Programming Assignment 2 - Functions and Conditionals

All solutions are to be written using Python 3. Make sure you provide comments including the file name, your name, and the date at the top of every file you submit. Also make sure to include appropriate docstrings for all functions.

The names of your functions must exactly match the names given in this assignment. For functions expecting more than one parameter, the order of the parameters in your parameter list must exactly match the order given in this assignment.

For any given problem below, you may want to write additional functions other than those specified for your solution. That's fine with us.

Problem 1

Write a function called `max4` that expects four arguments, all numbers, and returns the maximum of the four values. Note: you may not use the built-in `min` or `max` functions here -- you must provide your own logic for this function. Here are some examples of how your function should work:

```
>>> max4(1, 2, 3, 4)
4
>>> max4(4.0, 3.9, -1.7, 2)
4.0
>>> max4(-2, -1, -30, -27.2)
-1
>>> max4(7, 7, 7, 7)
7
```

Problem 2

Write a function called `grade` that expects one argument, a number representing a score between 0 and 100, and returns a single character as the corresponding letter grade shown in the following table:

Score	**Grade**
90-100	A
80-89	B
70-79	C
60-69	D
< 60	F

(this problem continues on the next page)

Assume the argument passed to the function is a valid argument; do not validate the argument. Here are some examples of how your function should work:

```
>>> grade(100)
'A'
>>> grade(0)
'F'
>>> grade(80)
'B'
>>> grade(79.99)
'C'
>>> grade(70)
'C'
>>> grade(65)
'D'
```

Problem 3

The following rhyme helps us remember how many days there are in any given month:

> Thirty days has September,
> April, June, and November,
> All the rest have thirty-one,
> Except February which has twenty-eight.

More accurate versions of the rhyme deal with the problem posed by the leap year, but let's ignore that for now. Write a function called `days` which expects one argument, the name of a month as a string, and returns the number of days in that month as given in the rhyme. Assume the argument passed to the function is a valid argument; do not validate the argument. Here are some examples of how your function should work:

```
>>> days("January")
31
>>> days("February")
28
>>> days("March")
31
>>> days("April")
30
>>>
```

Problem 4

Write a function called `inchesToMeters` that expects one argument, a number representing some length in inches, converts that number to its corresponding value in meters, and returns that value representing the length in meters. For purposes of this program, assume that 1 inch equals 0.0254 meters. Here are some examples of how your function should work:

```
>>> inchesToMeters(1)
0.0254
>>> inchesToMeters(39.37)
0.9999979999999999
```

Problem 5

Write a function called `poundsToKgs` that expects one argument, a number representing weight in pounds, converts that number to its corresponding value in kilograms, and returns that value representing the weight in kilograms. For purposes of this program, assume that 1 pound equals 0.453592 kilograms. Here are some examples of how your function should work:

```
>>> poundsToKgs(1)
0.453592
>>> poundsToKgs(2.2)
0.9979024000000001
```

Problem 6

Back in the mid-1800s, Adolphe Quetelet developed the idea that would eventually be called "body mass index". Body mass index is often used in discussions between patients and health professionals about weight-related risk factors. Body mass index is computed by plugging the appropriate values into this formula:

$$\text{body mass index (BMI)} = \text{weight in kilograms} / (\text{height in meters})^2$$

Write a function called `bmi` that expects two arguments, a number representing the subject's height in inches and a number representing the subject's weight in pounds, and returns the subject's body mass index as computed using the formula given above. Here are some examples of how your function should work:

```
>>> bmi(68, 151.5)
23.03524100992839
>>> bmi(71.5, 220)
30.255793701474445
```

Your `bmi` function should call the functions from Problems 4 and 5 to convert the arguments to the `bmi` function into the appropriate values.

Problem 7

As you learned from Problem 6, body mass index is often used in discussions between patients and health professionals about weight-related risk factors. (It should be noted that many health professionals believe that the body mass index isn't all that useful and the labels associated with the numbers are even less useful.)

Using the functions you wrote for Problems 4, 5, and 6, now write a new function called `bodyMassIndex` that expects no parameters, asks the user for information needed to compute a subject's body mass index, then displays the body mass index along with the BMI category obtained from the table below:

BMI < 18.5	underweight
18.5 <= BMI < 25	normal weight
25 <= BMI < 30	overweight
30 <= BMI	obese

Assume the values entered by the user are valid; do not validate the user input. This function should not explicitly return any value. Here are some examples of how your function should work:

```
>>> bodyMassIndex()
Please enter the subject's name: Eddie
Please enter the subject's height in inches: 70.5
Please enter the subject's weight in pounds: 220
Eddie has a body mass index of: 31.12020146881197
Eddie is obese
>>> bodyMassIndex()
Please enter the subject's name: Albert
Please enter the subject's height in inches: 68
Please enter the subject's weight in pounds: 150
Albert has a body mass index of: 22.80716931676078
Albert is normal weight
```

When we call your `bodyMassIndex` function and give the same keyboard input as in the examples above, everything printed by your function should be identical to what you see in the examples above.

Problem 8

Write a function called `weekly_pay` that expects two arguments, an employee's hourly wage and the number of hours the employee has worked in a week, computes the amount of money to be paid out to the employee for the week, and returns that value as a floating point number. Note that any overtime work (over 40 hours per week) is paid at 150 percent of the regular hourly wage. Assume the arguments passed to the function are valid arguments; do not validate the arguments. Here are some examples of how your function should work:

```
>>> weekly_pay(10, 40)
400.0
>>> weekly_pay(7.50, 20.5)
153.75
>>> weekly_pay(10, 50)
550.0
>>> weekly_pay(20, 41)
830.0
```

Problem 9

The following table contains a simplified explanation of how federal income tax is computed in the United States (the numbers are from a previous decade). Different tax rates are applied based on the taxpayer's marital status and total income.

If your status is Single and if the taxable income is	**the tax is**	**of the amount over**
at most $32,000	10%	$0
over $32,000	$3,200 + 25%	$32,000
If your status is Married and if the taxable income is	**the tax is**	**of the amount over**
at most $64,000	10%	$0
over $64,000	$6,400 + 25%	$64,000

Write a function named `taxes` that computes a taxpayer's income tax. The function expects two arguments. The first argument is either "s" for a single taxpayer or "m" for a married taxpayer. The second argument is a number representing the taxpayer's taxable income. Your function should use the information in the table above to compute the taxes due. Assume the arguments passed to the function are valid arguments; do not validate the arguments. Here are some examples of how your function should work:

```
>>> taxes("s", 25000)
2500.0
>>> taxes("s", 32000)
3200.0
>>> taxes("s", 80000.00)
15200.0
>>> taxes("m", 50000.00)
5000.0
>>> taxes("m", 64000)
6400.0
>>> taxes("m", 160000)
30400.0
>>> taxes("s", 14700.50)
1470.0500000000002
>>> taxes("m", 172535.38)
33533.845
```

Where to do the assignment

You can do this assignment on your own computer, or in the labs. In either case, use the IDLE development environment -- that's what we'll use when we grade your program. Put all the functions you created in a file called "prog2.py".

Submitting the assignment

We will be using Canvas to turn in assignments. Submit the file containing your functions. Do NOT cut-and-paste your functions into a text window. Do NOT hand in a screenshot of your functions' output. We want one file from you: "prog2.py".

Saving your work

If you are working in the lab, you will need to copy your program to your own flash-drive. To save it on flash-drive, plug the flash-drive into the computer (your TA or the staff in the labs can help you figure out how), open the flash-drive, and copy your work to it by moving the folder with your files from the Desktop onto the flash-drive.

Grading the assignment

Please note that we may not grade every one of the nine solutions you submit.