Session 13: *Patterns for Sequences*

Your name:____ ***SOLUTION*** ______

1. Implement (here, on paper, in the supplied box) the following function, per its specification.

```
def list_of_numbers(n):
    """
    Returns the list    [1, 2, 3, 4, ... n]     where n is the
    given non-negative, integer argument.  For example:
      -- If the argument is 5, this function returns: [1, 2, 3, 4, 5]
      -- If the argument is 2, this function returns: [1, 2]
      -- If the argument is 0, this function returns: []  (the empty list)
    """

    result = []
    for k in range(n):
        result = result + [k + 1]  # or equivalently: result.append(k + 1)

    return result
```

2. Implement (here, on paper, in the supplied box) the following function, per its specification.

```
def string_of_numbers(n):
    """
    Returns the string    "12345678910111213 ..."     where the last number
    in the string is the given non-negative, integer argument n.  For example:
      -- If the argument is  6, this function returns: "123456"
      -- If the argument is 25, this function returns:
             "12345678910111213141516171819202122232425"
      -- If the argument is  0, this function returns: ""  (the empty string)
    """

    result = ""
    for k in range(n):
        result = result + str(k + 1)

    return result
```

3. Indicate the ***pattern*** that one would use to implement the following function.

```
def number_of_stutters(string):
    """
    Returns the number of "stutters" in the given string, where
    a "stutter" is a character repeated twice in a row.  For example:
      -- If the argument is "xhhbrrs",      this function returns: 2
      -- If the argument is "zzzz",         this function returns: 3
      -- If the argument is "xxx xxx xxxx", this function returns: 7
      -- If the argument is "xxxyyyxxx",    this function returns: 7
    """
```

FIND ***MAX/MIN*** [✓ ***TWO-PLACES-AT-ONCE***] ***PARALLEL SEQUENCES***
(circle/underline your choice)

4. Indicate the ***pattern*** that one would use to implement the following function.

```
def largest_number(sequence, m):
    """
    Returns the largest number in the first  m  numbers of the given
    sequence of numbers, where the positive integer  m  is the second
    argument.  For example, if sequence X is [7, 4, 15, 20, 13. 40, 10], then
      -- largest_number(X, 1)   returns 7
      -- largest_number(X, 2)   returns 7
      -- largest_number(X, 3)   returns 15
      -- largest_number(X, 4)   returns 20
      -- largest_number(X, 6)   returns 40
      -- largest_number(X, 7)   returns 40
    """
```

FIND [✓ ***MAX/MIN***] ***TWO-PLACES-AT-ONCE*** ***PARALLEL SEQUENCES***
(circle/underline your choice)

5. Indicate the ***pattern*** that one would use to implement the following function.

```
def index_of_first_negative(sequence):
    """
    Returns the index of the first negative number in the given sequence
    of numbers.  Returns -1 if the sequence contains no negative numbers.
    For example, if the argument is:
      -- [4, 30, -19, 8, -3, -50, 100], this function returns: 2
      -- [-8, 44, 33, -20, -1],         this function returns: 0
      -- [1, 29, 22, 8],                this function returns: -1
    """
```

[✓ ***FIND***] ***MAX/MIN*** ***TWO-PLACES-AT-ONCE*** ***PARALLEL SEQUENCES***
(circle/underline your choice)

6. Indicate the ***pattern*** that one would use to implement the following function.

```
def vector_sum(seq1, seq2):
    """
    Returns a list that is the item-by-item sum of the two given
    sequences of numbers, where the sequences are guaranteed to be the same
    length.  For example:
      -- vector_sum([6, 2, 5, -10, 20],
                    [3, 1, 3, 400, 10]
            returns
                    [9, 3, 8, 390, 30]
    """
```

FIND ***MAX/MIN*** ***TWO-PLACES-AT-ONCE*** [✓ ***PARALLEL SEQUENCES***]
(circle/underline your choice)