Homework 2 Revision 1

Here is a summary of the changes:

- The examples in problem 2 were changed to conform to the question
- A phrase was added to question 3 emphasizing that *only* the function, and not a calling program, should be written.
- Problem 4 was changed so that a non-number should give only an error message, unless the non-number was an end of file or “done”; in those cases, exit the program.

Due: October 30, 2024 **Points:** 100

In the examples, input is shown in red, output in black, and the character “↵” means “return” or “enter”.

1. (*15 points*) Write a function to determine whether a year, given as input, is a leap year. A year is a leap year if it is evenly divisible by 4, unless it is evenly divisible by 100 and not 400. So 2000 was a leap year, but 2100 and 2200 will not be. Then write a program that asks the user to enter a year and uses the function you wrote to determine whether the year is a leap year. The program then prints the result.
   Your program must give an error message and exit if the user enters anything other than a positive integer.
   Here is sample output. Each is from a separate run of the program.

   ```
   Year> 2020↵
   2020 is a leap year
   ```

   ```
   Year> 2000↵
   2000 is a leap year
   ```

   ```
   Year> 1900↵
   1900 is not a leap year
   ```

   ```
   Year> hello↵
   You must enter the year as a positive integer
   ```

   ```
   Year> -12↵
   You must enter the year as a positive integer
   ```

   To turn in: Please turn in the program in the file *leap.py*.

2. (*30 points*) Write a function `gcd(m, n)` that calculates the greatest common divisor of *m* and *n*. The greatest divisor of *m* and *n* is the largest *positive* integer *k* that evenly divides *m* and *n* (that is, divides both of them giving a remainder of 0). Use Euclid’s algorithm to calculate this. Here is one very succinct way to describe the algorithm (as usual in Python, *m* % *n* is the remainder of *m* when divided by *n*):

   Repeatedly replace *m* with *n*, and *n* with *m* % *n*, until *n* is 0

   When *n* reaches 0, the value *m* is the greatest common divisor of *m* and *n*.
   Write a program that calls your function repeatedly, until the user enters 0 for *n*.

   ```
   First number: 113↵
   Second number (0 to stop): 293↵
   The greatest common divisor of 293 and 113 is 1
   First number: 14↵
   Second number (0 to stop):18↵
   The greatest common divisor of 18 and 14 is 2
   First number: -30↵
   Second number (0 to stop): -66↵
   The greatest common divisor of -66 and -30 is 6
   First number: 0↵
   ```

```
Second number (0 to stop): 7↵
The greatest common divisor of 0 and 7 is 7
First number): 7↵
Second number (0 to stop): 0↵
```

To turn in: Please turn in the program in the file *gcd.py*.

3. (*30 points*) Write a *recursive* function to print out the depth of lists. For example, [3, [4, 5], [6, [7, [8]]]] has depth 4 because 8 is in a list that's an element in a list that is itself an element in list, which is the the element of the main list.
   Make sure your function is recursive! Also, you only need to write the function; you do not need to write anything that calls the function.

   Examples:

   ```
   listdepth( [ 3, [ 4, 5 ], [ 6, [ 7, [ 8 ] ] ] ])
   4
   ```

   ```
   listdepth( [ [ [ [ [ [ [ ] ] ] ] ] ] ])
   7
   ```

   ```
   listdepth([ "hello [ there ]" ])
   1
   ```

   ```
   listdepth(5)
   Should never get here -- called listdepth on non-list
   ```

   To turn in: Please call your function `listdepth`, your file *listdepth.py*, and submit it to Canvas

4. (*25 points*) Write a program that prompts the user for a list of numbers, one per line. When the user enters "done", print the maximum, minimum, and mean (average) of the numbers entered. If the user enters anything other than a number, and end of file, or "done", give an error message. In those 2 cases, exit the program.
   Here is sample output. Each is from a separate run of the program.

   ```
   Enter number> 13↵
   Enter number> 12↵
   Enter number> 11↵
   Enter number> 10↵
   Enter number> 9↵
   Enter number> xyzzy↵
   Not a number
   Enter number> done↵
   The maximum is 13
   The mean is 11.00
   The minimum is 9
   ```

   ```
   Enter number> -75↵
   Enter number> -135↵
   Enter number> 32↵
   Enter number> 127↵
   Enter number> 21↵
   Enter number> 138↵
   Enter number> 56↵
   Enter number> done↵
   The maximum is 138
   The mean is 23.43
   The minimum is -135
   ```

   To turn in: Please turn in the program in the file *nums.py*.