Problem B. CandyBoxes

Problem Statement

Little Billy has several boxes full of candies in front of him. However he doesn't like that these boxes don't each contain the same number of candies, so he decides to eat some of the candies until they do.

Specifically, little Billy follows this algorithm:

Step 1: He calculates min, the minimum number of candies in a box, and max, the maximum number of candies in a box. If these numbers are equal the algorithm ends, otherwise continue with step 2.

Step 2: He arbitrarily chooses a box with max candies and eats min candies from it, then he returns to step 1.

You will be given a int[] **candies**. The ith element of **candies** is equal to the number of candies in the ith box. Return the number of candies remaining in each of the boxes after the algorithm described above is executed.

Definition

Class: CandyBoxes
Method: remainingCandies
Parameters: int[]
Returns: int
Method signature: int remainingCandies(int[] candies)
(be sure your method is public)

Constraints

- **candies** will contain between 1 and 50 elements, inclusive.

- Each element of **candies** will be between 1 and 1,000, inclusive.

Examples

0)

```
{7}
```

```
Returns: 7
```

There is only one box, so Billy doesn't eat any candies for now.

1)

```
{7, 21, 14}
```

```
Returns: 7
```

First, Billy eats 7 candies from the second box, leaving 14 candies in it. In the following two steps, he will eat 7 candies from the second and third boxes, leaving 7 candies in each.

2)

```
{3, 4, 5, 6, 7}
```

```
Returns: 1
```

3)

```
{366, 549, 915, 183, 549, 549, 183, 366, 915, 549, 915, 366}
```

```
Returns: 183
```