ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
ESTRUCTURAS DE DATOS
PRIMERA EVALUACIÓN - I TÉRMINO 2017

Nombre: _____ ***Matrícula:*** _____

TEMA 1. (10 PUNTOS)

Elija la opción correcta según corresponda y **justifique** su respuesta:

1. En una cola implementada con un arreglo circular, cuál es la condición para determinar que la cola está vacía:
    a. siguiente(fin) < inicio
    b. siguiente(fin) == inicio
    c. siguiente(siguiente(fin)) == inicio
    d. siguiente(fin) < inicio

2. En una pila implementada con un arreglo, que referencia/referencias son estrictamente necesarias para implementar las operaciones de push y pop.
    a. una referencia al inicio del arreglo
    b. una referencia al final efectivo del arreglo
    c. una referencia al inicio y al final efectivo del arreglo
    d. N/A es suficiente con el tamaño máximo del arreglo

3. ¿Cuál de las siguientes operaciones tiene un tiempo promedio O(n)?
    a. la cardinalidad de un conjunto basado en un vector de bits.
    b. obtener un elemento por índice en una lista implementada con un arreglo.
    c. pop en una pila implementada con un arreglo.
    d. desencolar en una cola implementada con un arreglo circular.

4. En la operación intersección, en conjuntos implementados con una lista ordenada, usted tiene una referencia al primer conjunto Xa y una referencia al segundo conjunto Xb. Cuál de las siguientes afirmaciones es correcta.
    a. Xa se desplaza a la derecha si Xb < Xa
    b. Xb se desplaza a la derecha si Xa < Xb
    c. Xa y Xb se desplazan a la derecha si son iguales.
    d. Ninguna de las anteriores

5. ¿Cuál de las siguientes operaciones en promedio tiene un tiempo constante?
    a. obtener un elemento por índice en una lista doblemente enlazada.
    b. obtener el nodo anterior al last en una lista simplemente enlazada.
    c. buscar un elemento en un conjunto implementado con una lista ordenada.
    d. buscar un elemento en una tabla de dispersión con hashing cerrado.

TEMA 2. (10 PUNTOS)

Para cada uno de los siguientes escenarios escoja la “mejor” estructura de datos. Puede seleccionar una o una combinación de las siguientes y justifique su respuesta:

- Lista basada en arreglos
- Pila

- Lista Simplemente Enlazada
- Lista Doblemente Enlazada
- Lista Circular Simplemente Enlazada
- Lista Circular Doblemente Enlazada
- Cola
- Cola de Prioridad
- HashMap
- HashTable

1. Se desea llevar el registro de avance de misiones en un juego, cada misión tiene varias sub-misiones, las cuales se puede seguir una por una, y cada sub-misión puede, a la vez, tener otras sub-misiones. Es importante que al cumplir una sub-misión el jugador regrese al punto donde se quedó antes de comenzarla.

2. En una oficina se desea configurar la impresora de red para que imprima según la cantidad de páginas de cada impresión, respetando el orden de solicitud para documentos con la misma cantidad de páginas.

3. Se desea implementar una aplicación en línea que permita consultar, de forma óptima, datos de estudiantes mediante el ingreso de números de matrícula.

4. Se desea implementar una aplicación que simula un juego tragamonedas con 5 cilindros que se encuentran girando a velocidades y direcciones aleatorias, hasta que el usuario presiona una tecla por cada cilindro.

5. Se requiere implementar una aplicación que registra el orden de llegada de los participantes de una maratón, de manera tal que se pueda consultar lo más rápido posible quien llegó en una posición específica.

TEMA 3. (20 PUNTOS)

Considere la siguiente definición de la Lista Doblemente Enlazada:

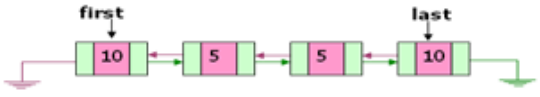
```
public class ListaDoblementeEnlazada {
private NodoLista first;
private NodoLista last;
...
}
```

```
public class NodoLista {
private int data;
private NodoLista next;
private NodoLista previous;
...
}
```

Se le solicita implementar dentro de la ListaDoblementeEnlazada los siguientes métodos recursivos:

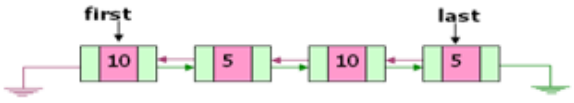
a) **esPalíndromoRecursivo**, que verifica si los elementos de la lista están dispuestos de tal manera que resulta la misma leída de izquierda a derecha que de derecha a izquierda. Por ejemplo:

lista1.esPalindromoRecursivo(); retorna Verdadero



lista1

lista2.esPalindromoRecursivo(); retorna Falso



lista2

```
public boolean esPalíndromoRecursivo()
```

b) **reversarRecursivo**, que invierte el orden de los elementos de la lista. por Ejemplo:

lista.reversarRecursivo(); cambia la lista a:


```
public void reversarRecursivo()
```

TEMA 4. (20 PUNTOS)

Un dispensario médico ha decidido atender a sus pacientes conforme a sus edades. Las personas que tengan mayor edad serán primeros en ser atendidos, sin embargo, el instituto cuenta con solo un médico para atender a todos sus pacientes, por lo tanto, ha decidido asignarle un tiempo de atención para cada paciente de 15 minutos y en caso de que no termine de atenderse en esos 15 minutos debe esperar hasta que el médico siga atendiendo al resto de pacientes. La enfermera del dispensario es la encargada de asignar los tiempos de consultas que requiere cada paciente en su atención referente a la dolencia que presenta y ha generado una lista de pacientes con sus respectivos tiempos necesarios de atención.

[juan de 20 años: 30 minutos, paula de 65 años: 45 minutos, luisa de 65 años: 20 minutos]

Usted deberá crear una simulación de la atención del médico a cada uno de sus pacientes, generando una nueva lista que contiene cada paciente y el tiempo que le resta para terminar de atenderlo. Ejemplo:

[paula: resta 30 minutos, luisa: resta 5 minutos, paula: resta 15 minutos, luisa resta 0 minutos, paula: resta 0 minutos, juan: resta 15 minutos, juan: resta 0 minutos]

```
LinkedList<Paciente> atencion (LinkedList<Paciente> pacientes)

public class Paciente
{
	private String Nombre;
	private int tiempo;
}
```

TEMA 5. (20 PUNTOS)

Considere un laberinto representado con una matriz de enteros, donde los números 1 representan las paredes del laberinto, los números 0 representan caminos del laberinto y los números 2 representan salidas del laberinto. Un ejemplo de la matriz es el siguiente:

1 1 1 1 1 1 1 1 2 1
1 1 0 0 0 0 1 1 0 1
1 1 1 1 1 0 0 0 0 1
0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1

Usted deberá implementar el método **encontrarCamino** que recibe una matriz de enteros (laberinto) y dos números enteros que representan el índice de fila y el índice de columna desde donde usted deberá encontrar la salida al laberinto. La función retorna una lista con las Casillas que conforman el camino hasta la salida del laberinto. Para la implementación de la función, usted cuenta con el TDA Casilla.

```
public LinkedList<Casilla> encontrarCamino(int matriz[][], int x, int y)

public class Casilla {
private int x,y;
...
```

}

TEMA 6. (20 PUNTOS)

Una agencia de eventos almacena los siguientes datos de ciudades: nombre, clima, población, impuestos, costo de vida, locales para eventos. Además, cada local para eventos tiene los siguientes datos: nombre, dirección, capacidad.

a) Se solicita implementar las clases necesarias para representar las entidades descritas en el párrafo anterior.

Además, Implementar los siguientes métodos estáticos:

b) **filtrarPorCapacidad**, que recibe una lista de ciudades y dos números enteros n y c. La función retorna una lista de ciudades que tienen al menos n locales con capacidad mayor a c, ordenadas descendentemente según su costo de vida. Por ejemplo, para las siguientes ciudades:

Guayaquil [local1:1500, local2:1800, local3:2500] costo de vida: 120
Quito [local4:500, local5:2800, local6:1000] costo de vida: 125
Ambato [local7: 500, local8:2300] costo de vida: 115

el método filtrarPorCapacidad (ciudades, 2, 800) retorna:

[Quito, Guayaquil]

```
public static LinkedList<Ciudad> filtrarPorCapacidad(LinkedList<Ciudad> ciudades,int n, int
c)
```

c) **categorizarLocales**, que recibe una lista de ciudades y retorna un mapa que organiza los locales, según su capacidad, en Grandes (sí tienen capacidad para más de 2000 personas) o Medianos (sí tienen capacidad entre 1000 y 2000 personas). Por ejemplo, para las ciudades del ejemplo anterior, el método categorizarLocales retorna:

```
{
 'Medianas' = [local1,local2,local6]
 'Grandes'  = [local3,local5,local8 ]
}
```

```
public static HashMap<String,LinkedList<Local>> categorizarLocales(LinkedList<Ciudad>
ciudades)
```