A Full-fill-ing Problem

Two requirements:

- Your solution to this problem must use a recursive function, using the definition presented in the problem.
- Your solution must not use global variables.

Solutions that violate either of these requirements will receive no credit.

Suppose we have a 0-indexed array a of integers like this:

values	7	2	2	1	1	1	1	2	3	5
index	0	1	2	3	4	5	6	7	8	9

We can define a `fill` operation that, starting at an index i, replaces the value at i, and any contiguous positions containing that same value, with a new value x. For example, performing the `fill` operation on the above array with $i = 4$ and $x = 0$ would result in the following array:

values	7	2	2	0	0	0	0	2	3	5
index	0	1	2	3	4	5	6	7	8	9

And performing the `fill` operation on the original array with $i = 1$ and $x = 0$ would result in the following array:

values	7	0	0	1	1	1	1	2	3	5
index	0	1	2	3	4	5	6	7	8	9

Another example would be taking the following array and performing the `fill` operation with $i = 7$ and $x = 1$.

values	1	0	0	1	1	0	0	0	1	1
index	0	1	2	3	4	5	6	7	8	9

Such operation on the original array would output the following:

values	1	0	0	1	1	1	1	1	1	1
index	0	1	2	3	4	5	6	7	8	9

Notice how, in this case, the element at index 7 (with value 2) is unaffected because the `fill` operation only affects positions *contiguous* to position i that have the same value contained in position i.

The `fill` operation is commonly used in two-dimensional arrays representing images, to replace some contiguous region of one color with a different color. For simplicity, we will only deal with one-dimensional arrays and, in particular, we will implement the `fill` operation recursively as follow: Given a 0-indexed array a with N elements, an index i, an original value v and a new value x:

- If $i < 0$ or $i \geqslant N$, do nothing and return (the index is out of bounds).
- If $a[i] \neq v$, do nothing and return (we've encountered a value that is not the one we want to replace).
- If $a[i] = x$, do nothing and return (we've already changed the value at this index).
- Otherwise, set $a[i]$ to x and call `fill` recursively two times: one with $i - 1$ and another with $i + 1$ (a, v, and x remain the same in the recursive calls)

Note that `fill` does not return anything. Its purpose is to modify an array in-place. Also, take into account that the initial call to the `fill` function should pass $a[i]$ as the value for v.

Reminders:

- Your solution to this problem must use a recursive function. The recursive function must be based on the definition presented above.
- Your solution must not use global variables.

A solution that violate either of these requirements, even if the testing system judges it correctly, will receive no credit.

Your task is to complete the function `fill`, which takes an array (a list in Python) of integer values, an integer index `i`, and an integer replacement value `x`. This function should update values as described above.

Here is the skeleton code for this task:

```
'''
Distribution file of the fill problem.
'''

from typing import List

def fill(array: List[int], i: int, x: int) -> None:
    '''
    Performs a fill operation on the list, replacing all continuous elements
    with same value as list[i] with x.

    Note that this function works recursively.

    Arguments:
        array: a list of integers
        i: an integer
        x: an integer

    Returns:
        None
    '''
    # TODO: Implement this function.
    pass
```

When you take the placement exam, you will be expected to copy the skeleton code into a file and then complete the function. For the practice problems, we have provided a file named problem7.py that includes the above header for your convenience.