The Supermarket Problem

*Note: Please **do not make either this document or your solution public**. Do not commit your solution to a public Git repository.*

We want to understand how you think and the level of craft you bring to bear when building software.

A full-scale, real-world problem would be ideal, but to better respect your time we have defined a simple exercise that we want you to solve in the same way you would solve a real problem in your workplace. That means using your real workflows, good design principles, standard coding conventions and a sensible project structure. It should take you around 8 hours to complete the task.

Your solution should be implemented in Java 11, without using external libraries (except for testing). The solution should read from standard input and file and print to standard output. Please follow the exact syntax for both input and output described below, as we will validate the correctness of your solution using automated scripts.

Please use Git for version control to help us understand your workflow and how your solution evolved. The *.git* folder must be included in your submission archive.

Please use Maven or Gradle as your build tool.

Please follow the instructions carefully. Remember that correctly interpreting and implementing requirements is an important part of software development.

Problem Statement

As a user, I want to be able to add items to a shopping cart so that I can see the total price. The total price takes into account the quantity of each item as well as offers and discounts. Items are limited and are stored in an inventory. Item names don't contain spaces. For all amounts, round to 2 decimal places using common-rounding.

Inventory

The inventory is a list of available items, the unit price, and their quantities. The initial values are loaded from a file. The file needs to be specified as an input when running the program.

Each row should have the following format:

```
<product_name>,<amount>,<quantity>
```

Ex.

inventory.csv

```
soap,10.00,100
bread,2.50,10
```

Note: You are not expected to update values in the file during runtime.

Offers

Your shopping cart should support the following offers:

1. Buy 2 Get One Free (*buy_2_get_1_free*): When applied to an item, every third item added is free;
2. Buy 1 Get 50% Off on next (*buy_1_get_half_off*): Every second item is charged at half the price.

Running the program

The application should allow input in two different ways:

- if only the inventory argument is provided, the application starts an interactive prompt, where the actions can be typed in;
- if the commands file path is provided as a second argument, such a file is parsed and the commands in it executed one by one.

In both cases, the commands are separated by a new line. If two offers are applied at the same product, only the more recent one stands.

Interactive mode

```
$ ./supermarket inventory.csv
$ checkout
```

```
empty cart
$ add soap 5
added soap 5
$ add bread 1
added bread 1
$ bill
subtotal:52.50, discount:0.00, total:52.50
$ offer buy_2_get_1_free soap
offer added
$ bill
subtotal:52.50, discount:10.00, total:42.50
$ add soap 1
added soap 1
$ bill
subtotal:62.50, discount:20.00, total:42.50
$ offer buy_1_get_half_off bread
offer added
$ add bread 1
added bread 1
$ bill
subtotal:65.00, discount:21.25, total:43.75
$ checkout
done
```

File mode

```
$ cat commands.txt
checkout
add soap 5
add bread 1
bill
offer buy_2_get_1_free soap
bill
add soap 1
bill
offer buy_1_get_half_off bread
add bread 1
bill
checkout

$ ./supermarket inventory.csv commands.txt
```

Solution submission

Before submitting your solution you need to update the following scripts (simple wrappers around the underlying build tool of your choice to invoke the specific target)

in the provided archive:

- **bin/cleanup.sh**: it should clean the build files;
- **bin/run.sh**: it should run the applicationand pass the input arguments to the app (as we will invoke it with our test CSV files);
- **bin/setup.sh**: it should build and package the application.

The specific instructions are in each script.

Once these are done you can run the script to submit the solution, *submit.sh* or *submit.bat*, which will build the application and run some basic tests. It will also generate an archive, this is the one you need to provide to submit the solution.

Please note that the submit script requires docker to be installed, as the commands are executed inside a container.