The input for the program is different from what you are accustomed to seeing. Rather than prompting for each individual value, it prompts for several values at once. What benefit does this provide? How does it complicate the reading of the input? How were you able to handle the commas separating the input values?

Firstly, this type of user input helps in better user experience. We do not have to ask the user for 5 different inputs of same product. Rather, ask 1 input that are related to each other in one run so from the user perspective the program is more intuitive and user friendly. So, for a better user experience, we are taking the input pertaining to one product in one run and are storing their values into their respective variables.

Secondly, this type of user input, helps in assigning, a particular products’ values like amount, category, price, byWeight and name in one run, when we take several inputs at once. Since the data pertaining to each product is assigned right away, we can evaluate and validate these inputs more efficiently just like we do in this project where, whenever a particular data of a product is entered, we right away validate whether it is correct input. If it is not, then continue to ask for the user input until they enter correct values.

Now, reading this type of data where, all the values are being read in one input is complicated. For that we need field delimiters. So, we inform the user to enter each value to be separated by commas. This way, we as programmer can separate the values from one single input when field delimiters are present, easily. All we must do is come up with a way to separate the values entered and store them into their respective variables.

Handling the input, separated by field delimiters was easy for me. My approach was to store the whole input as a string and then parse that string which is already separated by commas into their respective variables. To obtain the input separated by the commas, I first used fgets that takes the string as first argument, total length allowed for the string as second argument and pointer to a file object that identifies the stream where the characters are to be read from, for example stdin or a pointer to a file as its third argument. Once the string is stored, I then used sscanf function which takes the string identifier as its first parameter, the format of how the parsed input is to be read from the string identifier as second argument and addresses of the variables as its third argument. This way I was able to obtain and store the inputs individually into their respective variables, excluding the field delimiters.

In a nutshell, this program provides a better user experience and is efficient and effective to manage the data from the programmers’ point of view too.

Finding the top sellers in each category requires the development of an algorithm. Describe how you went about developing this algorithm. Then, describe how your algorithm works.

As seen in the output for Top 3 sellers in each category, its primary need is in which ‘category’ which 3 ‘products’ had the highest sales in ‘quantity’. I right away knew that I had to obtain at least this three information in order it print the top 3 sellers. To have this information available for me, I made a local array of structure called as SaleStock that is of size MAX\_ITEMS\_IN\_SALE that has, three variables, namely, quantity, product name and category. As the readSale function was executed multiple times and the sales were made throughout the program, I populated the three variables of the local structure which would help me further on in the program, when I need the quantities sold for each product. Once I know the quantities and for each product sold, I started designing the getTopSellers function. Here I created another local temporary structure called SalesProductTemp that would help me copy the data from SaleStock to SalesProductTemp, category wise. I could have used original local structure, SaleStock, however the data stored in this array of structure was not consistent for example, index 0,1&2 had products of produce category, then index 3 was a product for meat category and index 4 was baking and so on. I had to sort this array category wise, for which I used one of the objects of SalesProductTemp structure. I ran a for loop until the number of items sold during the whole period of sales, and made a condition to compare, whenever variable cat’s value, for example 1 is for produce was matched with the SaleStock[i]’s category list, I copied 2 contents from SaleStock, namely, the product name and quantity into to SalesProductTemp’s object so that I can have a sorted order of categories according to the requirements, which is, in order, produce, bakery, meat, dairy, baking, house wares and miscellaneous. Once the categories are sorted accordingly and quantities and product names are assigned at their respective index, I had to come up with a sorting algorithm.

This program requires a descending order sorting of products, for which I used bubble sort algorithm. I ran a nested loop, where the inner loop swaps the values if the value at previous index is lower than the value at next index, then their placeholder is exchanged, and the outer loop makes sure to swap it repeatedly until the condition is true. Within the if condition of inner for loop , I used a temporary object of structure, SalesProductTemp to hold the values temporarily as the sorting happened in descending order. Also, once the values were sorted, After the inner for loop ended, every time I assigned the respective product name who was the highest seller and the sorted quantity according to the highest sales to SalesRecords’ object, topsellers in a descending order, at the index starting from 0. As sorting happened during run time, the values of the topSellers array were automatically populated in the order of top 3 highest seller and the array was ready to print the result in sorted order in printTopSellers function as required. As the values were printed, every time the getTopSellers was called the values were overwritten at the whole topSellers array since its max size is 5, and we only need to print top 3 elements out of those 5 elements. The cat variable from main file helped me print the values category wise as required by the program requirements.

For each function you **personally** wrote, describe what it does and why it deserves to be a function. Be sure to include the functions you added which were not called directly from main.

Function: readStockItems()

This function receives 3 formal parameters, 1 array parameter of type StockRecord and 2 constants defined in main source file. The primary goal of this function is to read the initial stock from a file. It uses a pointer of type file and opens the file for read more, reads the necessary data, stores them into the respective the variables and closes the file. It uses the function readInput to read the input either from stdin or a file. Once readInput returns the data, it checks whether a 0 was returned. If it was then 0 will be returned to the caller of the function, otherwise the control will go through the if statement. Once inside the if condition, it parses the string into each individual variable separated by commas. Then, the category and soldByWeight variables are checked for validation, and the code makes sure that correct inputs were made before the control moves towards the end of the function. It returns a value of number of items in initial stock, before giving the control back to the caller of the function. This function is very helpful and important to read the data for initial stock in the inventory.

Function: catString()

This function receives an integer named category as its formal parameter. It uses a switch case to return a string matched with the number passed. Since the category is always received as an integer value, I use this function whenever I need to find the category related to the integer passed. The function checks which number corresponds to which category and a string with the value of a category name is returned to the caller of the function. This function helps me in numerous places throughout the code. Whenever I need to know a category of a certain product, I passed an category number associated with it and the function returned me the category as a string, to be used in the function where catString function was called.

Function: printStockReport()

The function has 2 formal parameters; 1, an array parameter of type StockRecord and 2, an integer which contains a value of number of items in stock. It uses a for loop and runs the loop until it reaches to the number of stock items in the inventory. Its main condition within the for loop is to check whether the qty is less or greater than 0. If it is greater than 0 it will print it will print the original quantity stored at the storeStock array otherwise it will print 0 for no items present of that specific product. This function helps in printing the opening and closing stock of the inventory, which is done twice in the program and provides a highly modular solution.

Function: readSaleData()

This function has 7 formal parameters. 4 of them are pointers, whereas 3 of them are array parameters of 3 different types. It assigns the values like product ID, quantities sold, updates the existing quantity, copies the product name, and assigns a category string to their respective members of the structures. Its primary functionality is to assign values obtained in readSale function into the objects of structures that are used throughout the program. It helps with clarity and conciseness of the readSale function.

Function: printSalesReport()

This function has 3 formal parameters. 2 of them being an object of 2 different structures and the third parameter has a value of total number of items sold. The primary goal of this function is to calculate the updated price of each product sold in quantity, total cost of the sale along with tax if applicable and print all this information out on terminal. This function is called at least twice in main source file and both the times, the data passed into this function is different and it handles those data well. For example, only categories, 6 and 7 needs tax assigned to it and printed out on console screen, and this function provides a solution for that problem as well. Also, it uses the flagging concept. For example, if a certain product from saleItems array is already displayed in any given run it will update the flag and thus the control will not go inside the if condition and it will not print details for that specific category again and continues with next iteration. Apart from that, it also returns a key information to the caller of the function, which is total price, which helps in finding the end of day summary like cost of items sold, number of sales and average items per sale as seen in main function. This function is one of the key functions and backbone of this program as it helps in calculating and printing a lot of useful information.

Function: getTopSellers()

This function receives 5 arguments as its formal parameters. 2 of them being an array of structure types and three of them being integers. The primary goal of this function is to return an updated array for top 3 sellers in each category. It needs an implementation of a search and sort algorithm which is efficiently done as seen at during run time. The functionality of this function is explained in question 3 of this reflection which is about designing an algorithm for top 3 sellers and explaining how the algorithm works. This function is yet another example of modular programming and provides a highly cohesive and coupling solution.