Programación I

Recuperatorio Parcial (turno noche) – 27/06/2019

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

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

a) Los algoritmos de ordenamiento QuickSort y MergeSort tienen ambos el mismo orden de complejidad en el peor caso.
b) Los métodos de clase pueden modificar las variables de instancia.
c) Map es un tipo abstracto de datos.
d) En el siguiente código no se produce aliasing:

```
Point p = new Point(2,1);
Point q = new Point(2,1);
p = q;
```

Ejercicio 2 *(20 pts.)*

Escribir una función recursiva String extremos(String s, **int** n, **int** m) que retorna el string formado por los primeros n caracteres y los últimos m caracteres de s. Se asume que la suma de n + m es menor a la longitud de la cadena s. Por ejemplo:

- extremos("solemnidad", 4, 3) debe devolver "soledad".
- extremos("solitario", 3, 1) debe devolver "solo".
- extremos("soledad", 0, 4) debe devolver "edad".
- extremos("soledad", 3, 0) debe devolver "sol".
- extremos("palabra", 0, 0) debe devolver "".

Se pide resolver **utilizando recursión**. Se pueden dar por hecha la función String resto(String s) que devuelve una cadena igual a s pero sin su primer carácter.

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

Consideremos las clases Aeropuerto, Aerolinea, Avion, Vuelo y Pasajero definidas como:

```
public class Aeropuerto
{
    Aerolinea[] aerolineas;
    ...
}
public class Aerolinea
{
    String nombre;
    Avion[] aviones;
    ...
}
public class Avion{
    String modelo;
    int anioFabricacion;
    Vuelo[] vuelos;
    int capacidad;
    ...
}
```

```
public class Vuelo
{
    int nroVuelo;
    Pasajero[] pasajeros;
    Empleado[] empleados;
    String origen;
    String destino;
    ...
}
public class Pasajero
{
    int pasaporte;
    int edad;
    String nacionalidad;
}
```

Estas clases modelan un aeropuerto. Un Aeropuerto esta compuesto por un arreglo con aerolineas y cada aerolinea tiene un nombre y un arreglo con los aviones que forman parte de ella. A su vez, cada avion tiene su modelo, el año en que fue fabricado y los vuelos que realiza. Por último, el vuelo tiene los empleados que realizan el vuelos, los pasajeros que van en él, un origen, un destino y la capacidad máxima de pasajeros que puede alojar. Para la clase Aeropuerto, se pide:

a) Escribir un método **int** cantViajesDe(Pasajero[] familia) que recibe un arreglo de Pasajero con los pasajeros integrantes de una familia y devuelve la cantidad de vuelos que realizo la familia. Un vuelo fue realizado por la familia, si todos los pasajeros del arreglo familia son pasajeros del vuelo.
b) Escribir un método LinkedList<Avion> avionesLLenos() que devuelva una lista con los aviones que alguna vez viajaron llenos en alguna aerolínea del aeropuerto. Un avión viajo lleno, si realizo algún vuelo, donde la cantidad de pasajeros de ese vuelo es igual a la capacidad del avión.
c) (*bonus track* 20 pts.) Se pide escribir un método String destinoMasPopular() que devuelva el destino que recibió la mayor cantidad de pasajeros, teniendo en cuenta todos los vuelos que realizaron las aerolíneas en el aeropuerto. En caso de haber más de un destino con estas características se 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 escribir el método de instancia **void** duplicarReflejando() que modifica la instancia insertando al comienzo de la misma una copia en orden inverso de los elementos de la lista original. Por ejemplo:

- Si la lista es [1,2,3,4], la lista debería quedar [4,3,2,1,1,2,3,4]
- Si la lista es [5,8,3], la lista debería quedar [3,8,5,5,8,3]
- Si la lista es [1], la lista debería quedar [1,1]
- Si la lista es [], la lista debería quedar [].

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. En el caso de utilizar métodos auxiliares deben escribirlo.