Partitioning Two Lists

Write a function, `partition`, that takes the following input:

- list1 - a list of integers
- list2 - a list of integers
- func - a function that takes a single integer as an argument and returns a real number

and returns a list of lists where each sublist contains values found in both of the two lists for which the function `func` returns the same value.

This point is important: a given value of `func` needs to be generated by values in *both* lists (see example 4 below for a case where this doesn't happen).

You should pass in the function in whatever form your language of choice allows. So, for example, in Python you can simply pass in a function parameter. In Objective C, on the other hand, you would likely use a block.

Your solution should have time complexity of O(N).

Examples

Example 1

For example, suppose that our function, `func` is:

```
def func(x): return x % 2
```

If `list1 = [1,2,3,4,5]` and `list2 = [6,7,8,9,10]` then `partition(list1, list2, func)` would return `[[1,3,5,7,9], [2,4,6,8,10]]`.

In this case, `func` returns its argument mod 2. Thus, `partition` returns a list containing 2 sublists: one of odd numbers (for which `func` returns 1) and the other of even numbers for which `func` returns 2).

Example 2

If we had:

```
def func(x): return x*x
```

then `partition([-1,2,3],[1,2,-3,4],func)` would return `[[-1,1],[2,2],[3,-3]]`.

Example 3

Finally, note that if:

```
def func(x): return x
```

then `partition` effectively computes the intersection of two lists.

Example 4

For example, suppose that our function, `func` is:

```
def func(x): return x % 2
```

If `list1 = [1,3]` and `list2 = [2]` then `partition(list1, list2, func)` would return `[]` because although 1 and 3 are both the same mod 2, there is nothing in `list2` that yields 1 when modded with 2.