GE-103
Coffee Machine

Chirag Ghodke[#1],Jarpula Uday Singh[#2], Jigyasa Singh[#3], Lokesh Jassal[#4]

[1]2021EEB1162, 2021eeb1162@iitrpr.ac.in
[2]2021EEB1179, 2021eeb1179@iitrpr.ac.in
[3]2021EEB1182, 2021eeb1182@iitrpr.ac.in
[4]2021EEB1185, 2021eeb1185@iitrpr.ac.in

***Abstract*— this document gives us an overview of the working of a commercial coffee machine. The code for the program is written in python programming language. The program is built using basic level if else statements, for and while loops, lists, and functions.**

***Keywords*— mac-coffee machine, function-fun, commercial-comm, coffee-kofi, ingredient-unit.**

I. Introduction

The 'Coffee Machine', as the name suggests, is a code written in python language for a comm kofi serving mac. It takes instructions from the customer and returns the kofi depending on the input from the customer.

II. Literature Review

On July 26, 1971 the first programmable mac was made by Edmund Abel Jr. The patent was granted on Sept 26, 1972 and the mac was named as Mr. Kofi. Since then, the mac has undergone many advancements, such as introducing a variety of kofi types, enhanced user friendliness, as well as feedback from the customer.

III. Objective

In the code, several funs and lists have been defined which work in a synchronized manner to make the code run efficiently.

A. Lists

1) *Report:* This global list contains the pre-set unit. Whenever the machine is switched on, or the resources are reset, all the unit are reset to according to this list.

2) *latte/espresso/cappuccino:* These lists contain the amount of unit required to make the specific kofi type, as well as the cost.

3) *Empty list A:* The list A is initially empty. It stores the feedback received from the customer. This list is displayed whenever the customer wishes to see the previous feedbacks.

4) *report:* This list contains the available resources: water, milk, kofi, sugar, and money. Every time an order is completed successfully, the resources are updated in this list.

B. Funs

1) *order:* Fun order () takes the order from the customer, and stores it into a string variable ‘a’, checks whether the order is valid , if valid, it stores it into a variable ‘b’ and returns it to the caller in a while loop (line 258). If the order is not valid, the machine displays the message “Sorry, we do not serve this”, and asks the user for order, until the customer gives a valid one.

2) *resource_check:* Fun resource_check () takes the type of kofi (latte, espresso, cappuccino) and the report list as parameters. It checks if the unit available in the machine are adequate for making the kofi. The availability is checked by comparing the similar indices in list report and latte/espresso/cappuccino. If the comparison is passed, the fun returns the value of ‘b’.

3) *coupon_code:* Fun coupon_code () asks the customer whether they have a coupon code. If typed yes, the machine prompts the customer to enter the coupon code. In the test case, only coupon code –‘super_20’ works and gives a 20% discount on the MRP of the kofi and returns c==20.
If customer enters any other code, the machine displays, ‘Sorry ☹ No coupon applied’. If the user enters incorrect code or typed no in the beginning, then the fun returns c==0.
If the customer enters anything other than yes or no, the fun calls itself again, until the customer has typed either a yes or a no.

4) *money_paid:* Fun money_paid () asks the customer to make cash payment. Firstly, it displays, ‘Insert cash in the following denominations’. It then takes input in the denominations of five hundred rupees, two hundred rupees, one hundred rupees, fifty rupees, twenty rupees, and ten rupees.
Then it calculates the amount by multiplying each input with its respective denomination and returns it.

5) *update_report:* Fun update_report () takes the lists report and ordered kofi type (latte/espresso/cappuccino) as parameters. It then subtracts the resources from the report list, and adds the money at index 4 in report. It then returns the updated report list.

6) *bill:* Fun bill () takes the ordered kofi type, price, and discount as parameters. It then generates a bill for the customer which displays the same along with the net amount after applying the discount.

7) *feedback:* Fun feedback () takes feedback from the customer if they want to provide a feedback about the kofi. If typed yes, the machine takes the feedback and stores it in a variable and returns the same and the feedback is saved in a list which can be accessed later, but before turning off the machine. If typed no, the variable is returned as empty.

8) *instruction:* Fun instruction () gives the customer four choices : see the unit report- type ‘report’, turn off the machine-type ‘off’, move to next order-type ‘order’, and see the past feedbacks-type ‘feedbacks’. It also give an exclusive choice to the machine owner: see the money collected. For this, the owner must enter the passcode, ‘money_collected’.
If the valid instructions are not provided, the fun calls itself again until it receives a valid one.

C. Working of Code

The lists are defined. A variable ‘m’ is initialised to 0. A while loop works on the test condition ‘m’==0. If the compiler enters the loop, the menu is printed. The fun order () is called. It takes input from the customer and returns it in a variable ‘a’. Now, according to the type of kofi asked by the customer, it calls the fun resource_check (), to check the availability of the unit required for that kofi. This fun returns the value of variable ‘b’ as 0 or 1.
If ‘b'==1, fun coupon_code () is called which gives discount only if the suitable code is applied. Then the fun money_paid () is called which takes cash input from the customer in the form of denominations and then calculates the amount from it. The calculated amount is stored in a variable ‘d’. If ‘d’ is greater than the cost of the ordered kofi (which is at the index 4 in the unit list of each kofi), the machine updates the unit list by calling the fun update_report (). The machine returns the change and the fun bill is called which prints the bill for the customer. Machine then asks for feedback from the customer. It is up to the customer whether to give the feedback or not. If the customer gives a

feedback, it is stored in the empty list ‘A’. Else the machine gets ready for the next order. However, if ‘d’ is less than the cost of the ordered kofi, the machine returns back the money and displays, ‘insufficient money’. The machine gets ready for the next order by calling the fun instruction ().
If ‘b’==0, the machine displays, ‘resources insufficient to complete the order’, and gives the customer a choice to reset the resources of the machine. The machine gets ready for the next order.
Every customer gets 3 options at the beginning: see the resources available, order the kofi, and see the previous feedbacks. The owner has 2 exclusive options: see the money collected and turn off the machine.

IV. CONCLUSIONS

A. Test Cases

1) *Machine is case insensitive:* The machine accepts the input in any case (upper, lower, mix). The code runs without any error because every input is converted to lowercase.

2) *When the customer enters an invalid order:* If the customer enters an invalid order (anything other than latte/espresso/cappuccino), the machine asks the customer to enter valid order.

3) *When resources are insufficient:* If the resources are insufficient, the machine gives the customer, an option of resetting the resources to the default values as present in the global list Report.

4) *When customer inputs less cash than the amount of kofi:* If the customer inputs less money than required, the machine returns all the cash back to the customer and says insufficient money.

5) *When the customer enters invalid/wrong coupon code:* If the customer enters invalid/wrong coupon code, the machine doesn’t apply any discount and proceeds with the default amount.

6) *Access to the money collected:* Only the owner of the machine may be able to see how much money is collected in the machine.

7) *Turning off the machine:* Only the owner of the machine may be able to turn off the machine, not any customer.

8) *Invalid instruction :* when an invalid instruction is given, it again asks for instruction until valid one

The code works efficiently for each one of them. The boundary conditions have been taken into account during the writing of the code. In a practical scenario, the machine would work flawlessly.

ACKNOWLEDGMENT

Towards the end of this project, we, team Error404 would like to show our gratitude towards our course coordinator-Sudarshan Iyengar sir as well as our mentor-Roshan Kumar Singh sir, for letting us work on this project and also for their timely guidance that led this project reach the conclusion. Though the journey wasn’t easy, but the enthusiasm and cooperation among the team members helped us overcome all the hurdles and to give their best in this project, both personally and as a team.