Lab 2 – CSCI291

Wed, Sept 25th, 2024
Rewa Raya 8287338

Introduction:

This laboratory report showcases the sections that make a coffee machine simulator with their descriptions of the simulator's code that was written in C language. There are many functions expected from the coffee machine other than ordering coffee, such as the admin mode and the alert of the insufficient amount of each ingredient used for the types of coffee the machine serves.

This coffee machine servers up to three types of coffee: Espresso, Cappuccino, and Mocha, each at a fixed price (in AED) of 3.50, 4.50 and 5.50, respectively – the prices can be changed in the admin mode if wished. Moreover, each coffee has specific ingredients to use, with the amount of each ingredient for each coffee being specified. This information is displayed in Table 1.

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	0	3.50
Cappuccino	8	30	70	0	4.50
Mocha	8	39	160	30	5.50

Coffee Maker Simulator - Design Description and Analysis:

At the start of any C program, libraries that will be needed for the program to run must be included using `#include<FileName>`. The program used three header files: `stdio`, `math` and `stdlib` – the first one is to be able to use input and output functions such as `printf`, the second one is to use mathematical processes, and the third is for the `rand` function that generates a number randomly. After the header files, the constants that will be used in the program must be defined using `#define`. The constants that were defined in the program are the ingredient quantities for each coffee type (if they have the same amount for a specific ingredient, then only one constant

variable is defined for that ingredient), the admin's password, and the minimum threshold quantities for each ingredient.

Declaring variables outside a function makes them global variables - these variables can be mentioned in any function with the same value. Nine global variables were set for this program, of which are the prices of each coffee (they are not a constant as the admin can change the prices), the total amount of money in the machine (it will be used in several functions hence it is a global variable), the variable "`price`" which is used in different functions and it must be processed with the same value, and lastly, the amount of each ingredient present in the coffee machine.

The coffee machine has seven different functions besides the main function; each function has its definition and reason for its purpose. No returns are expected from the functions; hence, they are `void` functions. Only two functions required input arguments: the function that updates and replenishes the ingredients needed inputs of each ingredient variable, and the payment check function needed the price as the input. Further description of each function will be discussed further in the report.

The main function of the code is to display the menu-driven interface to the user. It displays three options: the coffee menu, the admin mode, and an exit option, which stops the execution of the program. Each option is numbered in the interface – this is to be able to ask the user for a number referencing the option they want. If an incorrect number was the input, a message is printed to the user to inform them they input a wrong number, and the loop will showcase the main menu interface again. The main function is in a `while(1)` loop as a method to keep returning to this interface in specific cases, such as exiting admin mode and going to the coffee menu to check if prices have changed. Image 1 shows the main menu interface in the case of exiting the program.

Image 1 – Main Menu Interface

```
---- Coffee Machine Menu ----

1. Order Coffee
2. Admin Mode

0. Exit

Select an option from the list above by using the corresponding number:
0
Program exited successfully.
```

To be able to create the situation depending on the user's input, a `switch` is used instead of using nested `if…else` statements. Each `case` of the `switch` is labelled with an integer number, which corresponds to the input given by the user – an example is the "Order Coffee" option - `case 1` contains the function call to the ordering coffee function, and then a `break` is there to break the `case` to not move on to other cases. The second case will be `case 2,` which calls the admin mode function and also uses a `break`. The exit has zero as its number; hence, `case 0`, it will print a message that the function is successfully exited, and it uses `return 0` to exit the main function; it is an `int main()` function, so a data type of `int` is returned. A `default` case is used for this switch, which is the case when the user does not enter the correct number. To be able to refer the cases with the user's input variable `x` is declared to store the value of the scanned number the user prints.

The `void orderCoffee()` function displays the coffee menu interface with the list of the coffee types and their prices, and there is an exit option that takes the user back to the main menu interface. The coffee menu interface can be seen in Image 2, where the user inputs zero and the program displays the main menu.

Image 2 – Coffee Menu Interface

```
              MENU

1.  Espresso,           3.50 AED
2.  Cappauccino,        4.50 AED
3.  Mocha,              5.50 AED

0. Exit

Choose the coffee you want by typing its number represented in the Menu:
0

Please wait for a moment.

---- Coffee Machine Menu ----
```

Similarly, to the `switch` used in the `main` function, a `switch` was also used for this function in order to create the command calls depending on the chosen option by the user. If the user inputs 1, `scanf` reads the number and stores it in variable `opt,` which is used for the case labels of the `switch`; hence, the program does the code lines under `case 1` till the `break` is read by the compiler. `case 1` is for ordering an espresso, `case 2` is for ordering a cappuccino, `case 3` is for ordering a mocha, `case 0` is for exiting this interface, and a `default` case is when the user

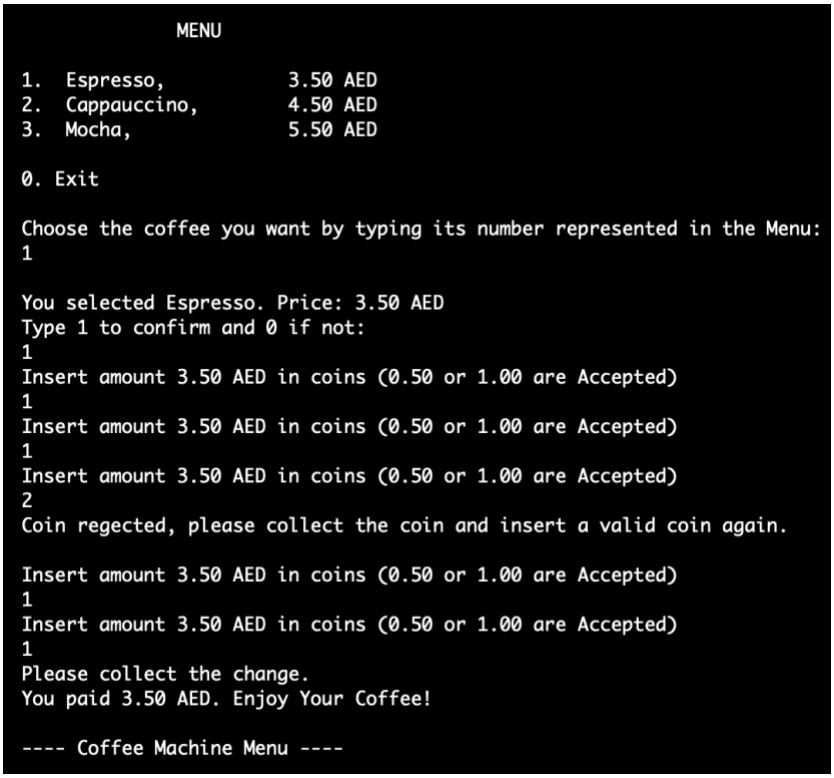
inputs the wrong number which loops the coffee menu interface until a valid number is inputted. The ordering of the coffee and the interface display occurs only when the quantities of the ingredients are above the minimum threshold amount, if every ingredient is below every threshold, then no coffee can be served, and the admin must be called – a message is displayed to inform the user of the coffee machine's status. There is a `while` loop for the ordering interface that has the conditions `x=0, x<3` and `x>0`. This is because the `while` loop will need to be exited in certain cases, such as for the exit case, where `x` will be rewritten as `-1` for the loop not to run again. It is also for when the coffee interface needs to be looped again when a wrong number is inputted by the user – this is because the loop will return to the main menu interface when the wrong number is inputted; hence, to loop the coffee ordering interface, `x` will be rewritten as `1`.

When ordering the coffee, there is a nested `if` statement in the `switch`. The first condition is when there is a sufficient amount of the ingredients, and the second is when there is no sufficient amount of the ingredients; hence, the user is informed and is asked to order again (the coffee menu will be looped again due to the condition). When there is a sufficient amount of ingredients, primarily, the price variable stores the price of the chosen option, then the user is asked to confirm their chosen option with its respective price showcased, using 1 and 0 as yes and no, respectively. If it is a no, then the coffee interface is showcased to choose a different option. When it is a yes, the program adds the money that the user will pay to the `total_amount` variable, which is the variable that keeps track of the amount of money in the machine. Then, the user goes through the payment check process, in which there is a function call to call for the payment check function, and lastly, the amount of the ingredients in the machine are updated by calling the function for updating ingredients.

The payment check function declares two variables, which are the local variables for that function: the paid amount variable and a coin variable, the paid amount variable is a variable where the coin inputted by the user - either a 1AED coin or a 0.5AED coin – so the payment will be looped until the price is met which is when the paid amount is equal to the price. The `while` loop is used with the condition that the `paidAmount` is less than the `price`. In the `while` loop, there are nested `if...else` statements. The user can only input the two types of coins, if a coin is a defective coin, the program will give the coin back to the user and display to try again, another condition is when they add more than the amount, it will display to the user to take the change. The final

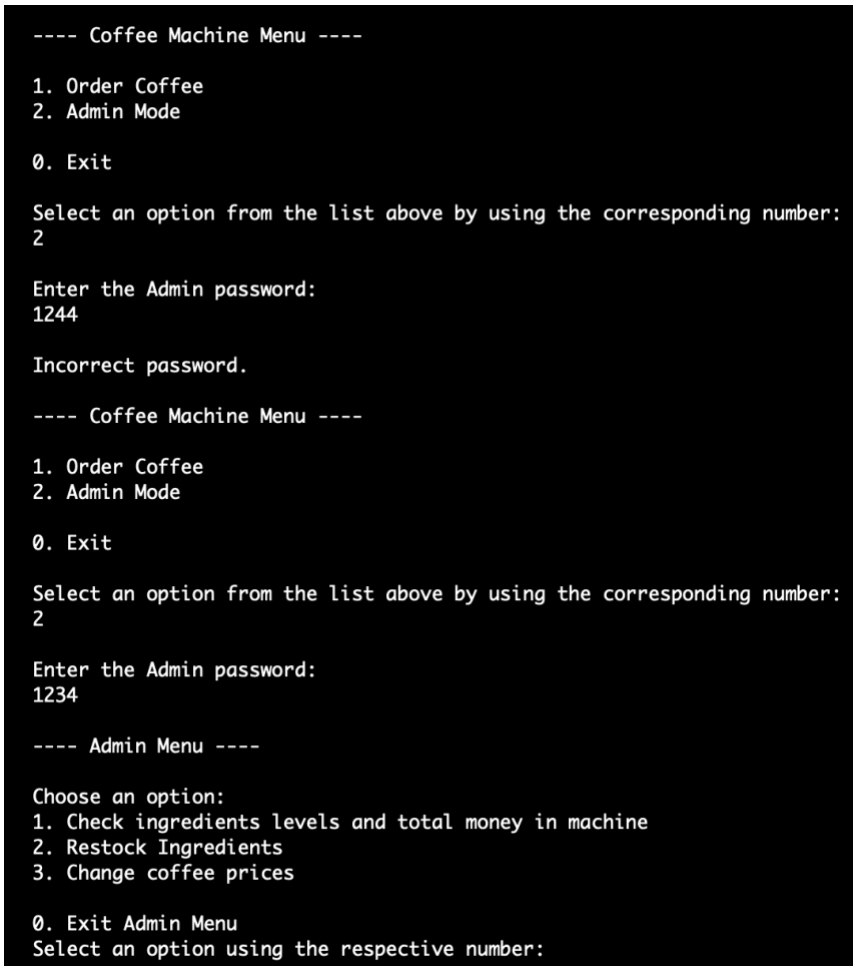
condition is when the amount due is cleared, it will print the message with how much the user has paid and to enjoy their coffee, and then the program goes to the main menu interface. The payment process can be seen in Image 3, where the user has to pay 3.50AED, with the case where the coin is not accepted and when the user paid over the amount, the change is given back.

Image 3 – Payment Process



The update the ingredient quantity function will subtract the amounts used for when the coffee is ordered – this function has input arguments, which are the ingredients, hence, the values stored in those variables will be used in this function. After the subtraction, the function has to go through a process to see if any ingredient on the list is below the minimum threshold quantity; hence, if they are, then an alert message is displayed on the coffee machine screen. To code these conditions, `if` statements are used. There is a condition in this function, which is when all the ingredients are below the minimum, and it emphasises that all the ingredients are below the minimum and not only one ingredient. At the end, the main menu interface will be displayed again to be able to replenish the ingredients.

Image 4 – Admin Mode Interface

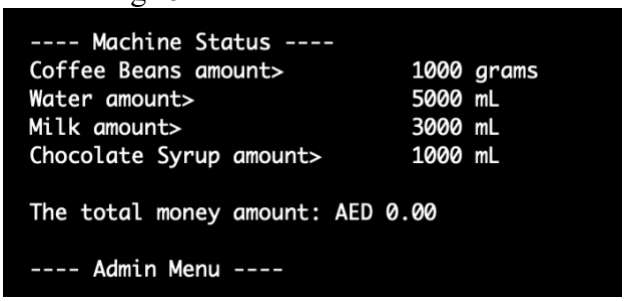


The admin mode function is called when the user inputs 2 in the main menu interface. Before entering the admin's menu, the admin password is asked, if it is incorrect, the user is displayed with a message on the incorrect password and is taken back to the main menu. When the password is correct, the admin's menu interface is displayed - which can be seen in Image 4 along with the case when an incorrect password is inputted. There are four options the admin can choose; the first option is to look at the status of the quantity of the ingredients and the amount of money in the machine where he can withdraw the money, the second option is to refill the ingredients, the third is to change the prices of the coffees, lastly is to exit the admin mode. A `switch` is used in this function as well. The `default` case is when the admin inputs the wrong number, a message displays to inform the admin that they inputted the wrong option and must try again. The admin interface will be displayed again due to the `while(1)` loop. For `case 1`, the `dispStatus` function is called, `case 2` calls to the function `refillIngre`, `case 3` calls the

`priceChange` function, and `case 0` will exit the program by using a `return` to exit the loop and go to the main menu interface, a `break` is placed for this case as to no move on to the `default` case and similarly to the other cases.

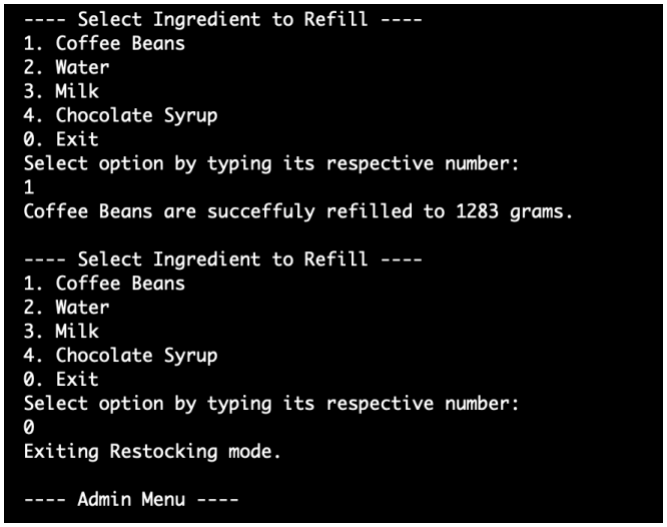
The display function for the ingredients' amount and the total sales amount has its own interface as well. It displays the quantities of each ingredient currently in the machine and the amount of money. Using an `if` statement, the program checks if the amount in the machine is not equal to zero, then it asks the admin if they wish to withdraw the money, where the admin replies with a `1` or `0` as yes or no, respectively. If no, then the interface is exited and goes back to the admin's menu interface. If yes, then the `total_amount` variable is written to zero and a display of the completion of the process is printed to the admin and goes back to the admin's menu interface. Image 5 shows the machine status interface when there is zero amount of money in the machine.

Image 5 – Machine Status Interface



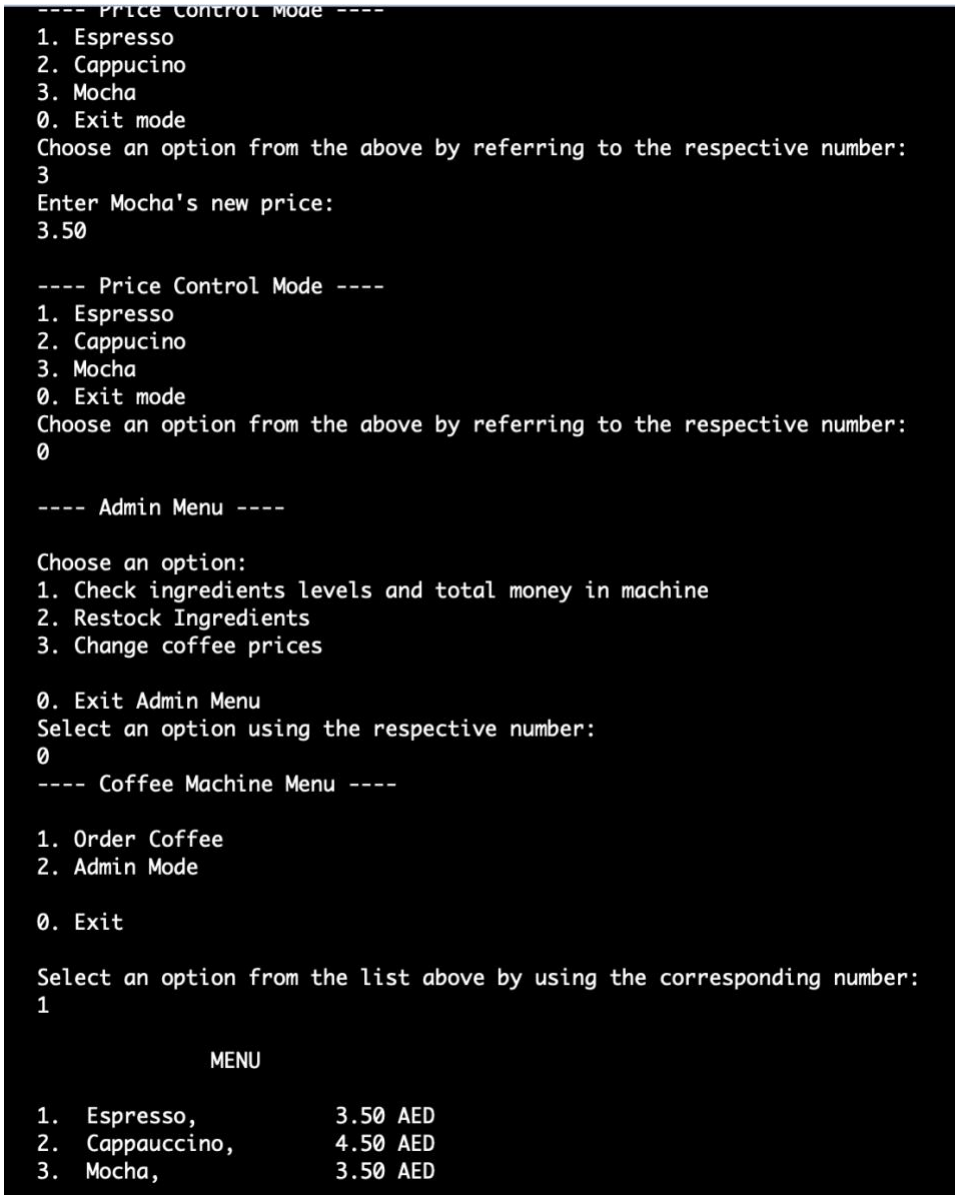
The function that refills the ingredients uses the `rand` function to generate the amount the ingredient will be refilled with. An interface is displayed for this function to ask the admin which ingredient they desire to replenish, there is also an exit option to go back to the admin's menu interface with `case 0` – a `switch` is used for this function as well. Each case represents an ingredient, where the ingredients are numbered from 1 to 4; hence, the admin has to input the respective number of the ingredient desired, if not, the incorrect number will go to the `default` case which asks the admin to enter a valid number and displays the refill ingredient interface again. For `case 1`, refills the coffee beans, `case 2` refills the water amount, `case 3` refills the milk amount, and `case 4` refills the chocolate syrup amount. The `rand` function generates the number under the intervals specified, and that number is added to the current ingredient amount. Image 6 showcases the ingredients refill interface when the admin refills the coffee beans amount and they exit this interface.

Image 6 – Ingredient Refill Interface



The final function used in this simulation is the price change function. This function also has an interface where the ingredients are listed with numbers, an exit option is given, and then the program asks the admin to choose the ingredient desired to change the price. A switch is used as well for this function, with `case 0` being the exit option and the `default` case is when the admin inputs the wrong number. The case labels refer to the numbers of the coffees; hence, `case 1` is for the espresso, `case 2` is for the cappuccino, and `case 3` is for the mocha. In each case of the coffee, the program asks the admin to enter the value of the new price, and then the code rewrites the specified price variable for that coffee with the new value. With every case option, the program goes back to the same interface unless the exit option was chosen as to be able to change the prices of the other coffee if wanted. There is a `while(1)` loops for this function to be able to loop the interface. Image 7 showcases the price change interface with the case of changing the mocha price to 3.50 and then goes back to the main menu to enter the coffee order interface to check if the price has changed.

Image 7 – Price Control Interface



- Evaluation:

For the improvement of the coffee machine simulator, more cases need to be accounted for, such as when the input is not an integer number or a number at all. It was observed that when a character is entered accidentally, the program runs forever in a loop, this will not be the case unless if it is accounted for. In C language, the `scanf` function needs a data type specifier, hence, to read a char when a number is inputted will be impossible.