Programación I

Parcial (turno tarde) – 13/11/2019

Ejercicio 1 *(20 pts. - 5 pts. cada item.)*

Discutir la veracidad de las siguientes afirmaciones, justificando claramente su respuesta.

a) En el siguiente código se produce una excepción:
```
Point[] pts = new Point[2];
pts[0] = new Point(2,3);
System.out.print(pst[0].y + pts[1].y)
```
b) En los métodos recursivos, los argumentos en cada llamada recursiva tienen valores que se acercan cada vez más al caso base, independientemente del valor inicial del argumento.
c) El método QuickSort es siempre de complejidad O(n*log(n)) igual que el método de Burbujeo.
d) Para especificar un TAD es necesario tener definidas ciertas clases y métodos.

Ejercicio 2 *(20 pts.)*

Escribir el método recursivo **public static int** prodCifras(**int** n), que devuelve el producto de las cifras no nulas de un número entero positivo.

Por ejemplo:

- prodCifras(2034) debe devolver 24.
- prodCifras(52) debe devolver 10.
- prodCifras(9) debe devolver 9.
- prodCifras(11020) debe devolver 2.

Se pide resolver **utilizando recursión**.

Ejercicio 3 *(35 pts. - item a. 15 pts. - item b. 20 pts.)*

Consideremos las clases Distribuidora, Producto, Proveedor y Precio definidas como:

```
public class Distribuidora {
    public Producto[] productos;
    ...
}
public class Producto {
    public String marca;
    public Proveedor[] proveedores;
    public Precio[] precios;
    public int stock;
    ...
}
```

```
public class Proveedor {
    public String direccion;
    public String[] barriosDeZonaDistribucion;
    ...
}

public class Precio {
    public double costoUnitario;
    public int descuento; //de 0 a 100
    ...
}
```

Estas clases modelan una distribuidora de bebidas sin alcohol. La distribuidora cuenta con un arreglo de los productos que comercializa. Cada Producto tiene una marca, el stock, un arreglo de proveedores y un arreglo de precios del mismo largo que el arreglo de proveedores que corresponde al precio de venta del producto según cada proveedor.

Para la clase Distribuidora, se pide:

a) Escribir un método **int** cantidadStockBajo() que devuelva la cantidad de productos suyo stock es menor a 5 unidades y que no cuentan con descuento en ningun precio.
b) Escribir un método LinkedList<Proveedor> proveedoresPorZonaYMarca(String marca, String barrio) que devuelva una lista con los proveedores que venden al menos un producto de la marca marca y que tienen a barrio entre los barrios de su zona de distribucion.
c) (*bonus track* 20 pts.) Escribir un método Proveedor elMasVariado() que devuelve el proveedor que puede vender la mayor cantidad de productos de la distribuidora. En caso de existir más de un proveedor con esta característica puede devolver cualquiera de ellos.

Ejercicio 4 *(25 pts.)*

Dadas las clases NodoInt y ListaInt cuyas variables de instancia son las siguientes:

```
public class NodoInt {
    int elemento;
    NodoInt siguiente;
}
```

```
public class ListaInt {
    NodoInt primero;
    ...
}
```

Se pide para esta clase escribir el método de instancia **void** mandarAdelanteCada(**int** k) que modifique la lista eliminando los elementos en todas las posiciones divisibles por k y colocandolos al principio de la lista. Por ejemplo:

- Si la lista es [5,0,8,3,7,9] y k=2 la lista podría quedar como [7,8,5,0,3,9].
- Si la lista es [1,4,7,6,1,5] y k=3 la lista podría quedar como [6,1,4,7,1,5].
- Si la lista es [9,8,7,6,5,4,3,2,1] y k=4 la lista podría quedar como [1,5,9,8,7,6,4,3,2].
- Si la lista es [] y k=3 la lista debe quedar como [].

Observación: puede haber varias formas de acomodar los primeros elementos de la lista, siempre que los que fueron sacados de la lista estén al comienzo y los demás al final. Por ejemplo en el primero de los casos de arriba, otros posibles resultados podrían ser:

- [7,8,5,0,3,9]
- [7,5,8,0,3,9]
- [5,8,7,0,3,9]
- [5,7,8,0,3,9]
- etc.

Se pide además que el método implementado **sea de orden lineal**, es decir, $O(n)$ donde n es la cantidad de elementos de la lista. Justificar la complejidad del mismo.