Section 1. Select the right option.

1.

```
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

2.

```
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

```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);
    }
}
```

3. **Choose the best setter for the `owner` variable:**

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;
}
```

4. **What would be the result of running the code below:**

```
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();
}
```

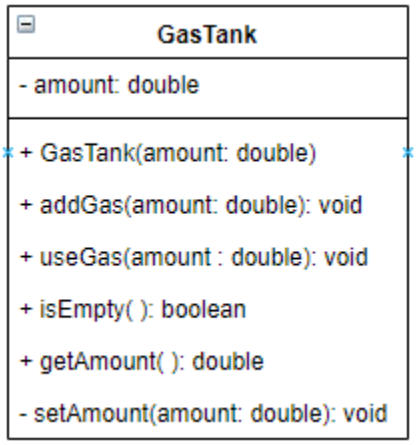
a) Name: Francisco Castillo
Capacity: 0.6L
Bottles in the world: 2

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

Section 2: Design a class called **`GasTank`** that represents a tank of gasoline:



- **Constructor:** Method to initialize the instance variable **amount**. Use the method **setAmount**.
- **addGas**: Increases the amount of gas by the amount received as a parameter. Validate that only positive values are processed.
- **useGas:** Reduces the amount of gas in the tank by the amount received as a parameter. Validate that only positive values are processed.
- **isEmpty:** Should return **true** when the amount of gas in the tank is smaller than 0.1. Otherwise, return **false**.
- **getAmount:** Getter for the **amount** variable.
- **setAmount:** Private method (should only be used by the Constructor) to set the value of **amount.**

Section 3.

Cineplus, a new movie theater complex, has decided to open its first movie theater in Monterrey. Its unique concept is to offer tickets with dynamic pricing depending on how many seats are available at purchase time.

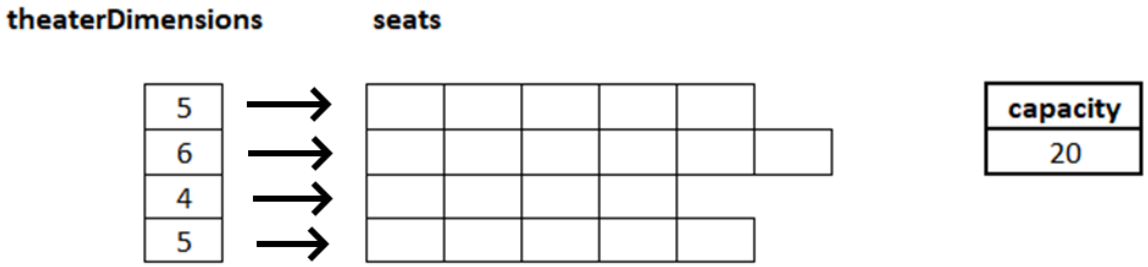
Code a class `CinemaShow` that allows the user to model a show at Cineplus. The class should have the attributes defined next. Use the best practices of object-oriented programming. Consider the proper visibility (public, private) and if every method or variable should be instance or static.

- Variable `movieName` to represent the movie that is being screened.
- Variable `movieDate` to represent the date when the movie will be screened.
- Variable `movieTime` to represent the time when the movie will be screened.
- Variable `capacity` to store the capacity of the movie theater (how many people the show can store).
- Variable `soldTickets` to keep track of how many tickets have been sold at any given point.
- Multidimensional array `seats` to keep track of the tickets sold.

Additionally, include the following methods:

- Constructor method to initialize: `movieName`, `movieDate`, and `movieTime`.
- Setter methods for `movieName`, `movieDate`, and `movieTime`.
- Getter methods for `capacity` and `soldTickets`.
- Method `initializeSeats` that received the theater dimensions as an array and initialize the `seats` multidimensional arrays. Additionally, once `seats` is initialized, `capacity` should be updated.

  The `theaterDimensions` array represents the number of seats for every row of the theater. For example:



- Method `assignPrice(int soldTickets, int totalTickets)` that calculates and returns the price for the next ticket sold using the following formula:

$$price = 50 + ((tickets\ sold)/(capacity)) \times 50$$

  For example, the ticket #16 in a theater with a capacity of 20 seats would be sold:

$$Precio = 50 + (16/20) \times 50 = \$90.00$$

- Method `sellSeat(int row, int column)` that receives two integer inputs: `row` and `column` and validates the seat is empty. If it isn't, it should immediately return false. If it is empty, it should calculate the price for the ticket using the method `assignPrice`, then store it on the row/column combination received as input and proceed to update the `soldTickets` counter.