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 EXAMEN AED1	**GRUPO**	TODOS	**FECHA**	10/05/2024
MATERIA	Algoritmos 1				
CARRERA	Analista Programador / Analista en Tecnologías de la Información				
CONDICIONES	**- Puntos: 100** **- Duración: 3 horas** **- Sin material**				

Ejercicio 1 (10 ptos)

Escriba un algoritmo que, dado una matriz de enteros de largo mxn indique (mediante un valor booleano) si las diagonales de la matriz suman lo mismo, como se muestra en el ejemplo.

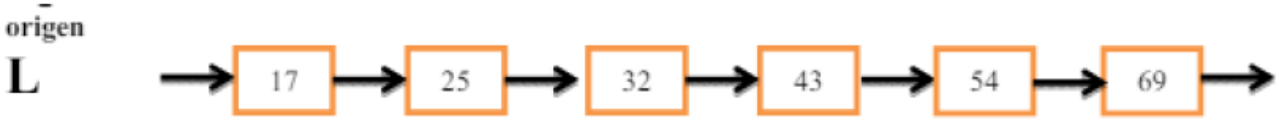
28	3	62	18
15	39	47	5
8	57	45	6
25	16	71	35

```
public boolean sumanIguales(int mat[][]){

        boolean iguales = true;

        int diagoPrincipal = 0;
        int diagoSecundaria = 0;

        for(int i=0; i<matl.length; i++){
                diagoPrincipal += mat[i][i];
                diagoSecundaria += mat[i][ mat.length – 1 - i];
        }

        If(diagoPrincipal != diagoSecundaria ){
                Iguales = false;
        }

        return Iguales;
}
```

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)

Dada la siguiente lista simplemente enlazada y ordenada:



a) Implemente un algoritmo recursivo que muestre los últimos n elementos impares de la lista
Firma sugerida: mostrarUltimosNImpares (Lista L, int n)
Ejemplo: para la lista anterior y un n = 2, el resultado sería: 69 43

b) Realice el diagrama de llamadas para el ejemplo anterior

```
public void mostrarUltimosNImpares (Lista L, int n){

        int totalNodosImpares = contarTotalNodosImpares(L.getInicio())
        mostrarUltimosNImpares (L.getInicio(), totalNodosImpares - n);

}


public void mostrarUltimosNImpares (Nodo n, int pos){

      if(n != null){
            if(n.getDato() % 2 != 0){
                  if(pos <= 0){
                        mostrarUltimosNImpares (n.getSiguiente(), pos-1);
                        System.out.prinln(n.getDato());
                  }
                  else{
                        mostrarUltimosNImpares (n.getSiguiente(), pos-1);

                  }
            }
            else{
                  mostrarUltimosNImpares (n.getSiguiente(), pos);
            }

      }
}
```

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

Diagrama de llamadas

mostrarUltimosNImpares (Nodo(17), 2)
mostrarUltimosNImpares (Nodo(25), 1)
mostrarUltimosNImpares (Nodo(32), 0)
mostrarUltimosNImpares (Nodo(43), 0)
mostrarUltimosNImpares (Nodo(54), -1)
mostrarUltimosNImpares (Nodo(69), -1)
mostrarUltimosNImpares (Nodo(null), -2)

FIN
SOUT(69)
FIN
FIN
SOUT(43)
FIN
FIN
FIN
FIN

Ejercicio 3 (50 ptos)

a) Realizar un algoritmo, que reciba 2 listas simplemente encadenadas ordenadas en forma ascendente y retorne una lista ordenada L3 ascendente que contenga los elementos de L1 y L2 **(15puntos)**:

L1: 1 – 5 – 16 – 34
L2: 3 – 4 – 35 – 37 – 44
L3: 1 – 3 – 4 – 5 – 16 – 34 – 35 – 37 – 44

```
public Lista encadenarListas(Lista l1, Lista l2){
        Lista l3 = new Lista();
        Nodo nodoL1 = l1.getInicio();
        Nodo nodo L2 = l2.getInicio();

         while(nodoL1 != null || nodoL2 != null){

                if(nodoL1 != null && nodoL2 != null){
                        if(nodoL1.getDato() <= nodoL2.getDato()){
                                l3.agregarFinal(nodoL1.getDato());
                                nodoL1 = nodoL1.getSiguiente();
                        }
                        else{
                                l3.agregarFinal(nodoL2.getDato());
                                nodoL2 = nodoL2.getSiguiente();

                        }

                }else{
```

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

```
                        if(nodoL1 != null){
                                l3.agregarFinal(nodoL1.getDato());
                                nodoL1 = nodoL1.getSiguiente();
                        }
                        else{
                                l3.agregarFinal(nodoL2.getDato());
                                nodoL2 = nodoL2.getSiguiente();

                        }
        }
        return l3;
}
```

Nota: se debe implementar método agregarFinal

b) Realizar un algoritmo de búsqueda eficiente que, dada la lista retornada en el punto 1 (enlazada y ordenada en forma ascendente), busque un número en la lista y retorne un valor booleano indicando si lo encuentra o no. **(20 puntos)**:

```
public boolean buscar(Lista l3, int numero){

        boolean esta = false;
        Nodo aux = l3.getInicio();
        while(aux !=null && !esta){

                if(aux.getDato() == numero){
                        esta = true ;
                }
                aux = aux.getSiguiente() ;
        }

        Return esta ;
}
```

c) Realizar un algoritmo – eficiente - que reciba las listas L1 y L2 de la parte a) y retorne la siguiente Pila apilada de menor a mayor según figura: (**15 puntos**)

44
37
35
34
16
5
4
3
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

Se disponen de las estructuras vistas en clase (Pila) con los métodos básicos vistos de dichas estructuras (esVacia(),top(), desapilar(), apila(elemento),etc.). En caso de necesitar alguna otra estructura (ej. Cola), se deberán implementar sus operaciones.

```
public Pila encadenarListasEnPila (Lista l1, Lista l2){
        Pila p = new Pila();
        Nodo nodoL1 = l1.getInicio();
        Nodo nodo L2 = l2.getInicio();

         while(nodoL1 != null || nodoL2 != null){

                    if(nodoL1 != null && nodoL2 != null){
                              if(nodoL1.getDato() <= nodoL2.getDato()){
                                        p.push(nodoL1.getDato());
                                        nodoL1 = nodoL1.getSiguiente();
                              }
                              else{
                                        p.push (nodoL2.getDato());
                                        nodoL2 = nodoL2.getSiguiente();

                              }

                    }else{
                             if(nodoL1 != null){
                                        p.push (nodoL1.getDato());
                                        nodoL1 = nodoL1.getSiguiente();
                              }
                              else{
                                        p.push (nodoL2.getDato());
                                        nodoL2 = nodoL2.getSiguiente();

                              }
        }
        return p;
}
```

Ejercicio 4 (20 ptos)

Dado una implementación de TAD Lista doblemente encadenada - desordenada - de nodos, con puntero al primero (inicio) y puntero al último (fin), realizar - dentro del TAD - una operación que me permita eliminar el último elemento de la lista. Se deberán contemplar todos los casos de borde e implementar todas las operaciones - adicionales - del TAD que sean necesarias la operación requerida.

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 void eliminarFinal(){

        if(this.getInicio() != null){

                if(this.getInicio() == this.getFin()){
                        inicio = null;
                        fin = null;
                }
                else{
                        Nodo aBorrar = fin;
                        fin = aBorrar.getAnterior();
                        aBorrar.setAnterior(null)
                        fin.setSiguiente(null);
                }
        }


}
```