APELLIDOS:	***NOMBRE:***
GRADO:	***FIRMA:***
Duración: 2h 30'	

Ejercicio 1	Ejercicio 2	Ejercicio 3	Ejercicio 4	Ejercicio 5	TOTAL

NOTA: Los ejercicios del 2 al 5 se codificarán en hojas aparte.

<u>Ejercicio 1</u>: [10 puntos] Este ejercicio se responderá en estas hojas de examen.
Definir las constantes, tipos de datos y flujos de E/S apropiados para:

a) Almacenar en memoria principal la información de los alumnos de Programación Visual, un máximo de 120 alumnos. De cada alumno se quiere almacenar: código de alumno, nombre, DNI y nota. **[2,5 Puntos]**

```
final int MAXALUM = 120;
class Alumno implements Serializable {
    int codalumno; // Código de alumno
    String nombre; //Nombre alumno
    String dni; // DNI alumno
    double nota; // Nota en Programación Visual
}
Alumno [] alumnos = new Alumno[MAXALUM];
```

b) Almacenar en memoria principal la información de los pasajeros de un avión Boeing 737 (este avión tiene 33 filas y 6 asientos por fila). De cada asiento se quiere saber: si está ocupado, nombre, DNI y el tipo de pasajero: a (standard), b (infantil) y c (especial). **[2,5 Puntos]**

```
final int MAXFILA = 33;
final int MAXCOL = 6;
class Avion implements Serializable {
    boolean ocupado; // Indica si el asiento está ocupado o no
    String nombre; //Nombre pasajero
    String dni; // DNI pasajero
    char tipo; // Tipo de pasajero
}
Avion [][] pasajeros = new Avion[MAXFILA][MAXCOL];
```

c) Crear un flujo de datos para escribir en un fichero de texto una línea para cada alumno almacenado en el apartado 1 a). **[2,5 Puntos]**

```
PrintStream flujo = new PrintStream(new FileOutputStream("misalumnos.txt"));
O bien
BufferedWriter out = new BufferedWriter(new FileWriter("misalumnos.txt"));
```

APELLIDOS:	***NOMBRE:***
GRADO:	***FIRMA:***
Duración: 2h 30'	

d) Crear un flujo de datos para leer los datos de los alumnos del apartado 1-a) que se han almacenado en un archivo binario llamado "alumnos.dat". **[2,5 Puntos]**

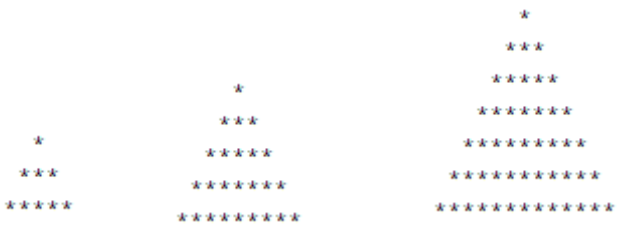
```
try{
  objectInputStream in = new ObjectInputStream(new FileInputStream("alumnos.dat"));
```

O bien

```
FileInputStream in = new FileInputStream("alumnos.dat");
ObjectInputStream inso = new ObjectInputStream(in);
```

Ejercicio 2: **[20 puntos]**

Escribir un método que dibuje por pantalla un árbol de Navidad. El método recibirá un número entero positivo que será el número de pisos y mostrará por pantalla un árbol de Navidad como se muestra en las figuras. **No se pueden usar arrays de ningún tipo**.
Por ejemplo si el método recibe el número 3 mostrará el árbol de la izquierda, si recibe un 5 el del centro y si recibe un 7 el de la derecha.



```
public static void ArboldeNavidad(int pisos) {
 // No se pueden usar matrices
    pisos = pisos * 2 - 1;

    for (int i = 0; i < pisos; i += 2) {
        for (int j = 0; j < pisos - i; j += 2) {
            System.out.print(" ");
        }
        for (int k = 0; k <= i; k++) {
            System.out.print("*");
        }
        System.out.println();
    }

} // Fin ArboldeNavidad
```

Ejercicio 3: **[10 puntos]**

Escribir un **método recursivo** que reciba un número entero positivo cuyos dígitos solo pueden ser ceros o unos. El método devolverá la posición del primer uno, empezando a contar desde la derecha (es decir, desde el menos significativo).
Por ejemplo, si n=1100100, el resultado es 2. Para n=1101, el resultado es 0. El número n siempre contendrá al menos un uno (al ser estrictamente mayor que 0).

APELLIDOS:	***NOMBRE:***
GRADO:	***FIRMA:***
Duración: 2h 30'	

```
public static int posicion_primer_1(int n) {
    if (n % 10 == 1) {
        return 0;
    } else {
        return 1 + posicion_primer_1(n / 10);
    }
}
```

Ejercicio 4: [45 puntos]

Se quiere realizar un programa que realice distintas operaciones sobre arrays uni y bidimensionales, para ello se piden los siguientes métodos:

a) [15 puntos] Escribir un método llamado **SinRepetidos** que reciba un array de una sola dimensión de números enteros y una variable llamada tope que marcará las posiciones del array que están ocupadas. El método modificará el array eliminando los elementos repetidos y devolverá el nuevo tope. El tamaño del array puede variar en las distintas llamadas al método.

Por ejemplo si el método recibe un array con estos valores:

3 3 9 5 4 5 tope=5

Devolverá el array y el nuevo valor de tope así:

3 9 5 4 tope=3

```
public static int SinRepetidos(int[] tabla, int tope) {
    //Recibe el array y el tope y deja el array
    //sin repetidos y devuelve el nuevo tope
    int[] unicos = new int[tabla.length];
    int ntope = 0;
    for (int i = 0; i <= tope; i++) {
        boolean existe = false;
        for (int j = 0; j < ntope; j++) {
            if (unicos[j] == tabla[i]) {
                existe = true;
            }
        }
        if (!existe) {
            unicos[ntope] = tabla[i];
            ntope++;
        }
    }
    tope = ntope - 1;
    for (int i = 0; i <= tope; i++) {
        tabla[i] = unicos[i];
    }
    return tope;
}
```

APELLIDOS:	***NOMBRE:***
GRADO:	***FIRMA:***
Duración: 2h 30'	

b) [15 puntos] Escribir un método llamado **InsertarOrdenado** que inserte un elemento en un array, en una posición que se le envía. Para ello el método recibirá un array de una sola dimensión de números enteros (la longitud puede variar en cada llamada al método), el elemento a insertar, la posición donde insertar y el tope. El método devolverá el nuevo tope. **No se puede usar otro array auxiliar.**
Por ejemplo si el método recibe un array con estos valores:
3 3 9 5 4 5 elemento a insertar=7 posición=3 tope=5
Dejará el array con estos valores y devolverá el tope:
3 3 9 7 5 4 5 y el tope= 6

```
public static int InsertaOrdenado(int[] tabla, int elem, int pos, int tope) {
    // recibe el array, la posición donde insertar, el elemento a insertar
    //y el tope NO SE PUEDE USAR OTRO ARRAY AUXILIAR
    int aux;
    if (tope < tabla.length - 1) {
        if (pos < tope) {
            aux = tabla[pos];//salvo el elemento en la posción a insertar
            tabla[pos] = elem;
            for (int i = tope; i > pos; i--) {
                tabla[i + 1] = tabla[i];
            }
            tabla[pos + 1] = aux;
            tope++;
        } else if (pos == tope) {
            aux = tabla[pos];
            tabla[pos] = elem;
            tabla[pos + 1] = aux;
            tope++;
        } else {
            System.out.print("No se ha podido insertar en el array, ");
            System.out.println("la posición está fuera de los límites del array");
        }
    } else {
        System.out.println("La tabla está llena no puedes insertar más");
    }
    return tope;
}
```

c) **[15 puntos]** Escribir un método que se llame **MatrizTraspuesta** que reciba una matriz genere su traspuesta en una nueva matriz y la muestre por pantalla. La matriz que se reciba puede variar en tamaño en cada llamada al método. La matriz no tiene por qué ser cuadrada.
Por ejemplo: si el método recibe la matriz de la izquierda mostrará la de la derecha

APELLIDOS:	***NOMBRE:***
GRADO:	***FIRMA:***
Duración: 2h 30'	

```
0 0 0 0 0
2 2 2 2 2
4 4 4 4 4
6 6 6 6 6
8 8 8 8 8
```

```
0 2 4 6 8
0 2 4 6 8
0 2 4 6 8
0 2 4 6 8
0 2 4 6 8
```

```
public static void MatrizTraspuesta(int[][] mat) {
    //Método que recibe una matriz genera su traspuesta
    //en una nueva matriz y la muestra por pantalla
    int fila = mat.length;
    int col = mat[0].length;
    int[][] matT = new int[fila][col];
    // para calcular su traspuesta
    for (int i = 0; i < mat.length; i++) {
        for (int j = 0; j < mat[i].length; j++) {
            matT[j][i] = mat[i][j];
        }
    }
    // para mostrar la transpuesta
    System.out.println();
    for (int i = 0; i < mat.length; i++) {
        for (int j = 0; j < mat[i].length; j++) {
            System.out.print(matT[i][j] + " ");
        }
        System.out.println();
    }
}
```

Ejercicio 5: **[15 puntos]**

a) Escribe las instrucciones necesarias para poder **leer de un fichero binario llamado "alumnos.dat"** que contiene almacenado el array de los alumnos del apartado 1-a). Además muestra por pantalla una línea por cada alumno (código, nombre, dni y nota)**. [7,5 puntos]**

APELLIDOS:	***NOMBRE:***
GRADO:	***FIRMA:***
Duración: 2h 30'	

```
try {
    FileInputStream in = new FileInputStream("alumnos.dat");
    ObjectInputStream inso = new ObjectInputStream(in);
    Alumno alum;
    alum = (Alumno) inso.readObject();
    while (alum != null) {
        System.out.println(alum.codalumno + alum.nombre + alum.dni + alum.nota);
        alum = (Alumno) inso.readObject();
    }
    inso.close();
} catch (IOException exc) {
    System.out.println("Otro tipo de excepción");
} catch (ClassNotFoundException ex) {
    System.out.println("Error al convertir");
}
```

b) Haz un método **EscribirFicheroTexto** que reciba el array de alumnos creado en el apartado 1-a) y escriba en un fichero de texto (Flujo creado en el apartado 1-c) una línea para cada alumno almacenado en el array. **[7,5 puntos]**

```
public static void escribirFicheroTexto (Alumno[]alum) throws IOException{
    try {
        PrintStream flujo = new PrintStream(new FileOutputStream("misalumnos.txt"));
        for (int i=0; i < alum.length; i++ ){
        flujo.println( alum[i].codalumno+"\t" +alum[i].nombre+"\t"+alum[i].dni+"\t"+ alum [i].nota);
        }
        flujo.close();
    }catch (FileNotFoundException e)
        {System.out.println("Fichero no emcontrado");}
     }// fin escribirFicheroTexto

        //o bien
public static void escribirFicheroTexto1 (Alumno[]alum) throws IOException{
  try {
        BufferedWriter out = new BufferedWriter(new FileWriter("misalumnos.txt"));
        for (int i=0; i < alum.length; i++ ){
        out.write("codalum"+alum[i].codalumno+"nom:"+alum[i].nombre+"DNI"+alum[i].dni+"nota"+alum[i].nota);
        out.newLine();
        }
        out.close();
  }catch (EOFException eof)
        {System.out.println("Fichero no se pudo abrir");}
  }// fin escribirFicheroTexto
```