Use the command `diary` to record your answers and submit them. Also submit code for the functions you write.

1. (25 points) Array construction and indexing. Let

   ```
   array1 = [ 10, 20, 30, 40, 50 ];
   array2 = [ 10, -20, 30, -40, 50 ];
   array3 = [ 50, 40, 30, 20, 10 ];
   ```

   (a) Use the double color operator to construct `array1`.

   (b) Use the double colon operator to get all the odd numbered elements of `array1`.

   (c) Use the double colon operator to get all the even numbered elements of `array1`.

   (d) Return the last element of `array1`.

   (e) Return elements 1, 2, and 4 of `array1`.

   (f) Construct `array2`. Use one statement to copy `array1` into the variable `array2` and another statement to negate the even elements of `array2`.

   (g) Construct `array3` from `array1` using the double colon operator.

2. (30 points) Array operations. Use the definitions of `array1`, `array2`, and `array3` from Problem 1.

   (a) Compute the elementwise sum of `array1` and `array2`. Compute the elementwise product of `array2` and `array3`.

   (b) Write a function `ElementwiseMax` that takes two arrays and returns a single array where each element in the resulting array is the maximum of the two elements in the corresponding positions of the input arrays. E.g., `ElementwiseMax([1, 2, 3],[1, -1, 5])` would return `[1, 2, 5]`. Do the following test cases

   ```
   >> ElementwiseMax(array1, array2);
   >> ElementwiseMax(array1, array3);
   >> ElementwiseMax(array2, array3);
   ```

   Note that this gives the same behavior as using the builtin Matlab function `max`. 

1
(c) An array can be said to be *monotonically increasing* if its elements are of non-decreasing value (i.e., Array(1) ≤ Array(2) ≤ ... ≤ Array(n)) and *monotonically decreasing* if its elements are of non-increasing value (i.e., Array(1) ≥ Array(2) ≥ ... ≥ Array(n)). The array is *monotonic* if it is either monotonically increasing or monotonically decreasing. Write a function `IsMonotonic` whose input is an array and whose output is true if the array is monotonic and false otherwise. You may find it useful to use the Matlab function `numel` which counts the number of elements in an array. Run your function on `array1`, `array2`, and `array3`.

3. (25 points) Formatted output. Given the arrays

```matlab
ids = [ 1, 2, 3, 4, 5 ];
initial = [ 'T', 'S', 'R', 'B', 'P' ];
ages = [ 15, 24, 19, 18, 30 ];
weights = [ 130.24, 145.2341, 190.123, 126.1, 215.12 ];
```

Write a script called `FormatTable.m` that uses `fprintf` to create the following table:

<table>
<thead>
<tr>
<th>ID</th>
<th>Initial</th>
<th>Age</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>T</td>
<td>15</td>
<td>130.24</td>
</tr>
<tr>
<td>20</td>
<td>S</td>
<td>24</td>
<td>145.23</td>
</tr>
<tr>
<td>30</td>
<td>R</td>
<td>19</td>
<td>90.12</td>
</tr>
<tr>
<td>40</td>
<td>B</td>
<td>18</td>
<td>126.10</td>
</tr>
<tr>
<td>50</td>
<td>P</td>
<td>30</td>
<td>215.12</td>
</tr>
</tbody>
</table>

Make your result match the table above exactly, including field width, alignment, and precision.

4. (20 points) Functions and simple plotting.

(a) Define an array `x` that samples the interval [0, 2π] with 10 evenly spaced points.

(b) Define an array `y` where each element is the sin of the corresponding element of `x`.

(c) Draw a plot of `x` vs. `y`.

(d) Repeat steps (a) - (c), but now sample the interval [0, 2π] using 100 evenly spaced points. How are your results different?