This second lab gets you to work array algorithms. Implement your answers in the `lab2.ipynb` notebook file. For each of the questions, **write down and run at least 3 tests**, trying to capture corner cases (e.g. empty arrays). In the notebook file, we have already provided you with tests for the first two Questions.

Marks (max 5): Questions 1-2: 2 | Questions 1-4: 3.5 | Questions 5-7: 0.5 each

Question 1

Write a Python function:

```
def multAll(A, k)
```

which takes an array of integers and an integer as inputs and multiplies every integer in the array by the integer argument.
For example, if it takes the array [5,12,31,7,25] and the integer 10, it will change the array to [50,120,310,70,250].

Question 2

Write a Python function:

```
def multAll2(A, k)
```

which performs the same task as `question1`, but does it by creating and returning a new array with the multiplied values rather than changing the array passed to it.
For example, if it takes the array [5,12,31,7,25] as one argument and the integer 10 as the other, it will return the array [50,120,310,70,250] but the initial array will remain [5,12,31,7,25].

Question 3

Write a Python function:

```
def biggestIn(A)
```

which takes an array of integers and returns the biggest integer in the array.
For example, if it takes the array [5,12,31,7,25] it will return 31.

Question 4

Write a Python function:

```
def biggestInPos(A)
```

which takes an array of integers and returns the index of the biggest integer in the array.
For example, if it takes the array [5,12,31,7,25] it will return 2.

Question 5

Write a Python function:

```
def occurInBoth(A, B)
```

which takes two arrays of integers and returns the number of integers that occur in both arrays. For example, if it takes the arrays [5,12,31,7,25] and [4,12,8,7,42,31] it will return 3, because exactly three integers (7, 12 and 31) occur in both the arrays.

Hint: you will probably find useful to use the function `isIn` that we saw in the lecture Exercises of week 1, which checks whether a given element appears inside a given array. Also, you may assume that each of the arrays has no repeating elements.

Bonus (no marks): consider the case where the arrays may have repeating elements.

Question 6

Consider how your answer to Question 4 would work if the largest integer occurred more than once in the array. For example, if the array were [5,12,31,7,25,31,18,7,31] would it return 2 or 5 or 8?
Also consider how would it work if it took an empty array as its argument.

Then write a function which takes an array of integers and returns an array containing (all) the positions of the largest integer in the array. For example, if it takes the array [5,12,31,7,25], it will return [2], and if it takes the array [5,12,31,7,25,31,18,7,31] it will return [2,5,8]. If it takes [] as it argument, it should return [].

Hint: the array of indices that you need to return can be of arbitrary size – e.g. in the examples above it is [2], or [2,5,8] or [] – so you might want to build it on the fly. You can use the function `append` that we saw in the lecture Exercises of week 1, which takes an array and an integer and creates new array extending the old one with that integer.

Question 7

Consider how a different algorithm could be used for Question 5 if it were known that both arrays passed to the method were sorted (in increasing order – e.g. [1,5,5,23,56] is sorted, whereas [1,23,5,56,5] is not). The algorithm would be more *efficient*.
Try to write a method which implements this more efficient algorithm.