Project #2 – Bailey's Brownies

1 Objective

Use expressions, calculations, operator precedence, and formatted output.

2 Problem Overview

Your friend Bailey has a side business baking delicious brownies. You volunteer to write a program to help Bailey estimate the amount of ingredients to buy for a given order, based on the number of people to be served. You will also calculate the total price of ingredients Bailey will pay at the grocery store.

3 User Interface

3.1 Input

The program should ask for the number of customers expected and let the user respond, e.g.,

```
How many people to be served?  100
```

3.2 Output

The program should display the supply list, with estimates shown in whole numbers only[1]. The output is formatted and aligned so it is attractive and easy to read, as shown below. Do not use tabs or spaces to right align the numbers; use the format() function instead. *Note that the numbers here are merely samples and are intentionally incorrect; you will need to figure out what is right!*

```
To serve 100 make 3 batches

------------------------------
      Grocery List
------------------------------
Cocoa               4
Salt                3
Baking Powder       1
Espresso Powder     6
Sugar               1
Flour               8
Chocolate Chips     2
Vanilla             9
Eggs                4
Butter              2

Total price $    96.70
```

[1] With fractional output, round up, e.g., if 1.5 cans of baking powder are calculated, then 2 cans must be purchased. But do not add a *whole* number, e.g., if *exactly* 1 bag of flour is calculated, do not show 2 bags. Use only the int() function and math to do this work; do not use any other Python functions.

4 Calculations

4.1 Batches

From history, Bailey knows this about customers and baking ingredients:

- The recipe makes 24 brownies, each about 2 inches square
- Each person will get two brownies, so one recipe feeds 12 people

4.2 Ingredients

Here are the ingredients for one batch, along with container purchase information from a local store[2].

Ingredient	Measured by	Amount	Container	Price $
Cocoa powder	weight	106 grams	8 ounces	1.99
Salt	weight	6 grams	26 ounces	0.49
Baking powder	weight	5 grams	8.1 ounces	1.89
Espresso powder	weight	2 grams	7.05 ounces	5.39
Sugar	weight	447 grams	4 pounds	1.99
Flour (all-purpose)	weight	180 grams	5 pounds	1.99
Chocolate chips	weight	340 grams	12 ounces	1.99
Butter	weight	.5 pounds	1 pound	2.99
Vanilla extract	volume	1 tablespoon	2 ounces	10.59
Eggs (large)	each	4	18	1.99

4.3 Conversions

Weight Conversion

- 1 ounce = 28.3495231 grams
- 1 pound = 16 ounces

Volume/Liquid Conversion

- 1 cup = 8 ounces
- 1 cup = 48 teaspoons
- 1 tablespoon = 3 teaspoons

4.4 Readability/Clarity

Do not hide your work by doing calculations outside the program and using the results; all calculations must be visible and clear. Use good variable names and code comments where helpful. Create constants for "magic numbers" that might not make sense to readers; put those in their own section at the top, and name them in all caps with underscore word separators, e.g., PRICE_COCOA.

[2] It is an unfortunate truth in baking that weights are always given in grams, while in the USA products are sold using imperial weights like pounds and ounces; I am not making this up to make life difficult for you. Luckily, electronic kitchen scales can usually be set to either scale. It is also fortunate that you have taken math that allows you to deal with these conversions much more easily than can most bakers like Bailey.

5 Code Specifications

- Include header comments at the top of the file. Include your name, the date, and a brief description of what the program does.
- Include comments for each section, introducing what is going on below, e.g., `# calculate ingredients needed`
- Use comments elsewhere as you think they help guide the reader. Do not overdo, though! Not every line needs a comment; think about describing a block of related code.
- Use blank lines to separate sections and provide visual "breathing room."
- Use descriptive variable names.

6 Hints

- Do this in stages, e.g., get the input and display it. Calculate a few things and display them so you know you are right. If you try to do too much at once you may have a hard time debugging.

7 Testing

- Develop an appropriate number of test cases. Calculate results using some other method (e.g., by hand, using Excel, etc.), then confirm the program yields the same results.
- Test a few error cases and see what happens. At this stage you are not equipped to solve all problems you would like to; keep notes about what you would like to do once you learn more.
- Remember that testing is not just about calculations; UI needs testing as well.
- Document your testing and results in comments at the bottom of the program as shown below.

8 Summary

At the bottom of your program, add comments that answer these questions:

- How did you approach this assignment? Where did you get stuck, and how did you get unstuck?
- How did you test your program? What does not work as you would like, perhaps things that you would like to fix as you learn more?1
- What did you learn from this assignment? What will you do differently on the next project?

9 Grading Matrix

Area	**Percent**
User input	10
Result correctness – ingredients	20
Result correctness – prices	10
Visibility/clarity of data and calculations	10
Output – formatting, alignment, spacing	20
Good internal documentation including comments, variable naming, use of blank lines	10
Test cases	10
Summary report provided and complete	10
Total	**100**