INSTRUCTIONS

The purpose of this assessment is to complete a simple programming assignment.
You are required to produce working and tested source code to solve the problem below.

The exercise must be completed in one of: **Python, Golang, or JavaScript (Node.js)**.
Your code must be fully self-contained and need no external libraries other than the core.

It is expected that this exercise will take around **half a day to complete**.

Materials to Submit
However you choose to compile / build your code, please do not make any assumptions about tooling. You MUST submit the following:

- a zip file containing the complete code
- a README.txt explaining how to build / run the code from the command line

Expectations

This exercise is for us to get to know how you think about problems, and provide us with discussion material in a technical interview rather than an exam with right or wrong answers.

The code should be well structured, suitably commented, have error handling and unit tests. The comments should help give context about *why* you have made a particular decision, as opposed to just saying what the code already says. Any log messages you include should indicate forethought about supporting / debugging in a live environment.

100% test coverage, exhaustive error handling, and full logging implementations are not required. We only wish to see that you know how to write testable code, how you deal with errors, and how you use logging to help with debugging in a production setting.

Post-Assessment

At the interview, we will use your submission as the basis for discussion. **We will expand the scope of the exercise by adding additional requirements (including some non-functional requirements)**.

Additionally, you will be expected to walk through your code with the assessor. For example, justifying the design decisions you made and walking them through your thought process.

You should be prepared to discuss how you would scale the code you've written, including the trade-offs of your choices.

CODING ASSIGNMENT

Write a program and associated unit tests that can price a basket of goods, accounting for special offers.

The goods that can be purchased, which are all priced in USD, are:

Soup – 65p per tin
Bread – 80p per loaf
Milk – £1.30 per bottle
Apples – £1.00 per bag

Current special offers are:

Apples have 10% off their normal price this week
Buy 2 tins of soup and get a loaf of bread for half price

FUNCTIONAL REQUIREMENTS

- The program should accept a list of items in the basket and output the subtotal, the special offer discounts and the final price

- Input should be via the command line in the form: PriceBasket item1 item2 item3
  For example: PriceBasket Apples Milk Bread

- Output should be to the console, for example:

  ```
  Subtotal: £3.10
  Apples 10% off: -10p
  Total: £3.00
  ```

- If no special offers are applicable, the code should output:

  ```
  Subtotal: £1.30
  (no offers available)
  Total: £1.30
  ```

Please see the next page for the non-functional requirements.

NON-FUNCTIONAL REQUIREMENTS

The code and design should meet the functional requirements but be sufficiently flexible to allow for future extensibility.

Particular attention will be paid to how well you've considered how things will be at large-scale (e.g. if there were ten of thousands of products and offers, or if there were thousands of baskets to be priced) - particularly with regard to:

- thread safety
- resource usage
- speed

You don't need to have solved every edge case for these things, we're just looking to see that you've structured your code in a way that shows you've considered them.

In some cases, it's good enough to just comment something as being inefficient or not thread-safe.

We can then discuss in your tech interview what the alternatives are and some of the pros/cons around those alternatives.