PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 – Programación Avanzada
Interrogación 2

2 de Junio 2015

Duración: 2,5 horas. Sin consultas.

1. (20 pts) Usted tiene una nueva relación de amistad con un robot, el cual le envía mensajes con saludos y buenos deseos todos los días a través de un cariñoso bytearray. Usted ha notado que la cuenta de la compañía que le ofrece servicios de envío de datos ha estado saliendo más cara debido a su nueva relación de amistad. Afortunadamente usted es expert@ en computación y ha detectado una forma que podría hacer que ahorre costos de transferencia de información sin tener que dejar de ser amig@ del robot. Usted se dio cuenta de que en general la mayoría de los bytearrays que le llegan siguen el siguiente patrón: uno de los dos bits ocurre con una frecuencia mucho mayor que el otro. Usted también detectó que los mensajes siempre tienen un largo del orden de los 4000 bits, y el bit menos frecuente tiene una frecuencia de alrededor de un 5 %. Desarrolle un algoritmo que le permita al robot comprimir los mensajes que le envía diariamente, de tal forma de no perder ninguna palabra del mensaje (eso podría herir los sentimientos del robot). Justifique claramente dónde ocurre el ahorro. La solución sólo debe estar en pseudocódigo (con los respectivos comentarios y aclaraciones).

2. (15 pts) Desarrolle los tests unitarios en Python que cree que deberían ser implementados para asegurar el buen funcionamiento del algoritmo propuesto por usted en la alternativa anterior. Puede usar unittest o pytest, asuma que el algoritmo que usted desarrolló en pseudocódigo ya está debidamente implementado (pero no testeado) en Python.

3. a) (5 pts) Mencione una ventaja y una desventaja de JSON sobre Pickle.

   b) (5 pts) Explique para qué sirven los métodos **__enter__** y **__exit__** en una clase cualquiera de Python.

   c) (5 pts) Explique el concepto de "Clase Abstracta", señalando cuáles son las principales ventajas que nos ofrecen este tipo de clases desde el punto de vista del modelamiento.

   d) (5 pts) Suponga que queremos simular una cola de venta de entradas para un concierto, donde se realizará una sola presentación en un local con capacidad limitada. Si tenemos un computador con 1000 cores y queremos correr la simulación para un tiempo de venta de 10^6 segundos con un código que simula en forma síncrona, ¿es posible paralelizar la simulación síncrona y lograr que termine en lo que demora un core en ejecutar 10^3 segundos de simulación? Justifique su respuesta.

4. a) (5 pts) Explique en palabras qué es lo que hace el código a continuación. Escriba todos los detalles necesarios.

   b) (20 pts) Implemente la misma solución utilizando threading.

```python
import random

class Embalse:
    def __init__(self, capacidad):
        self.capacidad = capacidad
        self.nivel = 0

    def lleno(self):
        return self.capacidad <= self.nivel

class Camiones:
    def __init__(self, max_tiempo_viaje, max_tiempo_vaciado):
        self.max_tiempo_viaje = max_tiempo_viaje
        self.max_tiempo_vaciado = max_tiempo_vaciado
        self.tiempo_viaje = random.randint(1, max_tiempo_viaje)
        self.tiempo_vaciado = random.randint(1,max_tiempo_vaciado)
        self.viajando = True

    def cargando(self):
        '''
        El camion viaja para recargarse. Se considera el tiempo de viaje ida y
        vuelata, mas el llenado del camion.
        '''
        self.tiempo_viaje = random.randint(1, self.max_tiempo_vaciado)
        self.tiempo_vaciado = random.randint(1, self.max_tiempo_vaciado)

    def vaciando(self):
        '''
        Cuando el camion regresa, se vierte el contenido en el embalse durante un
        tiempo aleatorio.
        '''
        return self.max_tiempo_viaje + self.tiempo_vaciado

class Simulacion:
    def __init__(self, capacidad_embalse,
                 numero_camiones, max_tiempo_viaje, max_tiempo_vaciado):
        self.tiempo_simulacion = 0
        self.embalse = Embalse(capacidad_embalse)
        self.num_camiones = numero_camiones
        self.lista_camiones = [Camiones(max_tiempo_viaje, max_tiempo_vaciado)
                               for i in range(numero_camiones)]
```

```
50      def run(self):
51          while not self.embalse.lleno():
52              for c in range(self.num_camiones):
53                  '''
54                  El camion va viajando para recargarse, si no se encuentra vaciando
55                  su contenido en el embalse.
56                  '''
57                  if self.lista_camiones[c].viajando:
58                      self.lista_camiones[c].cargando()
59                      self.lista_camiones[c].viajando = False
60                  else:
61                      self.embalse.nivel += 100
62                      self.tiempo_simulacion += self.lista_camiones[c].vaciando()
63                      self.lista_camiones[c].viajando = True
64
65          print("Termino la simulacion")
66          print('Tiempo simulacion: {} minutos'.format(self.tiempo_simulacion))
67
68  if __name__ == "__main__":
69      CAPACIDAD_EMBALSE = 5000
70      MAX_TIEMPO_VIAJE = 10
71      MAX_TIEMPO_VACIADO = 3
72      NUMERO_CAMIONES = 20
73
74      s = Simulacion(CAPACIDAD_EMBALSE, NUMERO_CAMIONES,
75                     MAX_TIEMPO_VIAJE, MAX_TIEMPO_VACIADO)
76      s.run()
```

5. Suponga que un servicio de correo electrónico recibe solo el e–mail (el browser ya llena automáticamente el password, despreocúpese de ese *textbox*) y al intentar acceder al servicio este verifica que es un e–mail válido.

   Una dirección de correo electrónico es válida si cumple con el formato: **x** @ **y**

   - Solo existe el caracter @ entre los strings **x** e **y**
   - El string **y** puede contener más de un dominio (ej: puc.*cl* , ing.puc.*cl*)
   - Solo se permiten dominios chilenos, *i.e*, dominio final **cl**
   - Ningún string puede ser vacío

   a) (15 pts) Escriba en Python la función verificadora.

   b) (5 pts) Dibuje y conecte los *widgets* principales de la interfaz con la función verificadora del *input* del usuario (preocúpese de manejar solo los elementos gráficos relacionados a esta función).