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 | **GRUPO** | TODOS | **FECHA** | 20/09/2024 |
| **MATERIA** | Algoritmos 1 | | | | |
| **CARRERA** | Analista Programador / Analista en Tecnologías de la Información | | | | |
| **CONDICIONES** | **- Puntos: 100**<br>**- Duración: 3 horas**<br>**- Sin material** | | | | |

## Ejercicio 1 (25 ptos)

Dado una matriz de enteros, no repetidos:

| | | | |
|---|---|---|---|
| 1 | 6 | 7 | 31 |
| 10 | 4 | 11 | 40 |
| 3 | 23 | 5 | 43 |
| 51 | 2 | 8 | 9 |

a) Escriba un algoritmo que retorne el mínimo de los elementos de la matriz, entre dos columnas dadas. **(5 ptos)**

//pre: col1 <= col2
Firma: **public static int minimoEntreCol (int[ ] mat, int col1, int col2)**
Ej: para col1: 1 y col2:2, el resultado debería ser: 2

```
public static int minimoEntreCol(int[][] mat, int col1, int col2) {
    int min = Integer.MAX_VALUE;

    for (int i = 0; i < mat.length; i++) {
        // Iteramos entre las columnas col1 y col2 (ambas incluidas)
        for (int j = col1; j <= col2; j++) {
            if (mat[i][j] < min) {
                min = mat[i][j];
            }
        }
    }
    return min;
}
```

**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) Escriba un algoritmo que retorne el índice de la fila que tenga el valor más grande **(10 ptos)**

Ej: para la matriz dispuesta, debería retornar 3 (el 51 es el valor más grande)

```
public static int indiceFilaMax(int[][] mat) {
    int max = Integer.MIN_VALUE;
    int filaMax = -1;

    // Recorremos todas las filas y columnas de la matriz
    for (int i = 0; i < mat.length; i++) {
        for (int j = 0; j < mat[i].length; j++) {
            if (mat[i][j] > max) {
                max = mat[i][j];
                filaMax = i;
            }
        }
    }
    return filaMax;
}
```

c) Escriba un método recursivo que muestre los valores de una columna dada, comenzando por el valor de la fila 0. Realice el diagrama de llamadas para la matriz dada y la columna:2 **(10 ptos)**

Ej: para la columna: 2 debería mostrar 7 11 5 8

```
public static void mostrarColumna(int[][] mat, int col, int fila) {
    // Caso base: si la fila supera el tamaño de la matriz
    if (fila != mat.length) {
            System.out.print(mat[fila][col] + " ");
            // Llamada recursiva a la siguiente fila
            mostrarColumna(mat, col, fila + 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

## Ejercicio 2 (20 ptos)

Dado dos vectores de enteros ordenados en forma ascendente, implemente un método que retorne un nuevo vector, con los números de ambos vectores dispuestos en forma ordenada, en forma ascendente. Se valorará que el método sea eficiente. Escriba la pre y post condiciones del método.

Firma: **public static int[] entrelazarOrdenado (int[ ] vec1, int[ ] vec2)**

Pre: vectores ordenados en forma ascendente, de cualquier largo.
Los dos vectores de entrada (vec1 y vec2) están ordenados en forma ascendente.
Post:
El vector resultante contiene todos los elementos de vec1 y vec2, ordenados en forma ascendente, sin modificar los valores originales.

```
public static int[] entrelazarOrdenado(int[] vec1, int[] vec2) {
    int n1 = vec1.length;
    int n2 = vec2.length;
    int[] resultado = new int[n1 + n2];

    int i = 0, j = 0, k = 0;

    // Fusionar los dos vectores mientras queden elementos en ambos
    while (i < n1 && j < n2) {
        if (vec1[i] <= vec2[j]) {
            resultado[k] = vec1[i];
            i++;
        } else {
            resultado[k] = vec2[j];
            j++;
        }
         K++;
    }

    // Si aún quedan elementos en vec1
    while (i < n1) {
        resultado[k] = vec1[i];
        k++;
        i++;
    }

    // Si aún quedan elementos en vec2
    while (j < n2) {
        resultado[k] = vec2[j];
        k++;
        j++;
    }

    return resultado;
}
```

**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 3 (55 ptos)

Se ha implementado una Lista simplemente enlazada que cuenta con un puntero al inicio y un entero para almacenar la cantidad de elementos.

```
public class Lista {
      private Nodo inicio;
      private int cantidad;
      //…….
}
```

```
public class Nodo {
      private int dato;
      private Nodo sig;

//Métodos de acceso y modificación disponibles
}
```

Implemente las siguientes operaciones en el TAD Lista:

a) Implemente la operación de instancia eliminar al final, que elimine el último elemento de la lista, retornando su valor **(10 ptos)**

Firma: **`public int eliminarFinal()`**

```
public int eliminarFinal() {
   if (inicio != null) {
        Nodo actual = inicio;

        // Si la lista tiene un solo elemento
        if (inicio.getSig() == null) {
                int dato = inicio.getDato();
                inicio = null;
                 cantidad--;
                 return dato;
         }
        else{
                 // Recorremos hasta el penúltimo nodo
                 while (actual.getSig().getSig() != null) {
                          actual = actual.getSig();
                 }

                int dato = actual.getSig().getDato();
                actual.setSig(null);
                cantidad--;
                 return dato;
        }
}
```

**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 la operación de instancia sumaPares (de forma recursiva), que retorne la suma de todos los elementos pares de la lista. **(20 ptos)**

Firma: **`public int sumaPares()`**

Ej: para la lista 10-3-5-76-11-2 debería retornar: 88

```
public int sumaPares() {
   return sumaParesAux(inicio);
}

// Método auxiliar recursivo
private int sumaParesAux(Nodo actual) {
   if (actual == null) {
      return 0; // Caso base:
   }

   Int suma = 0;
   // Si el dato es par
   If(actual.getDato() % 2 == 0) {
         return actual.getDato()  + sumaParesAux(actual.getSig());
   }
   else{
         return sumaParesAux(actual.getSig());
  }
}
```

c) Implemente la operación de instancia pilaDeMayores que recibe un número entero y retorna una pila con los elementos de la lista, cuyo valor sea mayor al número indicado. El primer número de la lista que cumpla la condición debe estar situado al tope de la lista. Los nodos de la lista y pila no deben compartir espacios en memoria (si quito un elemento de una estructura no debe afectar a la otra). Se dispone de las operaciones de pila vistas en el curso (apilar, desapilar, top) **(15 ptos)**

Firma: **`public Pila pilaDeMayores (int numero)`**

```
public Pila pilaDeMayores(int numero) {
   Pila pila = new Pila();
   Pila pRet  =  new Pila();
   Nodo actual = inicio;

   // Recorremos la lista
   while (actual != null) {
      if (actual.getDato() > numero) {
         pila.apilar(actual.getDato());  // Apilamos los elementos mayores que "numero"
      }
      actual = actual.getSig();
   }
   while(!pila.esVacia()){
         pRet.apilar(pila.desapilar()):
    }
   return pRet;
}
```

**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 nueva operación de instancia eliminarPrimerosN que recibe un entero N por parámetro y elimina los primeros N elementos de la lista. **(10 ptos)**

//pre: n <= cantidad
Firma: **`public void eliminarPrimerosN(int n)`**

```
public void eliminarPrimerosN(int n) {
    if (n <= cantidad) {

        for (int i = 0; i < n; i++) {
                if (inicio != null) {
                        inicio = inicio.getSig();
                        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.**