1. This question simulates birds or possibly a bear eating at a bird feeder. The following `Feeder` class contains information about how much food is in the bird feeder and simulates how much food is eaten. You will write two methods of the `Feeder` class.

```
public class Feeder
{
   /**
    *  The amount of food, in grams, currently in the bird feeder; initialized in the constructor and
    *  always greater than or equal to zero
    */
   private int currentFood;

   /**
    *  Simulates one day with numBirds birds or possibly a bear at the bird feeder,
    *  as described in part (a)
    *  Precondition: numBirds > 0
    */
   public void simulateOneDay(int numBirds)
   {  /* to be implemented in part (a) */  }

   /**
    *  Returns the number of days birds or a bear found food to eat at the feeder in this simulation,
    *  as described in part (b)
    *  Preconditions: numBirds > 0, numDays > 0
    */
   public int simulateManyDays(int numBirds, int numDays)
   {  /* to be implemented in part (b) */  }

   // There may be instance variables, constructors, or methods that are not shown.
}
```

GO ON TO THE NEXT PAGE.

(a) Write the `simulateOneDay` method, which simulates `numBirds` birds or possibly a bear at the feeder for one day. The method determines the amount of food taken from the feeder on this day and updates the `currentFood` instance variable. The simulation accounts for normal conditions, which occur 95% of the time, and abnormal conditions, which occur 5% of the time.

Under normal conditions, the simulation assumes that on any given day, only birds visit the feeder and that each bird at the feeder consumes the same amount of food. This standard amount consumed is between 10 and 50 grams of food, inclusive, in 1-gram increments. That is, on any given day, each bird might eat 10, 11, . . . , 49, or 50 grams of food. The amount of food eaten by each bird on a given day is randomly generated and each integer from 10 to 50, inclusive, has an equal chance of being chosen.

For example, a run of the simulation might predict that for a certain day under normal conditions, each bird coming to the feeder will eat 11 grams of food. If 10 birds come to the feeder on that day, then a total of 110 grams of food will be consumed.

If the simulated food consumed is greater than the amount of food in the feeder, the birds empty the feeder and the amount of food in the feeder at the end of the day is zero.

Under abnormal conditions, a bear empties the feeder and the amount of food in the feeder at the end of the day is zero.

The following examples show possible results of three calls to `simulateOneDay`.

- Example 1: If the feeder initially contains 500 grams of food, the call `simulateOneDay(12)` could result in 12 birds eating 20 grams of food each, leaving 260 grams of food in the feeder.
- Example 2: If the feeder initially contains 1,000 grams of food, the call `simulateOneDay(22)` could result in a bear eating all the food, leaving 0 grams of food in the feeder.
- Example 3: If the feeder initially contains 100 grams of food, the call `simulateOneDay(5)` could result in 5 birds attempting to eat 30 grams of food each. Since the feeder initially contains less than 150 grams of food, the feeder is emptied, leaving 0 grams of food in the feeder.

GO ON TO THE NEXT PAGE.

Complete the `simulateOneDay` method.

```
/**
 *   Simulates one day with numBirds birds or possibly a bear at the bird feeder,
 *   as described in part (a)
 *   Precondition: numBirds > 0
 */
public void simulateOneDay(int numBirds)
```

Class information for this question

```
public class Feeder

private int currentFood

public void simulateOneDay(int numBirds)
public int simulateManyDays(int numBirds, int numDays)
```

GO ON TO THE NEXT PAGE.

(b) Write the `simulateManyDays` method. The method uses `simulateOneDay` to simulate `numBirds` birds or a bear coming to the feeder on at most `numDays` consecutive days. The simulation returns the number of days that birds or a bear found food at the feeder.

Consider the following examples.

Value of `currentFood` and Method Call	**Possible Outcomes and Resulting Return Value**

`currentFood: 2400`
 	
`simulateManyDays(10, 4)`
	Day 1: `simulateOneDay` leaves 2100 grams of food in the feeder.
	Day 2: `simulateOneDay` leaves 1650 grams of food in the feeder.
	Day 3: `simulateOneDay` leaves 1500 grams of food in the feeder.
	Day 4: `simulateOneDay` leaves 1260 grams of food in the feeder.
	
	The simulation returns `4` because, on all four days of the simulation, birds or a bear found food at the feeder. The instance variable `currentFood` has the value `1260`.

`currentFood: 250`
 	
`simulateManyDays(10, 5)`
	Day 1: `simulateOneDay` leaves 150 grams of food in the feeder.
	Day 2: `simulateOneDay` leaves 0 grams of food in the feeder.
	
	The simulation returns `2` because, on two of the five simulated days, birds or a bear found food at the feeder. The instance variable `currentFood` has the value `0`.

`currentFood: 0`
 	
`simulateManyDays(5, 10)`
	The simulation returns `0` because no food was found at the feeder on any day. The instance variable `currentFood` has the value `0`.

GO ON TO THE NEXT PAGE.

Complete the `simulateManyDays` method. Assume that `simulateOneDay` works as intended, regardless of what you wrote in part (a). You must use `simulateOneDay` appropriately in order to receive full credit.

```
/**
 *   Returns the number of days birds or a bear found food to eat at the feeder in this simulation,
 *   as described in part (b)
 *   Preconditions: numBirds > 0, numDays > 0
 */
public int simulateManyDays(int numBirds, int numDays)
```

Class information for this question

```
public class Feeder

private int currentFood

public void simulateOneDay(int numBirds)
public int simulateManyDays(int numBirds, int numDays)
```

GO ON TO THE NEXT PAGE.