Functional Overload

Due: Monday, November 19th

PLEASE READ BEFORE STARTING

Objective: The aim of this assignment is to have you become more familiar with how to define functions, as well to further develop your critical thinking skills. You are required to write ***eight*** functions as specified below. However, it will be up to you to define the function and make sure that they are working as intended. Do note that the names of the function ***MUST*** be spelled as they are shown, along with any given parameter as well.

Required Functions

is*_int(data)*

Description: Given some information (can be a string, or a float, etc) in the form of the parameter data, determine if it is an integer data type. If it is, return **True**. Otherwise, return **False**.

Output: This function should not print anything onto the screen.

Return Value: Return **True** if the passed parameter data is an integer data type. Otherwise, return **False** if it is not.

is_even(n)

Description: Given an integer *n*, determine if it is an even integer, or an odd integer. The mathematical definition of an even integer is one such that the following condition is met: $n = 2k$ for some k. Likewise, for an odd integer the following condition must be met: $n = 2k + 1$ for some k. **Note that, based on these definitions you must determine if the operation of *n* and 2 returns a 0 (for even) or a 1 (for odd).** This operation is one you are already aware of.

Output: This function should not print anything onto the screen.

Return Value: Return **True** if the passed integer n is even. Otherwise, return **False**.

string_contains_char(str, ch)

Description: Determine if the passed string *str* contains the character *ch*.

Output: This function should not print anything onto the screen.

Return Value: Return **True** if the passed character is found within the string. Otherwise, return **False**.

find_distance_traveled(time, rate)

Description: Given a *time* (in minutes) and a *rate* (in feet per minute), determine the distance traveled. This is a well-known formula in mathematics, and as a refresher it is the following: *distance = time * rate*.

Output: This function should not print anything onto the screen.

Return Value: This function must return a float data type. Please disregard the precision of the computed value.

print_travel_data(dist, time, rate)

Description: Of the three parameters, one will be guaranteed to be a zero. With the other parameters (which will be positive and non-zero), calculate the value of the "missing" parameter and print it to the screen. For example, if the parameter *dist* is equal to 0, calculate the distance traveled. If the parameter *time* is equal to 0, calculate the time spent traveling.

Output: As examples, if the *dist* parameter is found to be 0, print the following string, excluding quotation marks: "Distance traveled: 12.00 feet". If the *time* parameter is found to be 0, "Time spent traveling: 2.50 minutes". If the *rate* parameter is found to be 0, "Rate traveled: 22.37 feet per minute". **NOTE THAT THESE ARE SAMPLE OUTPUTS AND THAT THE NUMERICAL VALUES MUST COME FROM THE VALUES YOU COMPUTED**. Do note that this output contains formatted output, and that the floats must be to a precision of 2 decimal points.

Return Value: None. This is a void function.

fahrenheit_to_celcius(f)

Description: Given a temperate *f* that is in degrees Fahrenheit, convert it to its

equivalent value in degrees Celsius. Please reference the following formula:
$C = (F - 32) * (5.0/9.0)$.

Output: This function should not print anything onto the screen.

Return Value: This function must return the equivalent Celsius temperature as a float data type. Please disregard the precision of the computed value.

find_floor_distance(floorOne, floorTwo)

Description: Two floors in a building will be passed to this function. The distance between these floors must be found. For example, if the 5th floor and the 3rd floor are passed, the distance between them is 2 floors. During testing, note that a value of 0 may be passed into the function. For the purposes of this assignment, assume that this cannot occur. As such, return a -1. In addition, if the calculated distance is found to be negative, convert it to a positive.

Output: This function should not print anything onto the screen.

Return Value: Return the value of the distance between both floors. If either of the values of the parameters is 0, return a -1.

Special Restriction: Do **NOT** use the *abs()* function.

find_inclusive_sum(start, end)

Description: Given a *start* and an *end*, determine the inclusive sum between them. For example, the return value of *find_inclusive_sum(1, 5)* would be 15 as 15 = 1 + 2 + 3 + 4 + 5. Note that this sum **CANNOT** be negative, which can only happen if the *end* is greater than the *start*. If this occurs, return a -1. If the *start* and *end* are equal to one another, return the value itself. For example, *find_inclusive_sum(5,5)* will return 5.

Output: This function should not print anything onto the screen.

Return Value: Return the inclusive sum of the specified numbers. Return -1 if *start* is greater than *end*.