ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
ESTRUCTURAS DE DATOS
PRIMERA EVALUACIÓN - I TÉRMINO 2018

Nombre: _____ ***Matrícula:*** _____

TEMA 1. (4 PUNTOS)

Elija la opción correcta y justifique su respuesta a las siguientes preguntas:

1. ¿Cuál es el tiempo de ejecución de la función ***encolar*** para colas implementadas por arreglos(n es el número de elementos en la cola)
   a. $O(n^2)$
   b. O(n)
   c. O(1)
   d. O(log n)
2. Si se tiene una tabla de dispersión de 7 cubetas con hashing modular y manejo de colisiones cerrado con rehash lineal, ¿Cuántas colisiones se producen al insertar A = {2,5,16,8,1} ?
   a. 2
   b. 3
   c. 4
   d. ninguna
3. El tiempo de ejecución de la función ***indexOf*** para listas simplemente enlazadas es O(1):
   a. cuando el elemento no está en la lista.
   b. en el mejor caso.
   c. siempre.
   d. nunca.
4. Cuál es el tiempo de ejecución del siguiente algoritmo:

```
SimplyLinkedList<Integer> lista = new SimplyLinkedList<>();
// elementos se agregan al objeto lista
ArrayList<Integer> nueva = new ArrayList<>();
for(int i=0; i<lista.size();i++)
        nueva.addLast(lista.get(i));
```

   a. $O(n^2)$
   b. O(n)
   c. O(1)
   d. O(log n)

TEMA 2. (4 PUNTOS)

Para cada uno de los siguientes escenarios seleccione una o alguna combinación de las siguientes estructuras que pueden resolver el problema y justifique su respuesta:

- Lista basada en arreglos
- Lista Simplemente Enlazada
- Lista Doblemente Enlazada
- Lista Circular Simplemente Enlazada
- Lista Circular Doblemente Enlazada
- Pila
- Cola
- Cola de Prioridad
- HashMap
- HashTable

1. En las pruebas de clasificación de la fórmula 1, los pilotos tienen un tiempo de 15 minutos para realizar su mejor vuelta al circuito luego de varios intentos. El tiempo de cada vuelta al circuito es cronometrado. Al final de la prueba de clasificación se necesita conocer los 3 pilotos más rápidos para entregarles un premio.

2. Las tarjetas de proximidad permiten abrir cerraduras magnéticas. Cada una de las tarjetas tiene un número de serie. En una ciudadela privada tienen 3 cerraduras magnéticas y han decidido entregar tarjetas de proximidad a los propietarios de las viviendas. La administración de la ciudadela requiere implementar un sistema de seguridad que registre los códigos de tarjetas que se utilizaron en cada cerradura para luego verificar si una tarjeta ha sido utilizada en una determinada cerradura.
3. Youtube tiene la opción de reproducción automática de videos a partir de una lista de videos sugeridos. Sin embargo, al finalizar el video que está observando el usuario, aparece el título del siguiente video a reproducirse y la opción de cancelar la reproducción automática.
4. Las personas que viajan en auto y visitan a una persona que vive en una ciudadela privada, deben esperar a que el guardia de seguridad llame al propietario para que autorice el ingreso del visitante.

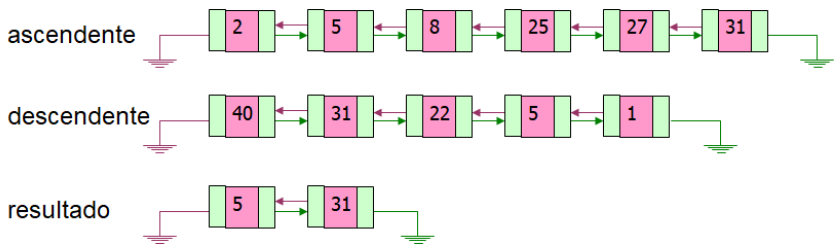
TEMA 3. (10 PUNTOS)

Implementar la función **elementosComunes** de forma recursiva e iterativa. Esta función recibe dos listas ordenadas, la primera lista está ordenada ascendentemente y la segunda lista está ordenada descendentemente. La función debe retornar una nueva lista ordenada ascendentemente con los elementos comunes de ambas listas. La complejidad del algoritmo debe ser O(n); donde n es la dimensión de la lista que tiene la menor cantidad de elementos.
La implementación recursiva debe hacerlo utilizando la LinkedList de java, por lo tanto, la implementación que usted realizará será una función estática en una clase Main.

public static LinkedList<Integer> elementosComunes(LinkedList<Integer> ascendente, LinkedList<Integer> desc)

La implementación iterativa debe hacerlo en la clase DoublyLinkedList, por lo tanto, la lista que estará ordenada ascendentemente es el objeto **this.**

public DoublyLinkedList<Integer> elementosComunes(DoublyLinkedList<Integer> descendente)



TEMA 4. (12 PUNTOS)

En los siguientes problemas debe al menos utilizar una pila en la implementación de su algoritmo.

1) En linux, las rutas absolutas son aquellas que son descritas desde la raíz **(/)**. El punto **(.)** significa mantenerse en el mismo directorio y los dos puntos **(..)** representan regresar un directorio anterior. Usted debe implementar la función **reducirRutaAbsoluta**. La función recibe un string con la ruta absoluta y retorna un string con la simplificación de la ruta.
public static String reducirRutaAbsoluta(String rutaAbsoluta)

Ejemplos:

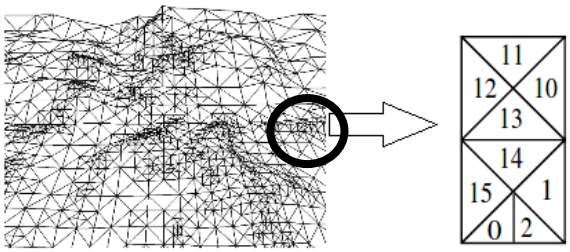
ruta absoluta	/a/./b/../../c/
ruta simplificada	/c
ruta absoluta	/../../../../../a
ruta simplificada	/a
ruta absoluta	/a//b//c/////d
ruta simplificada	/a/b/c/d

2) Implementar la función **eliminarPalabrasRepetidas**. Esta función recibe una lista de palabras y procede a eliminar las palabras repetidas subsecuentes. La función debe retornar una nueva lista de palabras.
public static List<String> eliminarPalabrasRepetidas(List<String> palabras)
Ejemplos:

palabras	[‘abc’,‘aa’,‘aa’,‘cd’]
resultado	[‘abc’, ‘cd’]
palabras	[‘juan’,‘juan’,‘juan’,‘pedro’,‘juan’]
resultado	[‘pedro’,‘juan’]

TEMA 5. (15 PUNTOS)

En videojuegos, la visualización de escenarios 3D irregulares se suele administrar por medio de mallas triangulares con diversos niveles de detalle (LOD). Una malla se puede representar usando un mapa cuya clave es un triángulo y el valor es una lista de sus triángulos adyacentes.



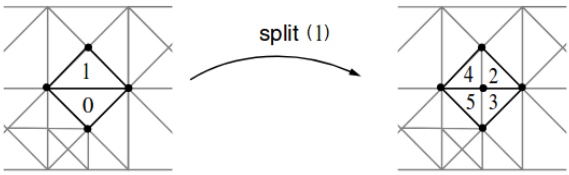
malla = {
13: [12, 10, 14],
14: [13, 15, 1],
15: [14, 0],
2: [1, 0],
...
}

Si se tiene implementada la clase Triángulo con atributos id, posición y área, junto con los métodos getters, setters y hashCode. Además, se define a la resolución de un triángulo como el área del triángulo sumada al promedio de las áreas de sus adyacentes. Implemente el método ***resoluciónRepetida***, que recibe un HashMap con la malla de un escenario y retorna un mapa de triángulos organizados según su resolución, donde la clave es la resolución y el valor es una lista de todos los triángulos con esa resolución. Por ejemplo, si los triángulos 0 y 2 del gráfico anterior tienen un área de 5 y los demás un área de 10, entonces resoluciónRepetida (malla), retorna:
resoluciones = {
20 : [11, 12, 10, 14, 13],
12.5: [0, 2],
17.5: [15, 1]
}

public static Map<Double, LinkedList<Triangulo>> **resoluciónRepetida** (HashMap <Triangulo, LinkedList <Triangulo>> malla)

Asuma que se cuenta con el método split, que recibe un triángulo junto a la malla del escenario y realiza una operación de división sobre ese triángulo para aumentar el nivel de detalle. Además, split retorna la lista de todos los triángulos que fueron creados (Nótese que split altera la cantidad de triángulos dentro de la malla).

LinkedList<Triangulo> split (Triangulo t, HashMap <Triangulo, LinkedList <Triangulo>> malla)



split(1) retorna:
[2,3,4,5]
y altera los valores en la malla.

Implementar el método ***calibrarMalla***, que recibe un HashMap con la malla de un escenario y realiza la operación split sobre los 500 triángulos de mayor resolución (Nótese que los triángulos de mayor resolución pueden variar luego de cada operación split).

public static void calibrarMalla (HashMap <Triangulo, LinkedList <Triangulo>> malla)