Problemario para el segundo examen de APALOO

1) Crear un método recursivo que dado dos arreglos con los datos desordenados regrese verdadero si son iguales. Decimos que dos arreglos son iguales si tienen los mismos datos en el mismo orden.

```
public static void main(String[] args) {
      int [] a = {1,2,3,4,5,6,7,8,9,10,20, 15};
      int [] b = {10, 1,2,3,4,5,6,7,8,9,10,20, 15};
      System.out.println(iguales(a,a));
      System.out.println(iguales(b,b));
      System.out.println(iguales(a,b));
   }
```

Regresa:
true
true
false

Importante: iguales(int [] a, int [] b) es la llamada no recursiva, a su vez deberá llamar al método recursivo.
Tip: dos arreglos son iguales si ya se llegó al final de los dos o los dos elementos del frente son iguales y los subarreglos de quitar el primer elemento son iguales (llamado recursivo).

2) Crear un método recursivo que busque un dato en un arreglo no ordenado, deberá de regresar true si lo encuentra, false si no.

Ejemplo:

```
public static void main(String[] args) {
      int [] a = {5,6, 1, 2, 9, 6, 23, 15, 5};

      System.out.println(busca(a,0));
      System.out.println(busca(a,9));
      System.out.println(busca(a,17));
      System.out.println(busca(a,5));

   }
```

Regresa:
false
true
false
true
Importante: busca(int [] datos, int x) es la llamada no recursiva, a su vez deberá llamar al método recursivo.

3) Cree un método recursivo que regrese el último elemento de un arreglo. Asuma que el arreglo por lo menos tiene un elemento.
Ejemplo:

```
public static void main(String[] args) {
      int [] a = {1,2,3,4,5,6,7,8,9,10,20, 15};
      int [] b = {10, 1,2,3,4,5,6,7,8,9,10,20};
      System.out.println(ultimo(a));
      System.out.println(ultimo(b));
   }
```

Regresa:

15

20

Importante: ultimo(int [] datos) es la llamada no recursiva, a su vez deberá llamar al método recursivo.

4) Cree un método recursivo que obtenga la mediana (no la media) de un arreglo de doubles ordenado.

Ejemplo:

```
public static void main(String[] args) {
      double [] a = {1,2,3,4,5,6,7,8,9,10,20};
      double [] b = {1,2,3,4,5,6,7,8,9,10,20,21};
      System.out.println(mediana(a));
      System.out.println(mediana(b));
  }
```

Regresa:

6.0

6.5

Recuerden, si el número de datos es impar regresa el elemento del medio, si es par, regresa la media de los dos elementos centrales.

Importante: mediana(double [] datos) es la llamada no recursiva, a su vez deberá llamar al método recursivo.

Tip: Trátenlo como una búsqueda binaria recursiva, para determinar si el arreglo tiene un número par de elementos la condición de terminación es inicio==fin-1, si el número de elementos es impar la condición de terminación es inicio==fin.

5) Escriba un método recursivo que cree una copia de un String.

Ejemplo:

```
public static void main(String[] args) {
     System.out.println(copia("perro cafe"));
     System.out.println(copia("La casa blanca"));
  }
```

Regresa:

perro cafe

La casa blanca

6) Cree un método recursivo que dado un arreglo regrese verdadero si es un capicúa (que se leen los mismos números de izquierda a derecha que de derecha a izquierda.

Ejemplo:

```
public static void main(String[] args) {
      System.out.println(capicua(new int[] {1,2,3,4,5,4,3,2,1}));
      System.out.println(capicua(new int[] {1,2,3,4,5,4,3,2,3}));
   }
```

Imprime:
true
false

7) Cree un método recursivo que se le pase un arreglo y regrese el arreglo invertido.

Ejemplo:

```
public static void main(String[] args) {
      System.out.println(Arrays.toString(invierte(new int[] {1,2,3,4,5,6,7,8,9})));
   }
```

Imprime:
[9, 8, 7, 6, 5, 4, 3, 2, 1]

8) Dado los siguientes arreglos, ejecute de manera manual la partición de QuickSort y muestre paso a paso la ejecución y la posición final del pivote.
[3,5,1,9,11,2,6]
[19,22,4,16,1]

9) En la clase que desarrollo de una cola implementada de manera ligada (Cola), crear un método recursivo que dado dos colas iguales regrese verdadero. Decimos que dos colas son iguales si tienen los mismos elementos en el mismo orden. Si la cola *a* tiene los elementos [1,2,3,4,5] y la cola *b* tiene los elementos [1,2,3,4,5] decimos que son iguales, por el contrario, si tenemos la cola *c* [1,2,3,4] decimos que *a* y *c* no son iguales porque a la cola *c* le falta un elemento que tiene la cola *a*.
Tip: dos colas son iguales si las dos son nulas o sus primeros elementos son iguales y sus sublcolas son iguales.

```
public boolean iguales(Cola b)
```

Ejemplo en el main

```
public static void main(String[] args) {
      Cola c = new Cola();
      c.insertaOrdR(20);
      c.insertaOrdR(10);
      c.insertaOrdR(15);
      c.insertaOrdR(30);
      System.out.println(c);
      Cola c2 = c.clonar();
      System.out.println(c.iguales(c2));
```

```
        c2.insertaOrdenado(5);
        System.out.println(c.iguales(c2));
        Nodo aux;
        System.out.println((aux=c.ultimo())==null?null:aux.dato);
        Cola c3 = new Cola();
        System.out.println((aux=c3.ultimo())==null?null:aux.dato);
    }
```

10) En la clase que desarrollo de una cola implementada de manera ligada (ColaSimple), crear un método recursivo que regrese la referencia al último elemento de la cola (si tiene por lo menos un elemento) o null si no tiene elementos.

```
public Nodo ultimo()
```