Estructuras de Datos Avanzadas
Convocatoria ordinaria – 07/07/2020
Duración: 2 horas

Universidad Rey Juan Carlos

Normas del examen

- En todos los ejercicios se valorará la eficiencia de las estructuras de datos utilizadas y del código implementado.
- Si un código resuelve el problema de forma no eficiente se podrá considerar como no válido, aunque la salida sea correcta.
- Para comenzar a resolver el examen, descarga el fichero con el código de apoyo del Aula Virtual, descomprime ese fichero y copia el contenido de la carpeta src dentro de la carpeta src de tu proyecto.
- La entrega se basará en un único fichero ZIP que contendrá todos los ejercicios del examen.
- Puedes implementar tantas clases y métodos como necesites, pero sin modificar las cabeceras de los métodos propuestos.
- No se permite la utilización de ningún tipo de apoyo, ni prácticas realizadas previamente, ni comunicación entre estudiantes. Cualquier intento de utilización de este tipo de herramientas implicará el suspenso automático de la convocatoria y las medidas disciplinarias oportunas.

Ejercicio 1 [2 puntos]

Implementa el método `countBalanced` definido en la clase `CountBalanced`. Dado un árbol binario, devuelve cuántos nodos balanceados hay en ese árbol. Un nodo está balanceado si la diferencia en valor absoluto entre la altura máxima de su rama izquierda y su rama derecha es menor o igual que 1.

Ejercicio 2 [2.5 puntos]

Implementa la clase MemoryMap, que añade la siguiente funcionalidad a los mapas:

- `countPutOverwrite`: Cuántas veces se ha sobrescrito una entrada al hacer `put`
- `countFailedRemoves`: Cuántas veces se ha llamado a `remove` sobre una clave que no existe
- `lastQueries`: Qué elementos se han recuperado haciendo `get`. En este caso se guardarán los últimos 5 elementos

Ejercicio 3 [3.5 puntos]

La temporada de piscinas del verano 2020 se plantea bastante complicada para poder gestionar todas las restricciones debidas al COVID19. En primer lugar, se han establecido unos horarios para que los vecinos puedan bajar a la piscina, habiendo un horario diferente para cada día de la semana. El horario de un vecino se define a través de dos enteros que indican la hora de inicio y la de fin en ese día, respectivamente, además del nombre del vecino. El socorrista necesita poder consultar el nombre de los vecinos que pueden estar en la piscina

en un horario determinado. De cada vecino, se quiere guardar los objetos que ha introducido en la piscina a lo largo de la semana para poder rastrear los contagios. Por último, dado un rango horario, se desea obtener el listado de objetos diferentes que se han introducido en la piscina.

Toda la funcionalidad debe estar implementada en la clase PoolManager proporcionada. Se pueden implementar tantas clases, métodos y atributos auxiliares como sean necesarios.

Se pide:

a) **[0.5 puntos]** Definir las estructuras de datos necesarias para almacenar la información indicada.
b) **[1 punto]** Implementar el método neighborsBetween, que dados dos enteros devuelva una colección iterable con el nombre de los vecinos que pueden estar en la piscina en ese horario.
c) **[1 punto]** Implementar los métodos addObject y getObjects. El primero, dado el nombre de un vecino y un objeto, lo almacena como introducido en la piscina. El segundo, dado el nombre de un vecino, devuelve los objetos que ha introducido.
d) **[1 punto]** Implementar el método objectsInPool. Dado un rango horario, devolverá una colección iterable con los objetos introducidos en la piscina.