CIS2107_Lab 04: "Processing1DArrays"

Points: **100** points.

Objective:

- To design and implement functions to process 1D Arrays.

Instructions:

- Be sure to document your code (add comments on top of each function).
- In the comments add your name, date, course, homework number, and statement of problem.
- Once you are done, upload your final solution through Blackboard.
- No need for input validation.
- Implement a program called **`Arrays1DDemo.c`**

Steps:

- It is up to you to choose what the return type should be. Be creative!
- Functions should not have pointers implementations.
- **`Arrays1DDemo.c`** has the following functions

Part1 [100 points]: (1-Dimensional Array Functions)

1. **[20 points]** Implement a function called **`fillArray`** that fills a one-dimensional array with random integers. Integers are picked in the range LOW to HIGH (inclusive.)

   Here is a demo of filling an array of 40 elements with integers in the range 0 and 100.

```
 0   56   19   81   59   48   35   90   83   75
17   86   71   51   30    1    9   36   14   16
99   45   12    0    0   38   53   57   60   61
16   66   45   35    5   61   79   81   52   30
```

2. **[20 points]** Implement a function called **`findWithRange`** that locates the largest element in a range of the same array. The range consists of the array cells indexed between indices LOW and HIGH, inclusive.

   Here is a demo where LOW==10 and HIGH==19.

```
 0   56   19   81   59   48   35   90   83   75
17   86   71   51   30    1    9   36   14   16
99   45   12    0    0   38   53   57   60   61
16   66   45   35    5   61   79   81   52   30

Max = 86
```

3. **[20 points]** Implement function called **`reverseArray`** that reverses the order of the array elements.

   Here is a sample run:

```
Original:
         0    56    19    81    59    48    35    90    83    75
        17    86    71    51    30     1     9    36    14    16
        99    45    12     0     0    38    53    57    60    61
        16    66    45    35     5    61    79    81    52    30

Reversed:
        30    52    81    79    61     5    35    45    66    16
        61    60    57    53    38     0     0    12    45    99
        16    14    36     9     1    30    51    71    86    17
        75    83    90    35    48    59    81    19    56     0
```

4. **[20 points]** Implement a function that reverses the order of the array elements in a range between two indexes, LOW and HIGH. Decide what your function should do for indexes out of bounds, or if HIGH<LOW.

   Here is a demo:

```
         0    56    19    81    59    48    35    90    83    75
        17    86    71    51    30     1     9    36    14    16
        99    45    12     0     0    38    53    57    60    61
        16    66    45    35     5    61    79    81    52    30

    Reversing between 15 and 24

         0    56    19    81    59    48    35    90    83    75
        17    86    71    51    30     0     0    12    45    99
        16    14    36     9     1    38    53    57    60    61
        16    66    45    35     5    61    79    81    52    30
```

5. **[20 points]** Implement a function called **`findSequence`** that looks for Tom and Jerry in sequence among the array. Return the index of the first element Tom, or -1 if the sequence is not found.

   Here is a demo.

```
    Enter two numbers: 56 19
    sequence found at index 1

    Enter two numbers: 52 30
    sequence found at index 39

    Enter two numbers: 61 61
    sequence not found
```

Part2 [10 points]: (Testing inside `main`)

- Call all functions in part 1 in order to demonstrate successful run.
- Use blank lines to separate outputs and make then more readable.
- Be creative when displaying outputs.