BJP5 Exercise 7.1: lastIndexOf

Show Header

Language/Type: Java arrays
Author: Marty Stepp (on 2019/09/19)

Write a method named `lastIndexOf` that accepts an array of integers and an integer value as its parameters and returns the last index at which the value occurs in the array. The method should return -1 if the value is not found. For example, in the list containing {74, 85, 102, 99, 101, 85, 56}, the last index of the value 85 is 5.

Type your solution here:

```
1  public int lastIndexOf(int array[], int x){
2      int place = -1;
3
4      for (int count = 0; count < array.length; count++){
5          if (array[count] == x){
6              place = count;
7          }
8      }
9      return place;
10 }
```

This is a **method problem.** Write a Java method as described. Do not write a complete program or class; just the method(s) above.

4 Indent

Submit

Sound F/X
Highlighting

You passed 8 of 8 tests.

Go to the next problem: range

test #1:	`lastIndexOf({74, 85, 102, 99, 101, 85, 56}, 85)`
return:	5
result:	pass
test #2:	`lastIndexOf({74, 85, 102, 99, 101, 85, 56}, 58)`
return:	-1
result:	pass

BJP5 Exercise 7.2: range

Show Header

Language/Type: Java arrays
Author: Lisa Fiedler (on 2019/09/19)

Write a static method named `range` that takes an array of integers as a parameter and returns the range of values contained in the array. The range of an array is defined to be one more than the difference between its largest and smallest element. For example, if the largest element in the array is 15 and the smallest is 4, the range is 12. If the largest and smallest values are the same, the range is 1.

The following table shows some calls to your method and their results (the largest and smallest values are underlined):

Call	Value Returned
`int[] a1 = {8, 3, 5, 7, 2, 4};`	`range(a1) returns 7`
`int[] a2 = {15, 22, 8, 19, 31};`	`range(a2) returns 24`
`int[] a3 = {3, 10000000, 5, -29, 4};`	`range(a3) returns 10000030`
`int[] a4 = {100, 5};`	`range(a4) returns 96`
`int[] a5 = {32};`	`range(a5) returns 1`

You may assume that the array contains at least one element (that its length is at least 1). You should not make any assumptions about the values of the particular elements in the array; they could be extremely large, very small, etc. You should not modify the contents of the array.

Type your solution here:

```
1  public static int range(int array[]){
2      int smallest = array[0];
3      int largest = array[0];
4
5      for (int count = 0; count < array.length; count++){
6          if (array[count] < smallest)
7              smallest = array[count];
8
9          if (array[count] > largest)
10             largest = array[count];
11     }
12     return (largest - smallest) + 1;
13
14
15 }
```

This is a **method problem.** Write a Java method as described. Do not write a complete program or class; just the method(s) above.

Indent: 4
Sound F/X
Highlighting

Submit

You passed 5 of 5 tests.

Go to the next problem: countInRange

test #1: `range({8, 3, 5, 7, 2, 4})`
return: 7
result: pass

BJP4 Self-Check 7.4: oddsArray

Language/Type: Java arrays
Author: Marty Stepp (on 2016/09/08)

Write code that creates an array named `odds` and stores all odd numbers between -6 and 38 into it using a `for` loop. Make the array's size exactly large enough to store the numbers.

Type your solution here:

```
1 int[] odds = new int[22];
2
3 for (int i = 0; i < odds.length; i++){
4         odds[i] = i * 2 - 5 ;
5 }
```

This problem asks for **bare code.** Submit a fragment of Java code as described. Do not write any class or method heading around your code; just write the lines of code that will produce the result described.

Indent: 4
Sound F/X
Highlighting

Submit

You passed 1 of 1 tests.

Go to the next problem: numbersArray

test #1: `test1`
console output: `odds = [-5, -3, -1, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 3`
result: pass