Quiz 4 Free Response (70 pts)

Time Limit: **15 mins**

Name: _____ AndrewID: _____

Time limit: 15 mins

Free Response 1: increasingLengthKSublists (30 pts)

Write the function `increasingLengthKSublists(L, k)` that takes a list of integers `L` and an integer `k`, and returns a tuple containing all the sublists of `L` of length `k` that are strictly increasing from left to right. You may assume `k <= len(L)`.

For example, if `L = [5, 2, 4, 7, 8, 8, 1, 3, 4]`, and `k = 3`, then `increasingLengthKSublists(L, k)` should return `([2, 4, 7], [4, 7, 8], [1, 3, 4])`. All three sublists strictly increase from left to right. Note that `[7, 8, 8]` is not included in the result because it is not strictly increasing.

Note: You may only use material that was covered in units 1-4. Thus, do not use sets, dictionaries, OOP, or recursion.

Test cases:

```
def testIncreasingLengthKSublists():
    L = [5, 2, 4, 7, 8, 8, 1, 3, 4]
    res = ([2, 4, 7], [4, 7, 8], [1, 3, 4])
    assert( increasingLengthKSublists(L, 3) == res )

    assert( increasingLengthKSublists([-3, -1, -2, 0], 2) == ([-3, -1], [-2, 0]) )
    assert( increasingLengthKSublists([1, 2, 3, 4], 2) == ([1, 2], [2, 3], [3, 4]) )
    assert( increasingLengthKSublists([3, 1, 2], 1) == ([3], [1], [2]) )

    assert( increasingLengthKSublists([1, 2, 3, 4], 4) == ([1, 2, 3, 4],) )
    assert( increasingLengthKSublists([1, 1, 2], 2) == ([1, 2],) )

    assert( increasingLengthKSublists([1, 2, 4, 3], 4) == () )
    assert( increasingLengthKSublists([5, 4, 3, 2], 2) == () )
```

Write your answer below.

Free Response 2: makeEdits (40 pts)

Write the function `makeEdits(L, edits)` that takes a 1D list of integers `L` and a list of edits, and mutatingly applies those edits to `L`. The edits should be applied in the order they appear in the list. As with most mutating functions, the function should return `None`.

All edits will be of one of the following forms:

- `Add X to index i`
- `Subtract X from index i`
- `Insert X at index i`
- `Remove all multiples of X`

See the comments in the test cases for examples. You may assume that `i` will always be a valid index in the list.

Note: You may only use material that was covered in units 1-4. **You may use lists however you would like**, but do not use sets, dictionaries, OOP, or recursion.

Test cases:

```
def testMakeEdits():
    L = [5, 1, 3, 7]
    edits = [
        'Add 5 to index 1',             # [5, 6, 3, 7]
        'Subtract 10 from index 2',     # [5, 6, -7, 7]
        'Insert -3 at index 0',         # [-3, 5, 6, -7, 7]
        'Add 1 to index 3',             # [-3, 5, 6, -6, 7]
    ]
    makeEdits(L, edits)
    assert(L == [-3, 5, 6, -6, 7])

    L = [5, 1, 8, 0, 3, 15]
    edits = [
        'Subtract 3 from index 2',      # [5, 1, 5, 0, 3, 15]
        'Insert -4 at index 3',         # [5, 1, 5, -4, 0, 3, 15]
        'Add 17 to index 5',            # [5, 1, 5, -4, 0, 20, 15]
        'Remove all multiples of 5',    # [1, -4]
    ]
    makeEdits(L, edits)
    assert(L == [1, -4])

    L = [1, 3, 4, 6, 3, 8]
    edits = [
        'Insert 5 at index 3',          # [1, 3, 4, 5, 6, 3, 8]
        'Remove all multiples of 3',    # [1, 4, 5, 8]
        'Insert 7 at index 2',          # [1, 4, 7, 5, 8]
    ]
    makeEdits(L, edits)
    assert(L == [1, 4, 7, 5, 8])

    L = [2, 4, 6]
    assert(makeEdits(L, ['Remove all multiples of 2']) == None)
    assert(L == [])
```

Write your answer below.