Sección 1. Indica (F) Falso o (V) verdadero para las siguientes afirmaciones.

1. ____ En el mejor caso, la búsqueda secuencial es un algoritmo $O(n)$.

2. ____ Cuando una clase es heredada, los métodos constructores de la clase base son heredados también.

3. ____ Un arreglo de variables de tipo entero (int) es una variable de tipo primitivo.

4. ____ Las excepciones lanzadas en código fuera de un bloque try-catch, detienen la ejecución del programa.

5. ____ Un algoritmo recursivo debe contar siempre con un caso base, y un caso recursivo más pequeño.

6. ____ Los métodos getter sirven para actualizar el contenido de una variable.

7. ____ Las variables estáticas también son llamadas variables de instancia.

8. ____ El método super( ) sirve para llamar el método constructor de una clase base.

9. ____ En el polimorfismo, un objeto puede comportarse como un objeto de su clase base.

10. ____ Sobrecargar un método es tener dos métodos con distintos nombres, pero con los mismos parámetros de entrada.

Sección 2. Selecciona la mejor opción.

1. Los siguientes son los conceptos básicos de la programación orientada a objetos, excepto:

a) Herencia
b) Polimorfismo
c) Interfaces gráficas
d) Encapsulación

2. Indica el resultado de ejecutar el siguiente código.

```
public static void main(String[] args) {
        int[] x = {7,3,6,1,1};
        updateArray(x);
        printArray(x);
}

public static void updateArray(int[] array) {
        array[3] = 10;
}

public static void printArray(int[] array) {
        for(int i = 0; i<array.length; i++) {
                System.out.print(array[i] + " ");
        }
}
```

a) 7 3 6 1 1
b) 7 3 10 1 1
c) 7 3 6 10 1
d) 7 3 10 1

3. Indica el resultado de ejecutar el siguiente código

```java
public static void main(String[] args) {
        double a = 17;
        double b = 0.5;

        int c = xOperation(a,b);
        System.out.println(c);
}

public static int xOperation(int a, int b){
        return a + b;
}

public static double xOperation(double a, int b){
        return a-b;
}

public static int xOperation(double a, double b){
        return (int)(a*b);
}

public static double xOperation(int a, double b){
        return (a-1)*(b+0.5);
}
```

a) 17.0
b) 8
c) 16.0
d) 16
e) 17
f) 8.0

4. Al analizar un algoritmo nos encontramos que su tasa de crecimiento se puede modelar a través de la siguiente ecuación:

$$O(n) = 3n^2 + 10n\log(n) + 1000$$

¿Cuál de las siguientes opciones representa su notación Big-O?

a) $O(n^2)$
b) $O(3n^2)$
c) $O(n\log(n))$
d) $O(1)$

```java
public class Bottle {

    private double capacity; //capacity in liters
    private String owner;
    private static int numberOfBottles;

    public Bottle() {
        this("", 0);
    }

    public Bottle(String owner, double capacity) {
        numberOfBottles++;
        this.owner = owner;
        this.capacity = capacity;
    }

    public void print() {
        System.out.println("Name: " + this.owner);
        System.out.println("Capacity: " + this.capacity + "L");
        System.out.println("# Bottles in the world: " + Bottle.numberOfBottles);
    }
}
```

5. **Elige la opción que mejor representa un setter para la variable `owner`:**

a)
```
public String setOwner() {
    return this.owner;
}
```

b)
```
public void setOwner(String owner) {
    owner = this.owner;
}
```

c)
```
public void setOwner(String owner) {
    this.owner = owner;
}
```

d)
```
public String setOwner(String owner) {
    this.owner = owner;
    return owner;
}
```

6. **Tomando como resultado la clase `Bottle`, y tu elección del método `setOwner`, cuál sería el resultado de ejecutar el siguiente código:**

```
public static void main(String[] args) {

  Bottle cokeBottle = new Bottle("Arturo Curry", 2.5);
  Bottle pepsiBottle = new Bottle("Francisco Castillo", 0.6);

  Bottle copyBottle;
  copyBottle = cokeBottle;
  copyBottle.setOwner("Bruno Díaz");

  copyBottle = pepsiBottle;
  copyBottle.print();
}
```

a) Name: Francisco Castillo
Capacity: 0.6L
Bottles in the world: 2

b) Name: Arturo Curry
Capacity: 2.5L
Bottles in the world: 1

c) Name: Francisco Castillo
Capacity: 0.6L
Bottles in the world: 1

d) Name: Bruno Díaz
Capacity: 0.6L
Bottles in the world: 1

Sección 3. Ordena ascendentemente la siguiente colección de datos utilizando el algoritmo Merge Sort. Muestra cada paso.

-5	10	17	-8	7	4	0	0

Sección 4. Utiliza el algoritmo de búsqueda binaria para encontrar el elemento 'H' en la siguiente colección de datos. Muestra cada paso.

A	B	C	D	E	F	G	H

Sección 5. Resuelve el siguiente caso.

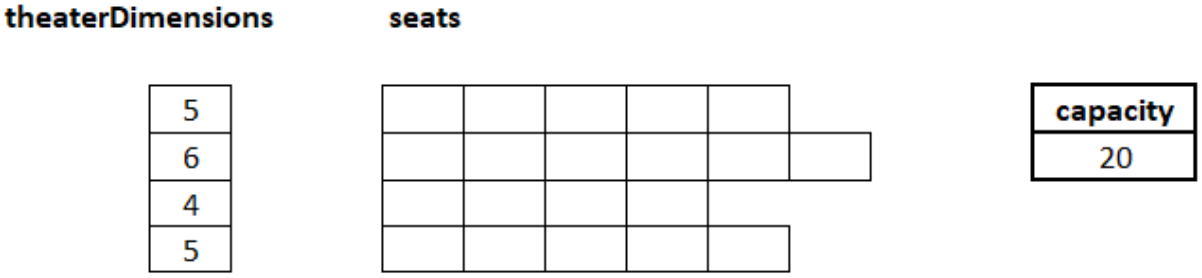
Cineplus, una nueva cadena de complejos de cine, ha decidido abrir su primera sucursal en la ciudad de Monterrey. Su concepto único está basado en ofrecer boletos de cine a un costo variable dependiendo de la disponibilidad de asientos en cada función.

Diseña y codifica una clase llamada `CinemaShow` que permita modelar una función de cine en un complejo de Cineplus. La clase deberá contar con los atributos definidos a continuación. Elige los tipos de datos que mejor se adecúan para representar la siguiente información. Utiliza las mejores prácticas y conceptos de la programación orientada a objetos. Considera si cada método debe ser estático o de instancia.

- Variable `movieName` que sirva para almacenar el nombre de la película que se va a proyectar.
- Variable `movieDate` que sirva para representar la fecha en la que se proyectará la película.
- Variable `movieTime` que sirva para representar la hora en la que se proyectará la película.
- Variable `capacity` que permita almacenar el aforo de la sala de cine (cantidad de personas que caben en la sala).
- Variable `soldTickets` que mantenga un registro actualizado de la cantidad de asientos vendidos de la función al momento.
- Arreglo bidimensional `seats` que sirva para llevar el control de los asientos vendidos.

Adicionalmente, la clase deberá contener las siguientes acciones:

- Método constructor que reciba e inicialice las variables: `movieName`, `movieDate` y `movieTime`.
- Métodos setter diferentes para las variables: `movieName`, `movieDate` y `movieTime`.
- Métodos getter diferentes para las variables: `capacity` y `soldTickets`.
- Método `initializeSeats` que reciba como parámetro de entrada un arreglo de enteros `theaterDimensions`, y que inicialice el arreglo `seats`. Adicionalmente, deberá calcular el aforo de la sala y almacenarlo en la variable de instancia `capacity`.
  El arreglo recibido `theaterDimensions` representará la cantidad de asientos en cada fila de la sala. Por ejemplo:



- Método `double assignPrice(int soldTickets, int totalTickets)` que calcule y retorne el precio de un boleto de acuerdo con la siguiente fórmula:

$$Precio = 50 + ((número\ de\ boleto)/(asientos\ totales)) \times 50$$

  Por ejemplo, el boleto #36 se vendería en:

$$Precio = 50 + (36/100) \times 50 = \$68.00$$

- Método `boolean sellSeat(int row, int column)` que reciba como parámetro de entrada dos enteros: `row` y `column`, calcule el precio del boleto (ver método `assignPrice`), y lo almacene en la fila y columna recibida. Posteriormente deberá actualizar la variable `soldTickets`.
- Sobrecarga el método `boolean sellSeat()` para que asigne automáticamente un espacio vacío en la sala.

- **Después de lanzar una aplicación web para vender boletos, se ha detectado que cuando dos clientes entran a comprar un boleto al mismo tiempo, el sistema les asigna precios idénticos. Si fuera a implementarse una lista de espera al comprar boletos para evitar este problema, ¿qué estructura de datos sería la más adecuada para hacerlo, y cómo se pudiera utilizar? Justifica tu respuesta. (5 puntos)**

- **El arreglo `seats` no debe ser retornado directamente por un método getter. ¿Cuál es el riesgo de implementar un getter para dicha variable y cómo pudiera mitigarse? Justifica tu respuesta. (5 puntos).**