1. Write a function **`nexthour`** that receives one integer argument, which is an hour of the day, and returns the next hour. This assumes a 12-hour clock; so, for example, the next hour after 12 would be 1. Here are two examples of calling this function.

```
>> fprintf('The next hour will be %d.\n', nexthour(3))
The next hour will be 4.

>> fprintf('The next hour will be %d.\n', nexthour(12))
The next hour will be 1.
```

2. Write a function called **`generationXYZ`** that takes as its only input argument one positive integer specifying the year of birth of a person and returns as its only output argument the name of the generation that the person is part of ('X', 'Y', or 'Z ') according to the table below. For births before 1966, return 'O' for Old and for births after 2012, return 'K' for Kid.

Generation	Born
X	1966-1980
Y	1981-1999
Z	2000-2012

3. In chemistry, the pH of an aqueous solution is a measure of its acidity. The pH scale ranges from 0 to 14, inclusive. A solution with a pH of 7 is said to be neutral, a solution with a pH greater than 7 is basic, and a solution with a pH less than 7 is acidic. Write a script that will prompt the user for the pH of a solution, and will print whether it is neutral, basic, or acidic. If the user enters an invalid pH, an error message will be printed.

`7< pH <= 14`	Acid
`pH = 7`	Neutral
`0<= pH < 7`	Base

4. Write a script that will prompt the user for a temperature in degrees Celsius, and then an ‘F’ for Fahrenheit or ‘K’ for Kelvin. The script will print the corresponding temperature in the scale specified by the user. For example, the output might look like this:

```
Enter the temp in degrees C: 29.3
Do you want K or F? F
The temp in degrees F is 84.7
```

5. Write a function called **`sort3`** that takes three scalar arguments. It uses if-statements, possibly nested, to return the three values of these arguments in a single row-vector in increasing order (or more precisely, non-decreasing order), i.e., element one of the output vector equals the smallest input argument and element three of the output vector equals the largest input argument.
NOTE: Your function may not use any built-in functions, e.g., **`sort`**.

6. Write a function called **`classify`** that takes one input argument **`x`**. That argument will have no more than two dimensions. If **`x`** is an empty matrix, the function returns -1. If **`x`** is a scalar, it returns 0. If **`x`** is a vector, it returns 1. Finally, if **`x`** is none of these, it returns 2. Do not use the built-in functions **`isempty`**, **`isscalar`**, or **`isvector`**.

7. Write a function called **`makemat`** that will receive two row vectors as input arguments; from them create and return a matrix with two rows. You may not assume that the length of the vectors is known. Also, the vectors may be of different lengths. If that is the case, add 0's to the end of one vector first to make it as long as the other. For example, a call to the function might be:

```
>> makemat(1:4, 2:7)
ans =
   1   2   3   4   0   0
   2   3   4   5   6   7
```