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** | | **FECHA** | Mayo 2022 |
| **MATERIA** | Algoritmos 1 | | | | |
| **CARRERA** | Analista Programador / Analista en Tecnologías de la Información | | | | |
| **CONDICIONES** | **- Puntaje: 0/100**<br>**- Duración: 2 horas**<br>**- Sin material** | | | | |
| **Nombre** | **Nro estudiante** | | **Nota** | | |

## Ejercicio 1 (20 ptos )

Sean disponibles los métodos minPosVec(int[] vec, int desde, int hasta) y maxPosVec(int[] vec, int desde, int hasta) que retornan respectivamente la posición del mínimo y máximo elemento del vector entre dos posiciones (inclusive) dadas:

a) Implementar un algoritmo que reciba un vector de enteros desordenado y lo ordene, utilizando los métodos anteriormente mencionados. (15 ptos)

```
public void ordenarPosMinVec(int [] vec){

   int desde = 0;
   int hasta = vec.length -1;

   while(desde <= hasta){

      int posMin =  posminvec(vec,desde,hasta);
      int aux = vec[desde];
      vec[desde] = vec[posMin];
      vec[posMin] = aux;

      int posMax = posmaxvec(vec,desde,hasta);
      aux = vec[hasta];
      vec[hasta] = vec[posMax];
      vec[posMax] = aux;

      desde ++;
      hasta --;
   }

}
```

b) Detallar pre y post condiciones. (5 ptos)

Pre: el vector esta desordenado. Desde >=0 y < largo de vector. Hasta >=0 y < largo de vector. Desde <= Hasta
Post: Ordena el vector en forma ascendente

Firma a utilizar: **public void ordenarPorMinMax(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

## Ejercicio 2 (20 ptos)

Dado el siguiente vector:

```
int v[] = {78, 4, 3, 20, 39, 32};
```

a) Indique a que método de ordenamiento corresponde la siguiente secuencia. (5 ptos)

[4, 78, 3, 20, 39, 32]
[3, 4, 78, 20, 39, 32]
[3, 4, 20, 78, 39, 32]
[3, 4, 20, 39, 78, 32]
[3, 4, 20, 32, 39, 78]

Insert Sort

b) Implemente el método de ordenamiento correspondiente. (15 ptos)

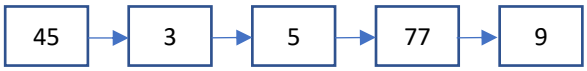
```
public void ordenar(int[] v){

    int largo = v.length;

    for(int i=1; i<largo;i++){

        int insert = v[i];
        int j=i-1;

        while(j>=0 && insert<v[j]){
            v[j+1] = v[j];
            j--;
        }

        v[j+1] = insert;
    }

}
```

**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 (20 ptos)

Dado una lista simplemente encadenada, implementar un método recursivo que permita indicar si se encuentra un elemento dado, dentro de una cantidad tope máxima de elementos indicada desde su inicio.

Ej: Si se buscara el número 5 con un tope máximo de 4 en la lista:




retornaría verdadero ya que el número 5 se encuentra en el lugar 3, siendo el tope máximo 4. Si se buscara el número 9 retornaría falso, ya que se encuentra en el lugar 5 (lugar superior al tope). Si se buscara el 8 retornaría falso dado que no está el número.

Firma a utilizar: **boolean busqueda(Nodo lista, int númeroBuscado, int tope)**

```
public boolean busqueda(Nodo lista, int numeroBuscado, int tope){

    boolean esta = false;

    if(tope > 0){

       if(lista != null){

          if(lista.getDato() == numeroBuscado){
             esta = true;
          }
          else{
               esta = busqueda(lista.getSiguiente(), numeroBuscado, tope - 1);
          }
       }
    }

    return esta;

}
```

**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 4 (20 ptos)

Escribir una función que permita eliminar de una pila todos los elementos mayores a un valor indicado:

Firma a utilizar: **boolean eliminarMayores(pila P, int valor)**

```
public boolean eliminarMayores(Pila p, int valor){

    Lista l = new Lista();
    int cantidad = p.getCantElementos();

    while(p != null){

        if(p.cima().getDato() <= valor){
            l.agregarInicio(p.cima().getDato());
            cantidad ++;
        }
           p.desapilar();
    }

    Nodo n = l.getInicio();

    while(n != null){
        p.apilar(n.getDato());
        n = n.getSiguiente();
    }

    return cantidad != p.getCantElementos();

}
```

**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 5 (20 ptos)**

Implementar un algoritmo que retorne una nueva matriz - de igual cantidad de filas, pero una columna más que la original - con la sumatoria de cada fila de la matriz original:

Firma a utilizar: **int [][] mat sumaFilas (int largFfila, int largoCol, int[][] m)**

**Ejemplo:**

**Original**

| | | |
|---|---|---|
| 1 | 4 | 3 |
| 1 | 3 | 5 |
| 6 | 9 | 1 |

**Retorno**

| | | | |
|---|---|---|---|
| 1 | 4 | 3 | **8** |
| 1 | 3 | 5 | **9** |
| 6 | 9 | 1 | **16** |

```
public int[][] sumaFilas (int largFfila, int largoCol, int[][] m){

   int[][] matRet = new int[largFfila][largoCol];

   for(int i=0; i<m.length; i++){
      int suma = 0;
      for(int j=0; j<m[i].length; j++){
         matRet[i][j] = m[i][j];
         suma += m[i][j];
      }
      matRet[i][matRet[0].length-1] = suma;
   }

   return matRet;

}
```