UNIVERSIDAD ORT
Uruguay

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

EVALUACION	EXAMEN	**GRUPO**	Todos	**FECHA**	28/10/2022
MATERIA	Algoritmos y Estructuras de Datos 1				
CARRERA	Analista en Tecnologías de Información / Analista Programador				
CONDICIONES	**Puntos: 100 MÍNIMO 70** ***Especifique las pre y post condiciones de los métodos que implemente.*** ***Toda función y estructura utilizada debe especificarla e implementarla.*** ***Si utiliza funciones auxiliares, éstas deberán ser implementadas, aclarando a qué clase pertenece.*** ***Serán tenidos en cuenta ejercicios o partes de ellos completas y que el estilo y metodología de desarrollo se ajuste al curso.*** **IMPORTANTE** - Duración **2 horas** - **NO** se puede consultar material				

Ejercicio 1 *(40 puntos)*

a) (10 puntos) Defina las estructuras necesarias para modelar una lista doblemente encadenada de enteros(solo las estructuras)

```
public interface IListadoble {
   public boolean esVacia();
   public void agregarInicio(int dato);
   public void agregarFinal(int dato)
   public void borrarInicio();
   public void mostrar();
   public int cantElementos();
   public void vaciar();
}

public class Listadoble implements IListadoble {
   nodoD primero;
   nodoD ultimo;
   int cantnodoDs;

   public Listadoble() {
      this.primero = null;
      this.ultimo = null;
      this.cantnodoDs = 0;
   }
```

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

```
  public nodoD getPrimero() {
     return primero;    }

  public void setPrimero(nodoD primero) {
     this.primero = primero;    }

  public nodoD getUltimo() {
     return ultimo;    }

  public void setUltimo(nodoD ultimo) {
     this.ultimo = ultimo;    }

  public int getCantnodoDs() {
     return cantnodoDs;    }

  public void setCantnodoDs(int cantnodoDs) {
     this.cantnodoDs = cantnodoDs;    }
}
```

b) (25 puntos) Implemente un método que, dado **dos listas ordenadas simplemente encadenadas** de enteros, intercale sus elementos en una **nueva lista doblemente encadenada** de enteros, también **ordenada.**

```
public static ListaD intercalarOrdenado (ListaS l1, ListaS l2){

   Listadoble l3= new Listadoble();
   Nodo aux1 =l1.primero;
   Nodo aux2 = l2.primero;

   while (aux1!=null && aux2!=null){
      if (aux1.getDato()< aux2.getDato()){
         l3.agregarFinal(aux1.getDato());
         aux1=aux1.getSiguiente();
      }else{
          l3.agregarFinal(aux2.getDato());
         aux2=aux2.getSiguiente();
      }
   }

   while (aux1!=null ){
       l3.agregarFinal(aux1.getDato());
       aux1=aux1.getSiguiente();
   }
```

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

```
 while (aux2!=null ){
        l3.agregarFinal(aux2.getDato());
       aux2=aux2.getSiguiente();
 }

 return l3;

}
```

c) (5 puntos) ¿Cómo implementaría una lista simplemente encadenada si tuviese que contener un método que retorne el largo de la lista de la manera más eficiente posible

Se define un atributo **int cantnodoDs** y los métodos de incremento y decremento correspondientes para sumar y restar a la variable cada vez que se agregan o eliminan nodos. Luego solo se retorna el valor de ese atributo para saber la cantidad de nodos.

Facultad de Ingeniería
Bernard Wand-Polak
Cuareim 1451
11.100 Montevideo, Uruguay
Tel 902 15 05 Fax 908 13 70
www.ort.edu.uy

Ejercicio 2 (15 puntos)

Realizar un algoritmo recursivo que, dado un vector de enteros retorne la suma de todos aquellos valores que son múltiplos de dos.

```
public int  sumaMultiplos (int lista[], int pos){

    int ret = 0 ;

    if(pos < lista.length){
        if(lista[pos]%2==0 ){
            ret = lista[pos] + sumaMultiplos (lista, pos +1);
        }
        else{
            ret = sumaMultiplos (lista, pos +1);

        }
    }

    return ret;
}
```

Invocación: sumaMultiplos(unaLista, 0);

Ejercicio 3 (45 puntos)

Dada una Matriz M de dimensión mxm y el un vector V de dimensión m, como muestra el ejemplo:

a) (15 puntos)

```
public static int buscarv(int [ ][ ] M, int[ ] V ){
    int colEncontrada = -1;
    boolean encontrada = false;
    int largoCol = M[0].length;
    int largoVect = V.length;

    for(int col = 0; col < largoCol && !encontrada; col ++){
        boolean todoOK = true;
        for(int i=0; i< largoVect; i++){
            if(M[i][col] != V[i]){
                todoOK = false
            }
        }
        If(todoOK){
            colEncontrada = col;
            encontrada = true;
        }
```

```
        }

        return colEncontrada;
    }
```

b) (15 puntos)

Escribir un algoritmo recursivo (sumadiagonal) que sume los elementos de la diagonal principal de una matriz cuadrada.

```
// matriz cuadrada, asumimos que comenzamos e 0,0
public static int sumadiagonalM (int f, int c, int[ ][ ] M ) {
if (f==M.length-1 && c==M[0].length-1)
        return M[f][c];
else
        return M[f][c] +sumadiagonalM (f+1, c+1, M ) ;
}
```

c) (15 puntos)

Hacer un algoritmo que intercambie la primera columna con la última columna de una matriz cuadrada

```
public static void intecambiocol (int[ ][ ] M){

        int largoFilas=M.length;
        int ultimaCol = M[0].length - 1

         for (int fila=0;fila<largoFilas; fila++){
                int aux=M[fila][0];
                M[fila][0]=M[fila][ ultimaCol];
                M[fila][ultimaCol]=aux;
         }
}
```