Question 2: Pizza Shop (85 points)

In this question we will do some calculations for a pizza shop: figuring out the cost of certain pizzas and pizza-related items, and letting the user interact with our program through a menu of different program options.

There will be a few different calculations, and each will be placed in its own function. Whenever you are faced with a complex problem, which will most often be the case when programming, it is best to break it down into smaller sub-parts. Each different computation should be put into its own function.

This organizational paradigm serves three purposes. First, it makes your code file much easier to read through and navigate. Secondly, and most importantly, it lets you focus on one thing at a time. Finally, it helps with testing. Each time you write a function, you should always **test it thoroughly**, giving it different inputs and checking that the output is what you expect it to be. It is easier to test functions when they are small and do not do too much on their own. Only once you are certain that a function is working properly, then you can move onto the next function.

Functions also help in code re-use. By putting some computation into a function, if we then needed to perform the same computation again, we can just call that same function instead of copy-and-pasting the code to a different location (as long as the function returns a value). Copy-and-pasting is bad in programming! One of the keys of programming is **don't repeat yourself** ('DRY'). Whenever you have repeated several lines of code, ask yourself if there is a way to simplify the code by putting it into a function instead (and then just calling it when needed). We will also be checking your code to make sure that not too much code is repeated throughout your file.

With the above in mind, we will begin writing our code. Write all your code for this question in a file called **`pizza.py`**.

First, define the following global variables at the top of your file. You will use these variables in certain functions of your code.

- `PIZZA_CAKE_COST_PER_CENTIMETRE_CUBED = 4.0`
- `PIZZA_POUTINE_COST_PER_CENTIMETRE_CUBED = 3.0`
- `SPECIAL_INGREDIENT = "guacamole"`
- `SPECIAL_INGREDIENT_COST = 19.99`
- `FAIR = True`

Note that the names of these global variables are in all-caps. Variables in all-caps are by convention known to be 'constant' variables, or 'constants' for short. A constant is a variable whose value will never change throughout the execution of the program (e.g., it does not depend on user input nor on the result of any calculation, and we never want to update it). It is set once, typically at the top of your code file, and not modified afterwards in the code. It is, however, always possible for you to modify the value of a constant yourself, by changing its value in the code directly.

The examples shown for the functions below will assume that the constants are set to the above values. However, during testing, we will modify the values of these global variables and check that your code still works properly. You should make sure that your code uses these global variables when appropriate and that you do not instead 'hard-code' the values (meaning writing the values directly into the code instead of using the global variables).

You must write the following functions for this assignment:

- `get_pizza_area(diameter)`: Takes a positive float for the diameter of a pizza, and returns its area as a float. The float should **not** be rounded. We round it only in the examples below by calling the `round` function.

```
>>> round(get_pizza_area(1.5), 2)
1.77
```

```
>>> round(get_pizza_area(2.0), 5)
3.14159
```

- `get_fair_quantity(diameter1, diameter2)`: Takes two positive floats, for the diameter of two pizzas. Returns as an integer the minimum number of small pizzas that must be ordered to get at least the same amount of pizza (by area) as one large pizza. (Note that either input to the function may be the smaller or larger one.)

  If the `FAIR` variable is set to `False`, then return 1.5 times the true amount (i.e., the amount that would otherwise be returned directly), rounded down to the nearest integer.

  In the first example below, given a pizza of diameter 14.0, it will take 4 pizzas of diameter 8.0 to get at least the same amount of pizza by area. We can calculate this by finding the area of each pizza size ($\pi(\frac{14}{2})^2$, $\pi(\frac{8}{2})^2$) and checking how many multiples of the smaller quantity are needed to get equal to or more than the bigger quantity.

```
>>> get_fair_quantity(14.0, 8.0)
4
>>> get_fair_quantity(3.0, 14.0)
22
```

- `pizza_formula(d_large, d_small, c_large, c_small, n_small)`: Takes four positive floats: the diameter of the large pizza, diameter of the small pizza, cost of one large pizza, cost of all the small pizzas, and one positive integer, the number of small pizzas, as input. Exactly one of the five inputs will be the integer `-1`, which we call the 'missing' input value. The function must use the other four values to calculate and return the missing input value as a float, rounded to two decimal places. For example, if the first input (diameter of large pizza) is `-1`, then your function must use the other four inputs to calculate and return the diameter of the large pizza. (Assume that the number of large pizzas is 1.)

  Your formula should be based on the following relationship: the total amount of small pizzas (by area) per dollar should be equal to the amount of one large pizza (by area) per dollar.

  Note: For this function, there will be some repetitive code. Try to reduce the amount of repetition where you can.

```
>>> pizza_formula(12.0, 6.0, 10.0, -1, 2)
5.0
>>> pizza_formula(14.0, 8.0, 9.55, -1, 4)
12.47
```

- `get_pizza_cake_cost(base_diameter, height_per_level)`: Takes one positive integer (diameter of the cake base) and one positive float (height in centimetres per level) as inputs. Returns the cost of the pizza cake as a float rounded to two decimal places.

  A pizza cake is a stack of pizzas all on top of each other, with the largest at the bottom (with the given integer base diameter). Each successive pizza on top will be 1cm smaller in diameter than the one below. The pizza on the very top will have diameter 1cm. The total cost of the cake will be the sum of all individual pizzas. A pizza cost can be calculated by multiplying its area by the given height and by the global variable that gives the cost of pizza cake by centimetre cubed.

  If the `FAIR` variable is set to `False`, then return 1.5 times the true amount (rounded to two decimal places).

  Note: You may need to use a loop for this function.

```
>>> get_pizza_cake_cost(2, 1.0)
15.71
```

```
>>> get_pizza_cake_cost(10, 2.0)
2419.03
```

- `get_pizza_poutine_cost(diameter, height)`: Takes one positive integer (diameter of the poutine cylindrical cup) and one positive float (height in centimetres) as inputs. Returns the cost of the pizza poutine as a float rounded to two decimal places.

  A pizza poutine is a cylinder containing pizza, fries and gravy. The total cost of the poutine can be calculated by multiplying its volume by the global variable indicating the cost of a pizza poutine per centimetre cubed. Recall that the volume of a cylinder is $\pi * r^2 * h$, where r is the cylinder's radius and h is its height.

  If the `FAIR` variable is set to `False`, then return 1.5 times the true amount (i.e., the amount that would otherwise be returned directly), rounded to two decimal places.

```
>>> get_pizza_poutine_cost(2, 1.0)
9.42
>>> get_pizza_poutine_cost(10, 2.0)
471.24
```

- `display_welcome_menu()`: Takes no inputs and returns nothing. Displays to the screen a menu with a welcome message and the three options available to the user. You can choose your own welcome message, but your menu must include the three options in the given order (and with A. B. and C. as the options). You can modify the option titles if you like, though they should still have the same meaning as the original (e.g., you could write 'Choose a special order' instead of 'Special Orders').

  Note: You only need one example for the docstring for this function.

```
>>> display_welcome_menu()
Welcome To The Best Pizza Place. Our Pizzas Made With 100% Real Pizza.
Please choose an option:
A. Special Orders
B. Formula Mode
C. Quantity Mode
```

- `special_orders()`: Takes no inputs and returns nothing. Asks the user to choose between cake and poutine, asks them to enter a value for the diameter and for the height, and if they want the special ingredient, then prints out the total cost of the order. The user should be able to type in either `'y'` or `'yes'` to indicate that they want the special ingredient, in which case the cost of the special ingredient is added to the cost of the order. If they type in anything else, then the cost of the special ingredient is not added.

  Note: Make sure to use the two global variables for the special ingredient, as discussed earlier in these instructions. Do not hardcode the name or value of the special ingredient directly.

  (Note: Text in examples with a light-gray background corresponds to input entered by the user.)

```
>>> special_orders()
Would you like the cake or poutine? cake
Enter diameter: 2
Enter height: 1.0
Do you want the guacamole? yes
The cost is $35.7
>>> special_orders()
Would you like the cake or poutine? poutine
Enter diameter: 2
Enter height: 1.0
```

```
Do you want the guacamole? no
The cost is $9.42
```

- `quantity_mode()`: Takes no inputs and returns nothing. Asks the user to enter in the diameters of two pizzas, and then prints out the minimum number of smaller pizzas they would need to buy to get at least the same amount of pizza (by area) as one large pizza.

```
>>> quantity_mode()
Enter diameter 1: 14.0
Enter diameter 2: 8.0
You should buy 4 pizzas.
```

- `formula_mode()`: Takes no inputs and returns nothing. Asks the user to enter in the diameter of a large and small pizza, cost of a large and small pizza, and number of small pizzas (in that order), with one of the values being `-1`, and prints out the actual value of the input that was given as `-1`.

```
>>> formula_mode()
Enter large diameter: 12.0
Enter small diameter: 6.0
Enter large price: 10.0
Enter small price: -1
Enter small number: 2
The missing value is: 5.0
```

- `run_pizza_calculator()`: Takes no inputs and returns nothing. Displays a welcome message to the user and list of program options, asks the user to input an option, and then calls the appropriate function. If the user inputs an invalid option, prints out `"Invalid mode."`.

```
>>> run_pizza_calculator()
Welcome To The Best Pizza Place. Our Pizzas Made With 100% Real Pizza.
Please choose an option:
A. Special Orders
B. Formula Mode
C. Quantity Mode
Your choice: A
Would you like the cake or poutine? cake
Enter diameter: 2
Enter height: 1.0
Do you want the guacamole? yes
The cost is $35.7

>>> run_pizza_calculator()
welcome to pizzas R us. no credit cards accepted, cash only. come to back alley 4 pickup.
choose option:
A. Special Orders
B. Formula Mode
C. Quantity Mode
Your choice: B
Enter large diameter: 12.0
Enter small diameter: 6.0
Enter large price: 10.0
Enter small price: -1
Enter small number: 2
```

```
The missing value is: 5.0
```

```
>>> run_pizza_calculator()
Welcome to Flying Pizzas. If you can catch the pizza as it flies out of the oven,
you get your pizza free! Note: Must bring your own protective gear.
Please choose an option:
A. Special Orders
B. Formula Mode
C. Quantity Mode
Your choice: D
Invalid mode.
```

In the example below, the welcome message, menu options and other text are customized (in reference to a popular video game). Note however that the menu options are still labelled A., B., and C., and still have the same meaning as the original text. Note also that the inputs are asked for in the same order as described in the functions above, and the values (in this case, the final cost) are still displayed as required.

(In addition, `FAIR` is set to `False` for this example.)

```
>>> run_pizza_calculator()
HEY      EVERY      !! EV3RY  BUDDY  'S FAVORITE [Pizza Shop]
HURRY UP AND BUY! I JUST NEED YOUR [Account Details] AND THE [Number on theB4ck]!
YUM YUM GREAT OPTIONS
A. [Specil Deals]
B. [Prize Winning] F0RMULA
C. HOW MUCH TO BUY?? [You Can Never Buy Enough]!!
TAKE AN OPTION: A
DO YOU WANT THE [[Mouth-watering cake]] or [[Terrifying]] POUTINE? cake
WOAH!! DIAMETER?? 2
AND HEIGHT??? 1.0
DO YOU WANT THE PIZZA INSURANCE? no
DON'T WORRY KIDS I'M AN [HonestMan]! JUST DEPOSIT 23.56 KROMER
```