**Document explaining code for CA2 OOP Adlane Boulmelh (C19367031) 17/12/2020**

**Classes and Methods**

This report will be on the Continuous Assessment 2 which is design an application to manage a shopping cart.

Designing this project we needed to create two classes ShopCart() and Customer(). These were my two classes. Along with that we needed to sub classes which are called LoyalCust(Customer) and BargainCust(Customer). These are my two sub classes located under the class Customer.

Starting in the class ShopCart I have 4 items declared in \_\_init\_\_ these were basket\_list, exclusive\_items, normal\_items, total\_bask.

Basket\_list would help store and keep all items added by the user in a list.

Exclusive items and normal items is a dictionary that stored available products for the customer. Total\_bask was used to store the total cost of all items added in to the basket.

This class contained four methods which were used to add a product to basket, remove a product from the basket display the current basket of the user and to calculate the total cost of the current basket.

The add\_prod method would ask the user for a string input this would then be stored in a variable. Then I would get the keys and values of the exclusive or normal items depending on whether the customer is a loyal customer or a bargain hunter. If the entry matches a key this product and its value will be appended to the list. Error checking is implemented if entry is not found.

I also lowercased each entry to prevent any errors arising with a capital letter etc.

The next method is remove\_prod which is exactly the same as add\_prod except this time it removes the entry instead of using append I used remove().

The next method display\_basket prints out the basket and its contents. It checks if nothing is in the basket as this is not allowed. If this is the case it returns the user to the main menu and prints an error message telling the user to add a product before trying to display a basket.

Next will be Calc\_basket method which will calculate the total basket for the user. It will do this by appending the integer values from the list to a temp list then using the sum() function we will calculate the sum of the list and store in self.total\_bask. If the basket totals €0 then an error will be displayed saying you will need to add something to your cart to checkout. If value is greater than € then user will be asked if they want to do a home delivery option or click and collect. If delivery is selected a charge of 5.99 is applied and this is stated. If c & c is selected then user will be asked if they are happy with their basket if not they can select n or no to go back to the main menu.

I used two operator overloads the \_\_gt\_\_ and \_\_lt\_\_.With these I figured out which product list has the most expensive item and which has the least expensive. I imported from heapq import nlargest and nsmallest.

nlargest is a great function to find the largest value or values within a dictionary. Also nsmallest does the same but finds the smallest value or values. This was done and a comparison was done using the operator overloads.

Customer class contained a simple input from user to store first and second name using method cust\_info.

Subclasses LoyalCust and BargainCust printed out the dedicated items available whether the user is loyal or a bargain hunter.

I designed a main function which was used to navigate through the menu. It included all necessary error checking within the program. Main() function took input from user and if the input matches any available options the specified classes would be run for the user along with the specific methods.

I also designed a test function with some instances in it. I wasn’t too sure what you really wanted in this test function so I tried to match what I could. A pre menu option is available if you want to test or use menu interface.

I designed a function error\_print which was used whenever I needed to print an error message in the program. This helps keep the code clean and looks better as I just reused this function when I needed.

**User Manual**

A pre menu is available with two options 1 and 2. 1 to access test function and 2 to access menu interface as normal.

Through normal menu interface you have 5 options.1 Create a customer, 2 list products , 3 add/remove a product, 4 list basket, 5 checkout.

You must create a customer first to proceed to add/remove a product.

You must have something in your basket to display the basket.

Value of the basket must be greater than €0 to checkout.

**Difficulties**

I didn’t face much difficulties in this except for figuring out what operator overloads to use exactly. Other than that it was straight forward enough to complete. Also, I faced some difficulties with the test function I did not know how to implement it properly and what to include in it as it was vague. The most challenging part was figuring out the best way to implement everything.