3. (5 points) Scanning

We all know the higher order functions map, filter, and reduce. Today we’re going to talk about their not-quite-so-famous fourth sibling, scan. Scan is like reduce, only instead of accumulating the result into a single value, scan returns a list that contains all the intermediate values in reducing the list.

Cross out lines from the implementation of the scan function below so that all doctests pass and the implementation contains as few lines of code as possible. You may want to look at the return statement first. Do not cross out any docstrings or doctests.

```python
def scan(f, lst, start):
    """Returns a list containing the intermediate values of reducing the list."

    >>> scan(add, [1, 2, 3, 4], 0)
    [1, 3, 6, 10]
    >>> scan(mul, [3, 2, 1, 0], 10)
    [30, 60, 60, 0]
    ""

    start = []
    start = 0

    accumulated = f(start)
    accumulated = start

    def closure(item):
        nonlocal accumulated

        nonlocal start

        accumulated = f(item)
        accumulated += f(item)
        accumulated = f(accumulated, item)
        accumulated += f(accumulated, item)

        return accumulated

        return start + accumulated

        return item + accumulated

        return list(map(f(lst)))

        return list(map(f, lst))

        return list(map(closure(lst)))

        return list(map(closure, lst))
```