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** | SOLUCIÓN DE EXAMEN AED1 | **GRUPO** | TODOS | **FECHA** | 31/07/2024 |
| **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 cuadrada de entero:

| | | | |
|---|---|---|---|
| 1 | 4 | 2 | 3 |
| 2 | 4 | 7 | 4 |
| 5 | 1 | 5 | 5 |
| 3 | 2 | 5 | 7 |

a) Escriba un algoritmo que retorne el máximo de los elementos de la matriz. **(5 ptos)**

Firma: **public static int obtenerMaximo (int[ ] mat)**

```
public static int obtenerMaximo (int[][] matriz) {

      int maximo = matriz[0][0];

      for (int i = 0; i < matriz.length; i++) {
         for (int j = 0; j < matriz[i].length; j++) {
            if (matriz[i][j] > maximo) {
               maximo = matriz[i][j];
            }
         }
      }

      return maximo;
   }
}
```

b) Escriba un algoritmo que retorne el índice de la columna que tenga la mayor suma. En caso de haber más de una, retornar la de índice mayor. **(10 ptos)**

Ej: para la matriz dispuesta, la columna con la mayor suma es la de índice 2 y 3 (ambas suman 19). En este caso, se debería retornar 3.

```
public static int encontrarColumnaMayorSuma(int[][] matriz) {

      int columnas = matriz[0].length;
      int filas = matriz.length;

      int maxSuma = Integer.MIN_VALUE;
      int indiceMaxColumna = -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

```
        for (int j = 0; j < columnas; j++) {
            int sumaColumna = 0;

            for (int i = 0; i < filas; i++) {
                sumaColumna += matriz[i][j];
            }

            if (sumaColumna >= maxSuma) {
                maxSuma = sumaColumna;
                indiceMaxColumna = j;
            }
        }

        return indiceMaxColumna;
    }
}
```

c) Escriba un método recursivo que retorne la suma de la diagonal principal de la matriz. **(10 ptos)**

```
    System.out.println("La suma de la diagonal principal es: " + sumaDiagonal);

    public static int sumaDiagonalRecursiva(int[][] matriz, int indice) {

        // Caso base: Si el índice ha alcanzado el tamaño de la matriz, se retorna 0
        if (indice == matriz.length) {
            return 0;
        }

        // Caso recursivo: Sumar el elemento en la diagonal principal y llamar recursivamente para el siguiente
        return matriz[indice][indice] + sumaDiagonalRecursiva(matriz, indice + 1);
    }
```

## Ejercicio 2 (20 ptos)

Dado un vector de enteros ordenado en forma ascendente y un número entero M, implemente un método de búsqueda que indique si se encuentra dicho número en el vector. Se valorará que el método sea eficiente. Escriba la pre y post condiciones del método.

Firma: **public static boolean buscarM (int[ ] vec, int M)**

```
public static boolean buscarM (int[] vector, int M) {
        int inicio = 0;
        int fin = vector.length - 1;

        while (inicio <= fin) {
            // Calcular el punto medio del intervalo
            int medio = (inicio + fin) / 2;

            if (vector[medio] == M) {
                return true; // Número encontrado
            } else if (vector[medio] > M) {
```

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

```
            fin = medio - 1;
        } else {
            inicio = medio + 1;
        }
    }

    return false;
  }
}
```

## Ejercicio 3 (55 ptos)

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

```
public class Lista {
      private Nodo inicio;
      private Nodo fin;
      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 agregarInicio que agregue un elemento al inicio retornando la cantidad de elementos actualizada. **(10 ptos)**

Firma: **`public int agregarInicio(int dato)`**

```
public int agregarInicio(int dato) {

    Nodo nuevo = new Nodo(dato);

    if (estaVacia()) {
        inicio = nuevo;
        fin = nuevo;
    } else {
        nuevo.setSig(inicio);
        inicio = nuevo;
    }

    // Incrementar la cantidad de elementos en la lista
    cantidad++;

    return cantidad;
}

  // Método para verificar si la lista está vacía
  public boolean estaVacia() {
      return inicio == null;
  }
}
```

**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 mostrar (de forma recursiva) que muestre por consola los primeros n elementos de la lista. Escribir el diagrama de llamadas para una lista con que tenga los siguientes elementos agregados. Lista: 4 – 3 – 2 – 1 **(20 ptos)**

//pre: n <= cantidad

Firma: **`public void mostrar(int n)`**

```
public void mostrar(int n) {
    mostrarRecursivo(inicio, n);
}

// Método auxiliar recursivo
private void mostrarRecursivo(Nodo nodo, int n) {

    // Caso base: Si n es 0 o el nodo es null, se detiene la recursión
    if (n == 0 || nodo == null) {
        return;
    }

    // Imprimir el dato del nodo actual
    System.out.println(nodo.getDato());
    mostrarRecursivo(nodo.getSig(), n - 1);
}
}
```

c) Implemente la operación de instancia obtenerInvertida que retorne una nueva lista con los elementos de forma invertida. Las listas no deben compartir espacios en memoria (si quito un elemento de una no debe afectar a la otra) **(15 ptos)**

Firma: **`public Lista obtenerInvertida ( )`**

```
public Lista obtenerInvertida() {

        Lista listaInvertida = new Lista();
        Nodo actual = inicio;

        // Recorrer la lista original y agregar cada elemento al inicio de la nueva lista
        while (actual != null) {
            listaInvertida.agregarInicio(actual.getDato());   //Implementado en 3a
            actual = actual.getSig();
        }

        return listaInvertida;
}
```

**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 concatenar que recibe una lista por parámetro y agregue los elementos al final. Las listas no deben compartir espacios en memoria (si quito un elemento de una no debe afectar a la otra) **(10 ptos)**

Firma: **`public void concatenar(Lista lista)`**

```
public void concatenar(Lista lista) {
    Nodo actual = lista.inicio;

    while (actual != null) {
        this.agregarAlFinal(actual.getDato());
        actual = actual.getSig();
    }
}


// Método para agregar un elemento al final de la lista
public void agregarAlFinal(int dato) {
    Nodo nuevo = new Nodo(dato);

    if (estaVacia()) {
        inicio = nuevo;
        fin = nuevo;
    } else {
        fin.setSig(nuevo);
        fin = 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.**