben sincronizarse unos con otros y con el profesor de forma que: (i) solo una persona esté hablando a la vez, (ii) cada pregunta del estudiante es respondida por el profesor, y (iii) ningún estudiante puede hacer otra pregunta antes que el profesor haya contestado la pregunta previa.

Debe escribir cuatro procedimientos: `iniciaRespuesta()`, `terminaRespuesta()`, `iniciaPregunta()` y `terminaPregunta()`.

El profesor repite la ejecución del siguiente código: `iniciaRespuesta(); responde_pregunta; terminaRespuesta()`. `IniciaRespuesta()` no retorna hasta que una pregunta haya sido hecha.

Cada estudiante repite la ejecución del siguiente código: `iniciaPregunta(); realiza_pregunta; terminaPregunta()`. `IniciaPregunta()` no retorna hasta que sea el turno del estudiante de hacer una pregunta.

Dado que el profesor considera de mala educación que un estudiante no espere una respuesta, `terminaPregunta()` no debe regresar hasta que el profesor haya terminado de responder la pregunta.

Elabore el programa empleando *Semaphore Objects*.

Elija impresiones a la pantalla apropiadas para verificar que se cumplen los requisitos. Elabore su código para que el número de alumnos sea aleatorio en cada ejecución.

Usted debe de mostrar una salida análoga a la siguiente:

```
alejandro@abdebian:2020-2$ python3 asesoria.py
Preguntando(0)
Respondiendo(0)
Preguntando(1)
Respondiendo(1)
Preguntando(2)
Respondiendo(2)
Preguntando(5)
Respondiendo(5)
Preguntando(8)
Respondiendo(8)
Preguntando(3)
Respondiendo(3)
Preguntando(12)
Respondiendo(12)
Preguntando(7)
Respondiendo(7)
Preguntando(18)
Respondiendo(18)
Preguntando(21)
Respondiendo(21)
Preguntando(10)
Respondiendo(10)
Preguntando(11)
Respondiendo(11)
Preguntando(6)
Respondiendo(6)
Preguntando(13)
Respondiendo(13)
Preguntando(14)
Respondiendo(14)
Preguntando(15)
Respondiendo(15)
Preguntando(16)
Respondiendo(16)
Preguntando(17)
Respondiendo(17)
Preguntando(4)
Respondiendo(4)
Preguntando(19)
Respondiendo(19)
Preguntando(20)
Respondiendo(20)
Preguntando(9)
Respondiendo(9)
```

Lima, 20 de octubre de 2020.

Prof. Alejandro T. Bello Ruiz