CSCI291 Lab1: Coffee Maker Simulator(50 marks)
Deadline: 27/10 @ 11:55 via Moodle

The aim of this task is to implement in C a simulator for a self-service coffee maker. The program must include several user-defined functions; **it should NOT make use of arrays, pointers or C-structs.**

For simplicity, we assume that the coffee maker can make three types of coffee: Espresso, Cappuccino, and Mocha. Table 1 gives the price of each coffee type cup along with the quantities of ingredients to make one cup.

Table 1: Coffee Type Cup: Ingredients, Quantity Requirements, and Price

Coffee Type	Coffee Beans (grams)	Water (milliliters)	Milk (milliliters)	Chocolate Syrup (milliliters)	Price (AED)
Espresso	8	30	-	0	3.5
Cappuccino	8	30	70	0	4.5
Mocha	8	39	160	30	5.5

The program should declare/define:

- A set of defined constants (use the directive # define) for,
  - The quantity requirement of each ingredient to make a cup of coffee.
  - The admin password.
  - The low threshold quantity for each ingredient in the machine. When the quantity of any ingredient in the machine becomes less than or equal to its corresponding threshold value, an alert should be sent to the coffee maker operator. This alert is modelled by printing a relevant message.
- A real-value variable *total_amount* which gives the total sales amount.
- A variable for each attribute in table 1

In the following, we assume all user inputs are valid; any printed value should occupy a field of 12 spaces, be left aligned, with 2 fractional precisions when applicable.

Requirements:

First, the program should display a menu-driven interface for the user; the code of the menu should be embedded within an *infinite* loop, whereby three options are proposed:

1. Order a coffee type
2. Admin mode for the coffee maker operator
3. End the application execution.

[**5.5 marks** for the main, data declaration and overall structure of the program]

1. Order a Coffee Type for a User [17.5 marks]:

This part of the program aims to allow a user to buy a cup of coffee from the machine. It should display a menu-driven interface; embedded within an infinite loop, following this sequence of operations:

1.a. Display all coffee types that the coffee maker machine can serve. For each type, show its name and price. A coffee type can be served only if the required quantities of its ingredients are available in the coffee maker. Otherwise, display the coffee type with the message " Unavailable due to temporarily insufficient ingredients". If no coffee can be served, exit the function.

1.b. Prompt the user to input his/her selection: a dedicated integer value for each coffee type or zero to exit the function.

1.c. Prompt the user to confirm his/her selection after displaying the selected coffee type and its price. If not confirmed, go back to step (1.a).

1.d. Prompt the user to pay for his/her order by inserting one of two coins: 1 and 0.5 Dirham, one at a time until the total price of the cup is covered. If an invalid coin is inserted, print a message asking the user to collect the invalid coin and insert a valid one. For simplicity, we assume that the user will ultimately pay the correct amount.

1.e. Update the ingredient quantities used to prepare the coffee and print to the user the coffee cup (s)he has just bought, its price and how much (s)he paid for it along with any due change.

1.f. If the new quantity of any ingredient falls below or equals its associated minimum threshold, send an alert to the operator by printing a relevant message.

2. Admin Mode [10 marks]:

The following sequence of operations should be coded:

2.a. Prompt the user to input the admin password; if it is incorrect, print a relevant message and exit the Admin mode.

2.b. Once the correct password is entered, display to the coffee maker operator a menu-driven interface; the code of the menu should be embedded within an infinite loop, presenting the following options:

1: Display the quantity of each ingredient in the machine and the total sale amount
2: Replenish ingredients in the machine
3: Change coffee price
0: Exit the Admin Mode function

Each of the above operations should be implemented in a dedicated function as follows:

2.b.1. Replenish ingredients: Randomly[1] reset the quantity of each ingredient in the coffee maker, choose a relevant range.

2.b.2. Allow the operator the option to change the price of any coffee type in the coffee maker and update it accordingly.

[1] https://www.javatpoint.com/random-function-in-c

2.b.3. Display the total sale amount (*total_amount*); prompt the operator if (s)he would like to reset it to zero. If so, reset the *total_amount* variable and remind the operator to collect the money from the machine.

3. **Exit**: Exit the infinite loop of the program and end it.

What to submit

Your submission must include:

- Source file (*.c) with concisely commented code.
- A description of the design solution, detailing the modular/functions structure of the implementation with explanation.

  [3 marks]
- Testing evidence with discussion.

  [**4 marks]**
- Weekly progress evidence submission to github and in-lab assessment through lab instructor observation and QA.

  [10 marks]