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 AED1 **(solución)**	**GRUPO**	TODOS	**FECHA**	16/05/2025
MATERIA	Algoritmos				
CARRERA	Analista Programador / Analista en Tecnologías de la Información				
CONDICIONES	**- Puntos: 100** **- Duración: 3 horas** **- Sin material**				

Ejercicio 1 (25 ptos)

Dado una matriz cuadrada de enteros:

1	4	2	3
2	4	7	4
5	1	1	5
3	2	1	7

a) Escriba un algoritmo que retorne la suma de los elementos de la diagonal secundaria.
Firma: **public static int sumaDiagonalSecundaria(int[][] mat) (5 ptos)**

```
public static int sumaDiagonalSecundaria(int[][] mat) {
    int suma = 0;
    for (int i = 0; i < mat.length; i++) {
        suma += mat[i][mat.length - 1 - i];
    }
    return suma;
}
```

b) Escriba un algoritmo que retorne un boolean, indicando si existe en la matriz dos columnas consecutivas que sumen los mismo.
Firma: **public boolean sumanIgual(int[][] mat) (10 ptos)**
Nota: en el ejemplo dado, la columna de índice 1 y 2 suman los mismo y son consecutivas (11)

```
public static boolean sumanIgual(int[][] mat) {
    int columnas = mat[0].length;
    for (int j = 0; j < columnas - 1; j++) {
        int sumaCol1 = 0, sumaCol2 = 0;
        for (int i = 0; i < mat.length; i++) {
            sumaCol1 += mat[i][j];
            sumaCol2 += mat[i][j + 1];
        }
        if (sumaCol1 == sumaCol2){
                return true;
          }
        }
        return false;
    }
```

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

c) Escriba un método recursivo que retorne la cantidad de números pares presentes en una columna específica.
Firma: **public static int contarParesColumna(int[][] mat, int col) (10 ptos)**
Nota: es posible cambiar la firma del método si lo desea

```
public static int contarParesColumna(int[][] mat, int col) {
      return contarParesColumnaAux(mat, col, 0);
   }

   private static int contarParesColumnaAux(int[][] mat, int col, int fila) {
      if (fila == mat.length){
        return 0;
      }
      else{
            int actual = 0;
            if(mat[fila][col]%2==0){
                  actual = mat[fila][col];
            }
            return actual + contarParesColumnaAux(mat, col, fila + 1);


       }
   }
```

Ejercicio 2 (20 ptos)

Dado un vector de enteros desordenado y teniendo a disposición los métodos posMinVec(int vec[], int desde, int hasta) y posMaxVec(int vec[], int desde, int hasta) vistos en el curso, que retornan el índice en donde se encuentra el menor y mayor elemento del vector entre las posiciones desde y hasta respectivamente, implemente un método que, utilizando los métodos anteriores, permita ordenar un vector en forma ascendente.

Firma: **public static void ordenarAsc (int[ ] vec)**

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 static void ordenarAscConMinYMax(int[] vec) {
   int inicio = 0;
   int fin = vec.length - 1;

   while (inicio < fin) {
      int posMin = posMinVec(vec, inicio, fin);
      swap(vec, inicio, posMin);

      if (posMin == fin) {
         posMin = inicio;
      }

      int posMax = posMaxVec(vec, inicio + 1, fin);
      swap(vec, fin, posMax);

      inicio++;
      fin--;
   }
}
```

Ejercicio 3 (55 ptos)

a) Implemente una operación agregarFinal(int dato), que agrega un elemento al final de la lista.
Firma: **public void agregarFinal (int dato) (10 ptos)**

```
public void agregarFinal(int dato) {
        Nodo nuevo = new Nodo(dato);
        if (inicio == null) {
            inicio = nuevo;
            fin = nuevo;
        } else {
            fin.setSig(nuevo);
            fin = nuevo;
        }
        cantidad++;
    }
```

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

b) Implemente una operación de instancia **eliminarIndice** que elimina el elemento de la lista que se encuentra en el índice indicado (el primer elemento se encuentra en el índice 0). Definir las pre y post condiciones.
Firma: **public void eliminarIndice (int indice) (15 ptos)**

```
// pre: 0 <= indice < cantidad
    // post: el nodo en esa posición es eliminado
    public void eliminarIndice(int indice) {
        if (indice == 0) {
            inicio = inicio.getSig();
            if (inicio == null){
                fin = null;
            }
        } else {
            Nodo actual = inicio;
            for (int i = 0; i < indice - 1; i++){
                actual = actual.getSig();
            }
            Nodo eliminar = actual.getSig();
            actual.setSig(eliminar.getSig());
            if (eliminar == fin){
                fin = actual;
            }
        }
        cantidad--;
    }
```

c) Implemente la operación **sumarDesde** de forma recursiva, que reciba un índice desde el cual comenzar a sumar los valores de la lista (el primer elemento se encuentra en el índice 0)
Pre: desde < cantidad de elementos
Firma: **public int sumarDesde(int desde) (15 ptos)**

```
c) Sumar desde índice (recursivo)
public int sumarDesde(int desde) {
    return sumarDesdeAux(inicio, desde, 0);
}
private int sumarDesdeAux(Nodo nodo, int desde, int actual) {
    if (nodo == null){
        return 0;
    }
    if (actual >= desde){
        return nodo.getDato() + sumarDesdeAux(nodo.getSig(), desde, actual+1);
    }
    else{
        return sumarDesdeAux(nodo.getSig(), desde, actual+1);
    }
}
```

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

d) Implemente la operación insertarOrdenado, que inserte un dato manteniendo la lista ordenada en forma ascendente. Se cuenta con los métodos disponibles agregarInicio(int dato) y agregarFinal(int dato). Se tomará en cuenta la eficiencia.
Firma: **public void insertarOrdenado(int dato) (15 ptos)**

```
// d) Insertar ordenado
    public void insertarOrdenado(int dato) {
        if (inicio == null || dato < inicio.getDato()) {
            agregarInicio(dato);
        } else if (dato >= fin.getDato()) {
            agregarFinal(dato);
        } else {
            Nodo nuevo = new Nodo(dato);
            Nodo actual = inicio;
            while (actual.getSig().getDato() < dato) {
                actual = actual.getSig();
            }
            nuevo.setSig(actual.getSig());
            actual.setSig(nuevo);
            cantidad++;
        }
    }
```

Notas:

- **Para todas las operaciones solicitadas se dispone de gets y sets**
- **Se pueden utilizar funciones o métodos auxiliares, pero se deben implementar.**
- **Indicar claramente que parte se está resolviendo.**
- **Escribir con letra legible ya que se considerará durante la corrección.**