**Report**



**Grocery Management System**



**Project Id: 4**

Student Name: Graahil Rastogi

University Roll Number: 2025200

Class Roll Number: 34

Section: L



1. **Problem Description……………………………………………………………3**
   1. **Flow diagram………………………………………………………………….4**
2. **Modules for Project Implementation**………………………………………..5

**2.1. Module description……………………………………………………………5**

* 1. **Platform Used…………………………………………………………………8**

1. **Screenshots of Project Output…………………………………………..8**
2. **Conclusion and Future Scope……………………………………………12**
3. **References…………………………………………………………………12**

* **Problem Statement:**

Grocery Management System (GMS) for Departmental Stores

The Grocery Management System (GMS) is a software application designed to assist departmental stores in efficiently managing their grocery inventory. The system aims to streamline the processes of purchasing, stocking, and selling groceries, ensuring that store owners can maintain an organized inventory and optimize their operations.

The GMS will cater to the management of 10 essential grocery items, defined by the user

By implementing the GMS, departmental stores can:

- Track inventory levels and receive alerts for low-stock items

- Manage supplier information and automate purchase orders

- Monitor sales trends and optimize product placement

- Generate reports for inventory, sales, and profits

- Improve customer satisfaction through efficient inventory management

The GMS will provide a user-friendly interface for store owners and staff to easily navigate and manage their grocery inventory, ultimately leading to increased productivity and profitability.

* **Roadmap For the Project**
* **Proposed Modules:**
* Authentication Module
* **void initAuth():** This function will ask the administrator or the store owner to register.
* **void authenticator():** This function will ask the user to enter his credentials if valid will allow program access otherwise will declare him invalid
* **void adduser():** This function will be used to add the username to the workspace, it will ask the user to input the username, then store it in the file.
* **int netvalidity(char a[]):** It will return 1 if the given string passes parameters, given by charcheck(), lenpass(), then returns 1 if passed, 0 if not.
* Utility Module
* **void purchase(int a[10][3], float b[10][2]):** This function will allow the user to update the purchased quantity of any item chooses, in a[][1]. It also asks for the cost price of the item, stores it in b[][0]. This contains a loop that will exit if user enters a designated quantity.
* **void sell(int a[10][3], float b[10][2]):**  This function will allow the user to update the quantity sold of any item in an attribute in a[][2] . Here the user will input the selling price of each item he inputted in b[][1]. This contains a loop as well which will exit once the designated quantity is inputted by the user. It will also make sure that what user inputted as sold amount, it should be less than the net stock.
* **void stock(char s[10][100], int a[10][3], float b[10][2], float c[10][2], float d[10]):** This function will call the function calculate() and then display all the arrays as a table .Where s is for the item names, a contains current stock, purchased quantity, sold quantity. The array b contains cost price, selling price. The array c contains profit, loss and profit, loss %. And d stores current inventory value.
* **void calculate(int a[10][3], float b[10][2], float c[10][2], float d[10]):** This functioncalculates profit, loss for all items, their percentage, and the current inventory value. The formulas are given:

(profit /loss) (profit /loss %) (Inventory Value)

* Data Module
* **void store():** This function will ask the user to input item names, the initial quantity of items he has/ current stock. It will store the names of items in a structure attribute, the quantity in another structure attribute, for easier access. Just like the above function it will work only the first time the application is used.
* **void buy():** Calls void purchase(), in the file with the structure, so that it can be easily accessed, in the main function.
* **void seller():** Calls void sell(), in the file with the structure, so that it can be easily accessed, in the main function.
* **void dsave():**  This funtion is save all the data in the structure using the function save, calling it with the arguments as the structure members.
* **void dated():** This function is used to access the date by the user, allow the user to store the data in the file as a table that is pretty readable.
* **void display():** Calls void calculate(), void stock(), in the file with the structure, so that it can be easily accessed, in the main function, and stocks are shown.
* **void alloc():** This function allocates memory to pointers, also calls appender, readbackup for reading of file, storing data in the structure attributes.
* **void dealloc():** This function deallocates memory from dynamically allocated structure members**.**
* File Handling Module
* **void appender(char a[10][100], int b[10][3]):** This function will append the quantity of items in a[][0] in the following way:

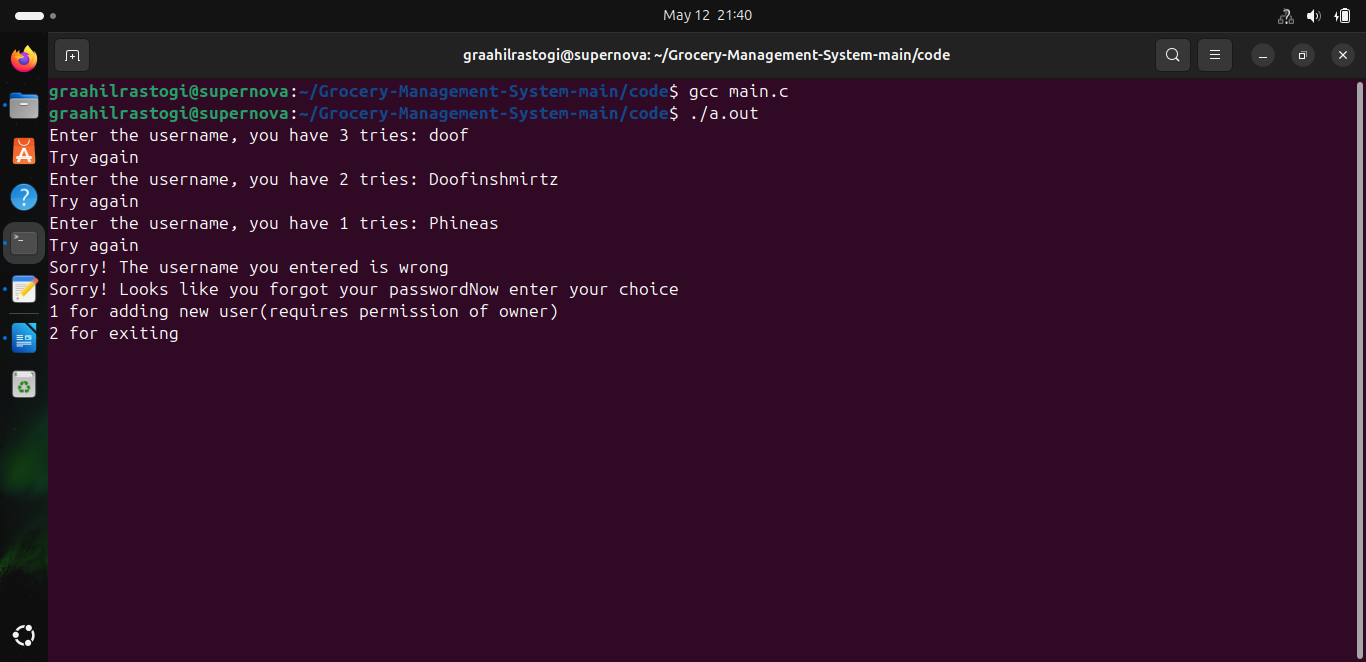
a[i][0] = current stock + Purchased amount – Sold Goods

* **void readbackup(char a[10][100], int b[10][3]):** This function will append the names int a, their quantity into b, but if there is a data.txt file already it will just append the names into a ignoring b.
* **void clean():** This function will clean the screen of the user’s console, compatible with all platform.
* **void store\_items():** This item stores data in the “itemlist.txt”, facilitates reading after making the file.
* **void save():**  This function saves all the attributes of the structure as a readable table in a .txt file.
* String Manipulation Module
* **char\* encode(char a[]):**  This function follows a simple ceaser-cipher algorithm to encode the given string, and will return a character string as well.
* **char\* decode(char a[]):** This function decodes the string a, and returns it for initial checks, it is used to decode entered password.
* **int equal(char a[], char b[]):** Returns 1 if strings are equal and 0 if they are not equal, used to compare password.
* **int lenpass(char a[]):** It is used to check whether the password entered is of length range 8-16 characters, if not returns 0.
* **int charcheck(char a[]):** Returns 1 if the given string has at least 1 special character, 1 number, 1 capital letter, and 0 if not.
* **void corrector(char a[10][100]):** This function will remove a ‘\n’ character from the string..
* Date Module
* **void dateddate():** This is a special function used to print the stock of the shopkeeper on a specific date. This module uses special techniques of reading a file and displaying the appropriate contents in the table of the given date.
* **void printdata():** This function will print the details of groceries in a particular month and year.
* String\_Former Module
* **void sTU(char a[]):**  This function is used to replace the spaces in a string with underscore, it facilitates in data reading, writing, and data representation. As many functions in C that read strings terminate at a space.
* **void uTS(char a[]):** On the other hand this function replaces underscores with a space, it makes a string more readable, and better to look at.
* **Platform Required:**

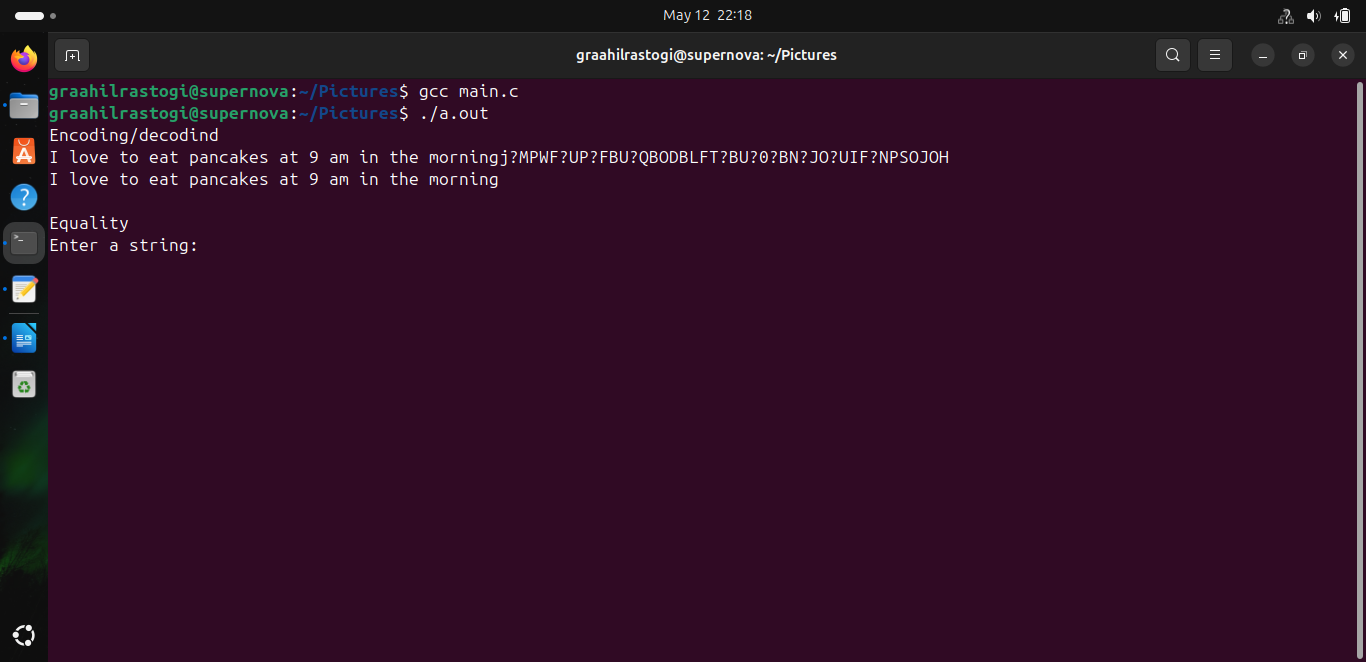
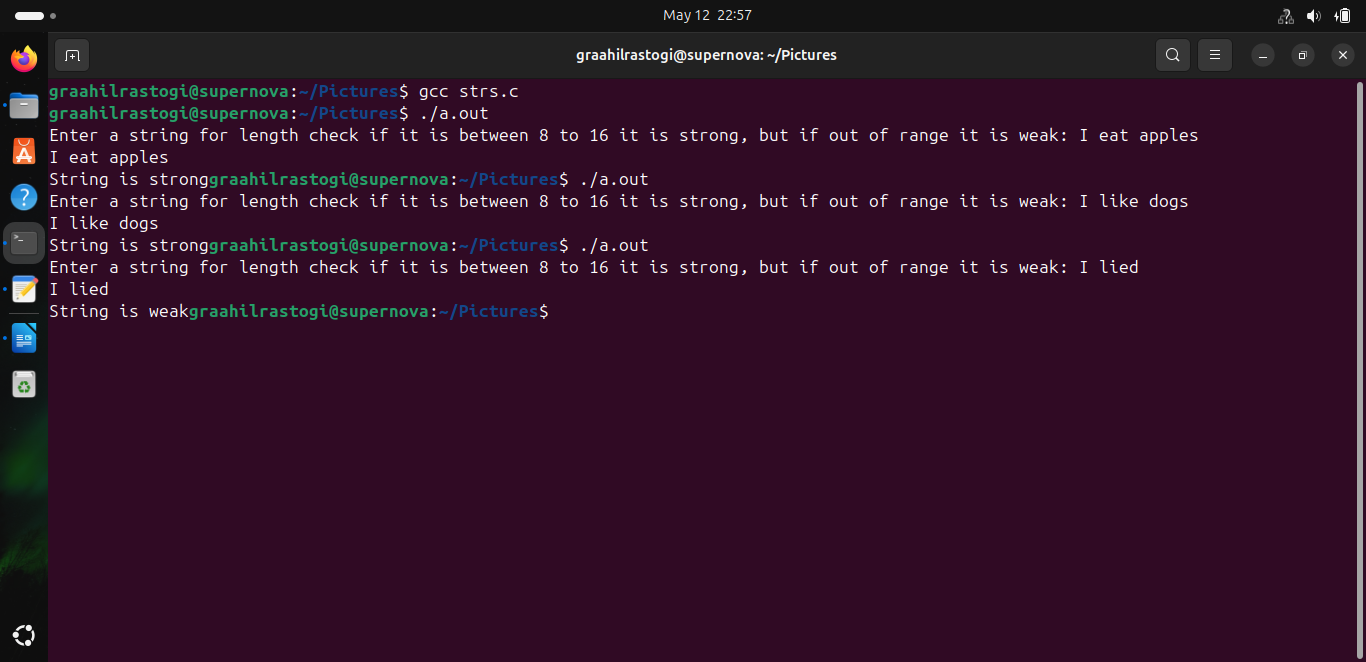
CodeBlocks IDE

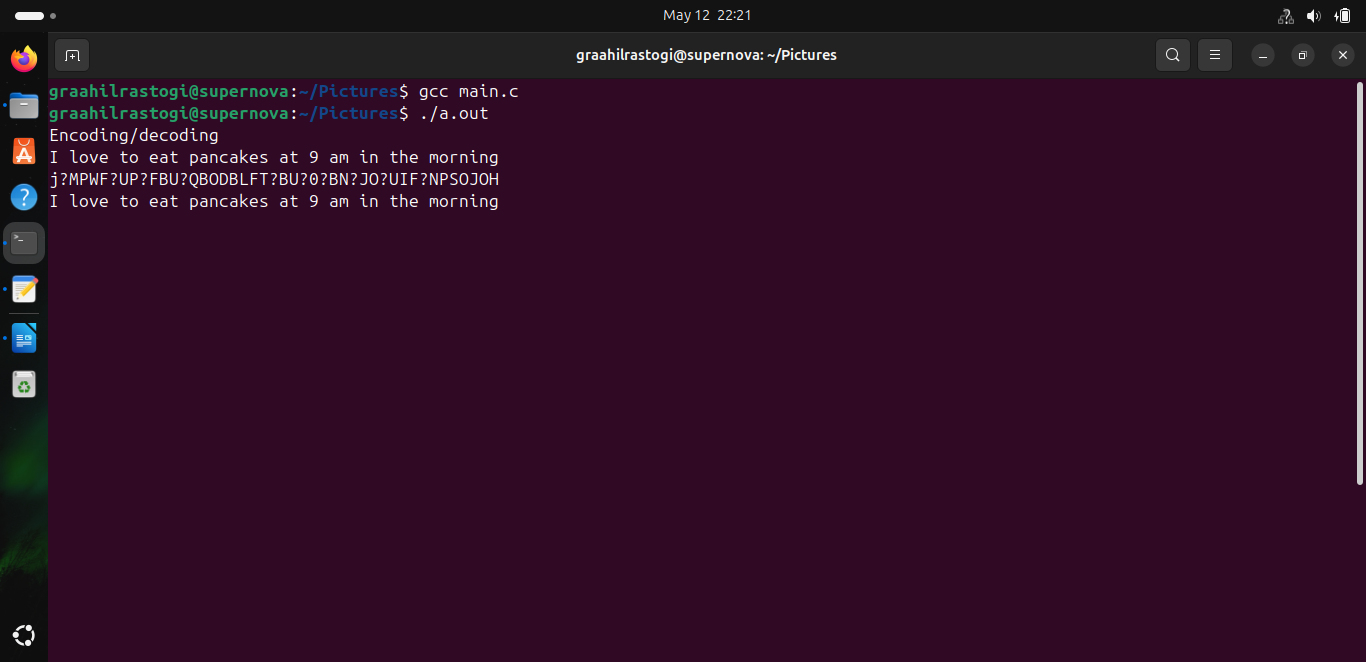
VS Code

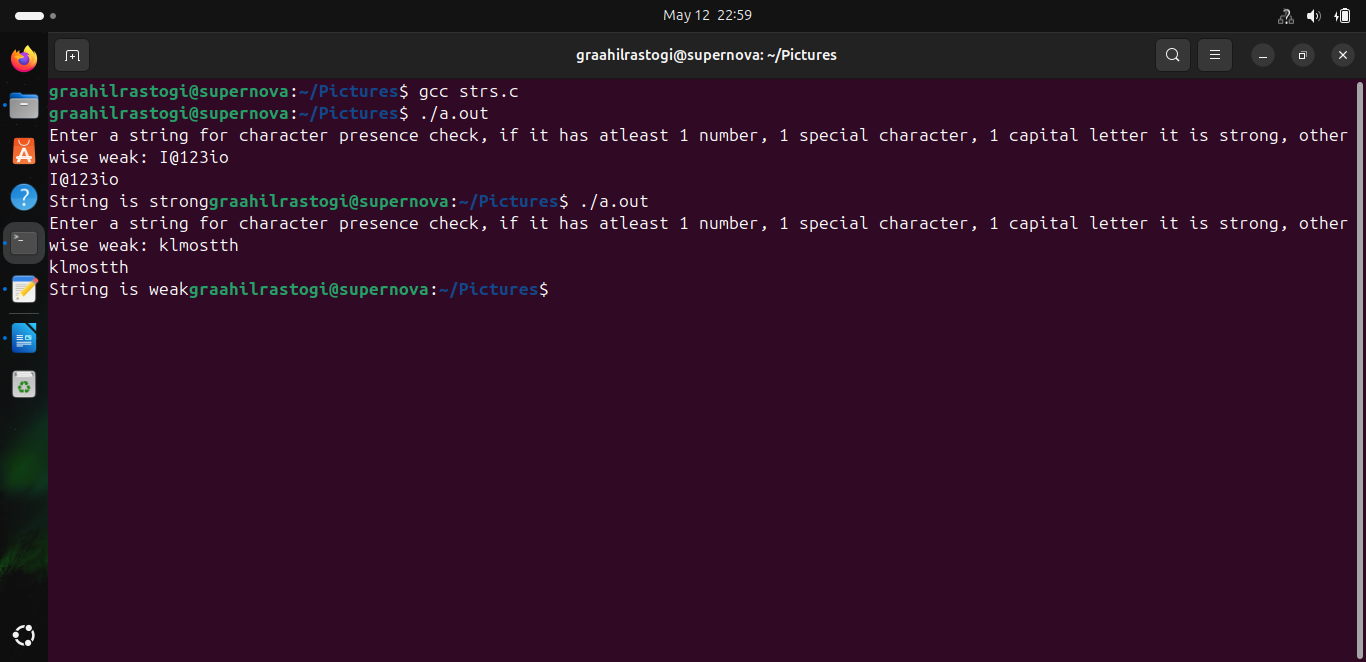
* **Screenshot of modules**
* **Auth Module**

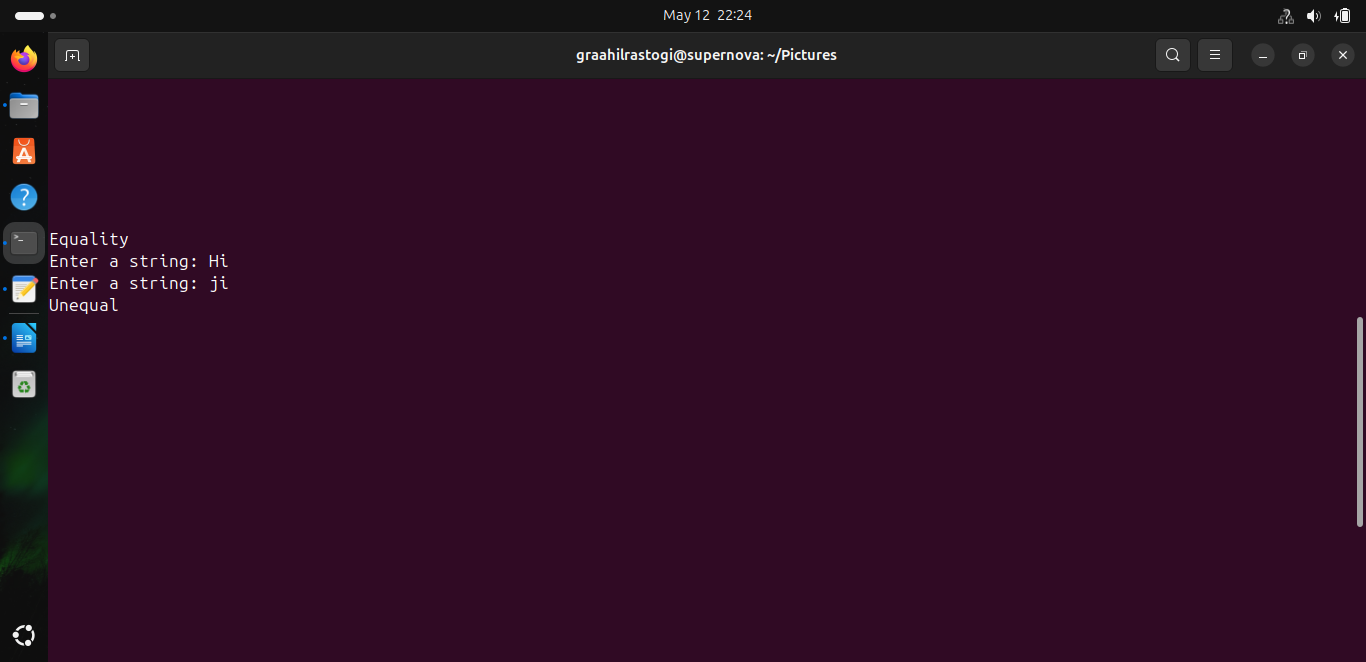


* **Strs Module**

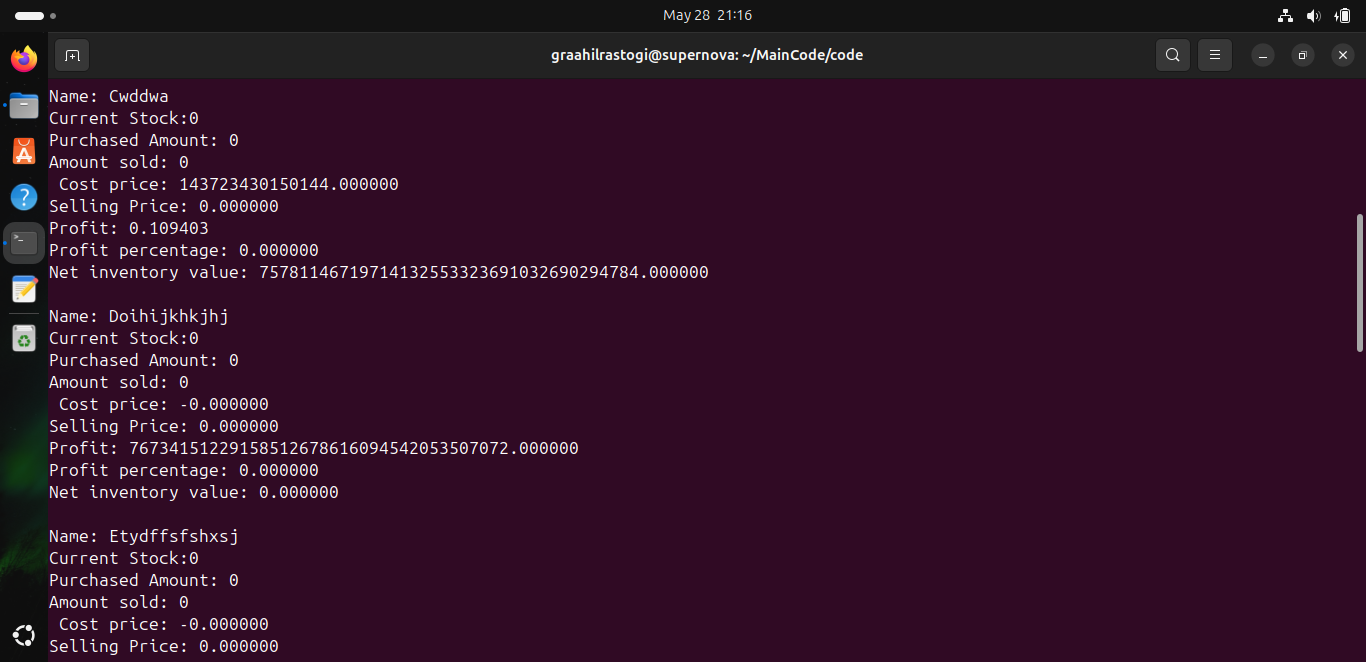
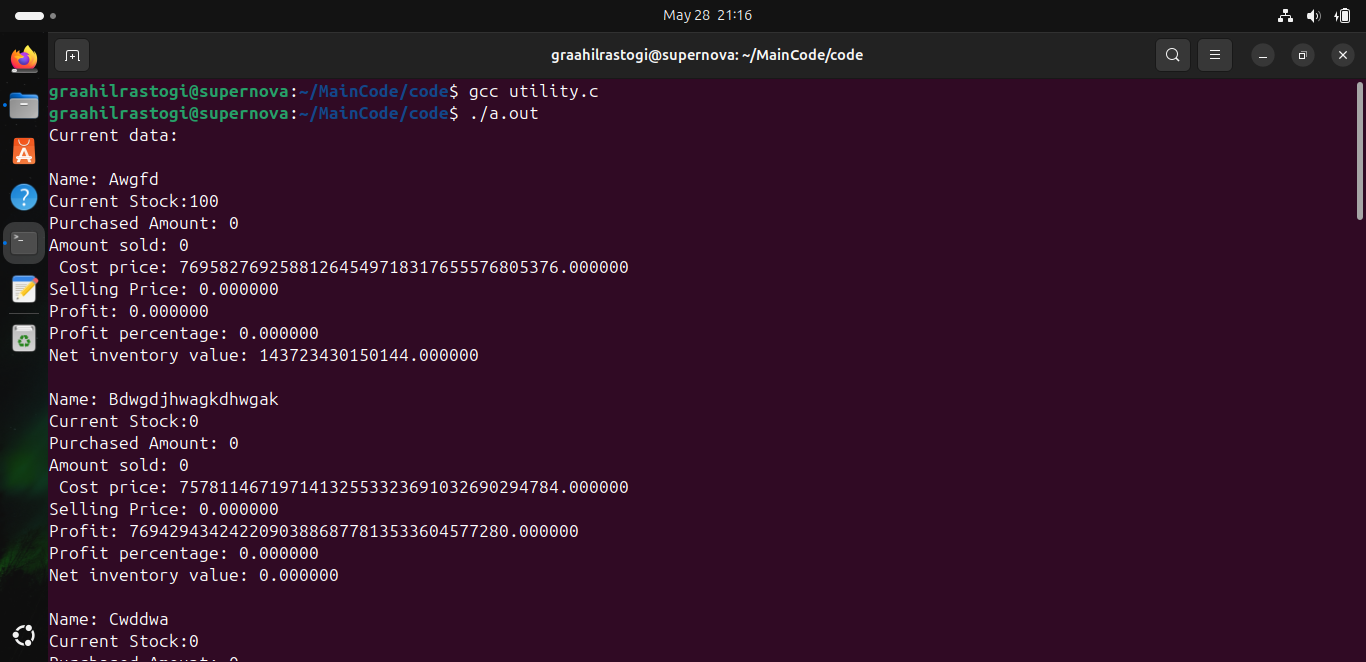
****

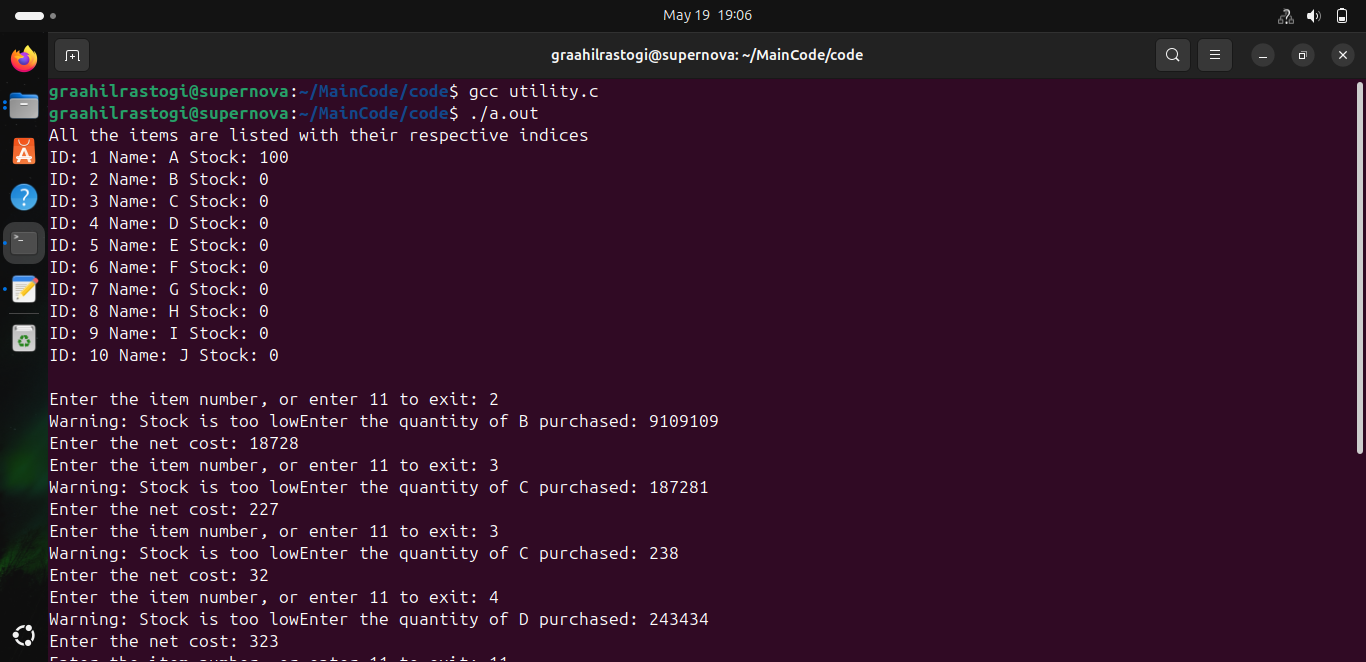


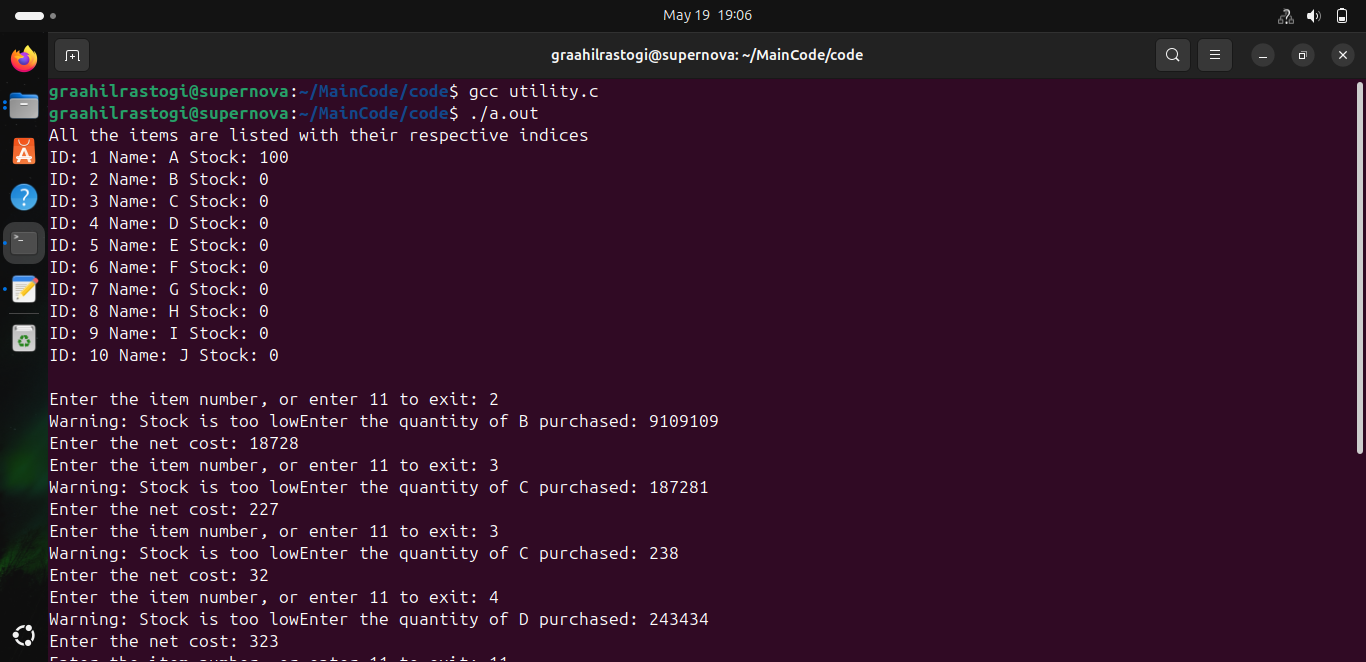




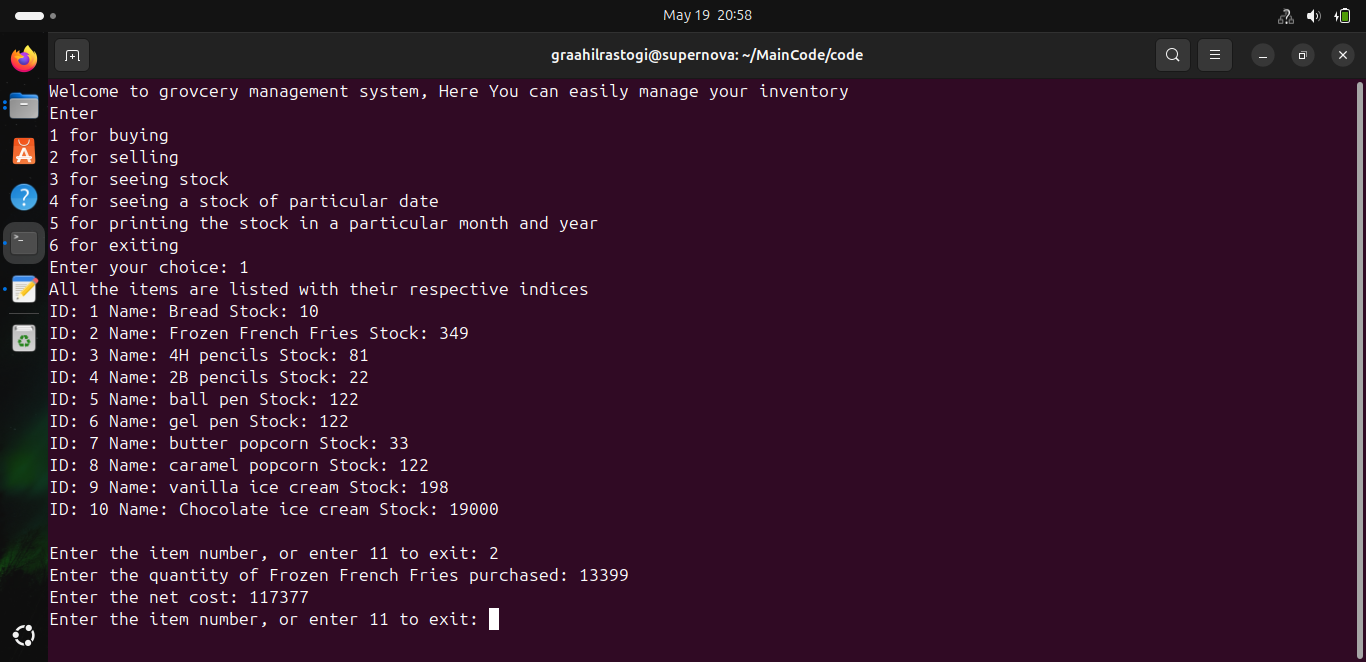
* **Utility Module**



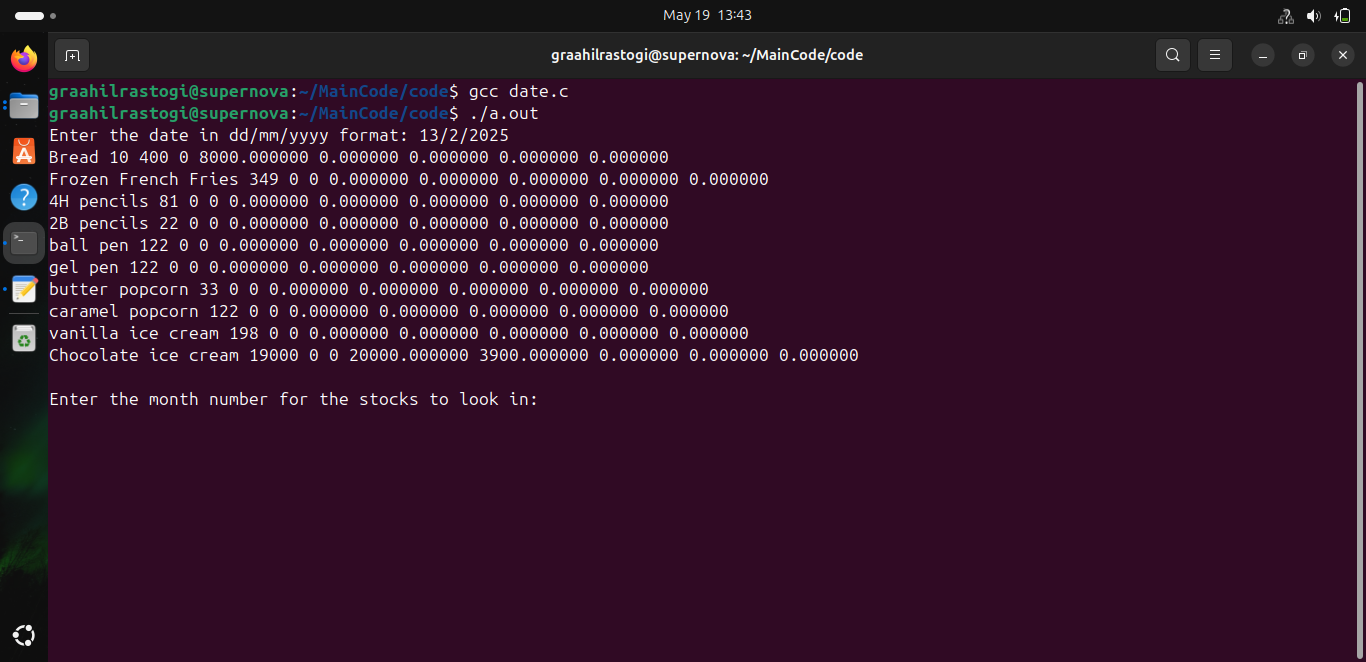
****

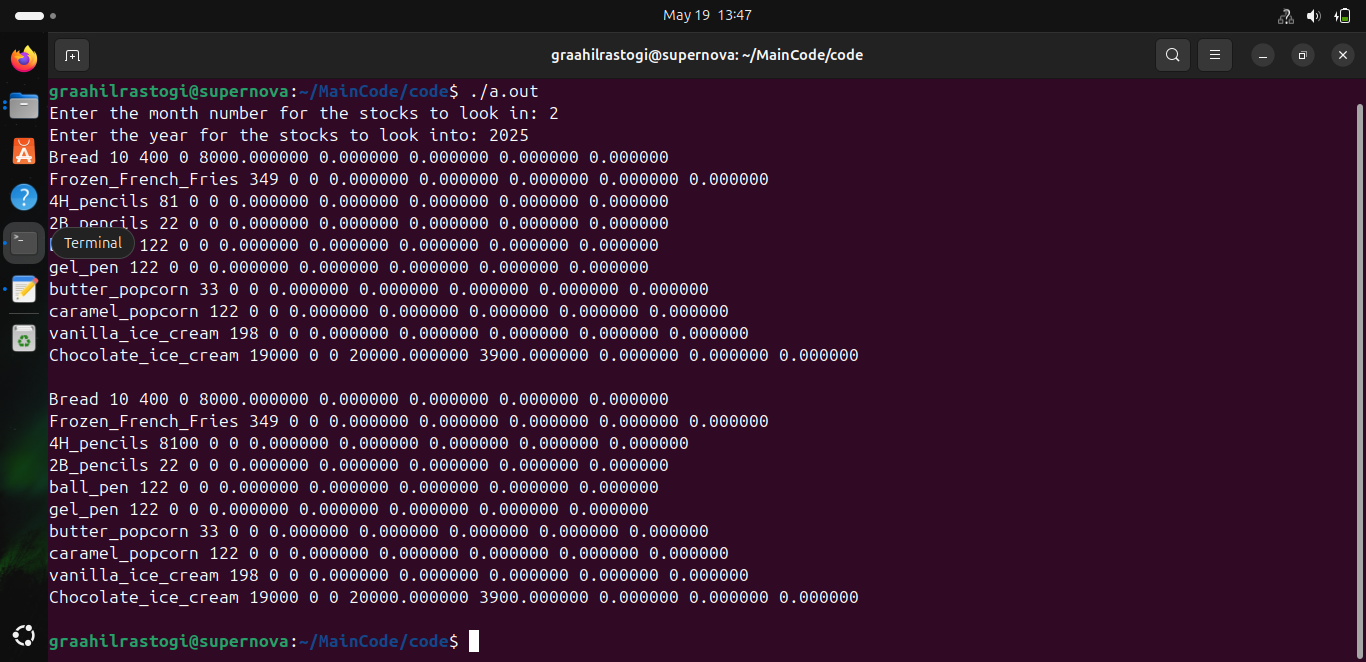


* **Data Module**

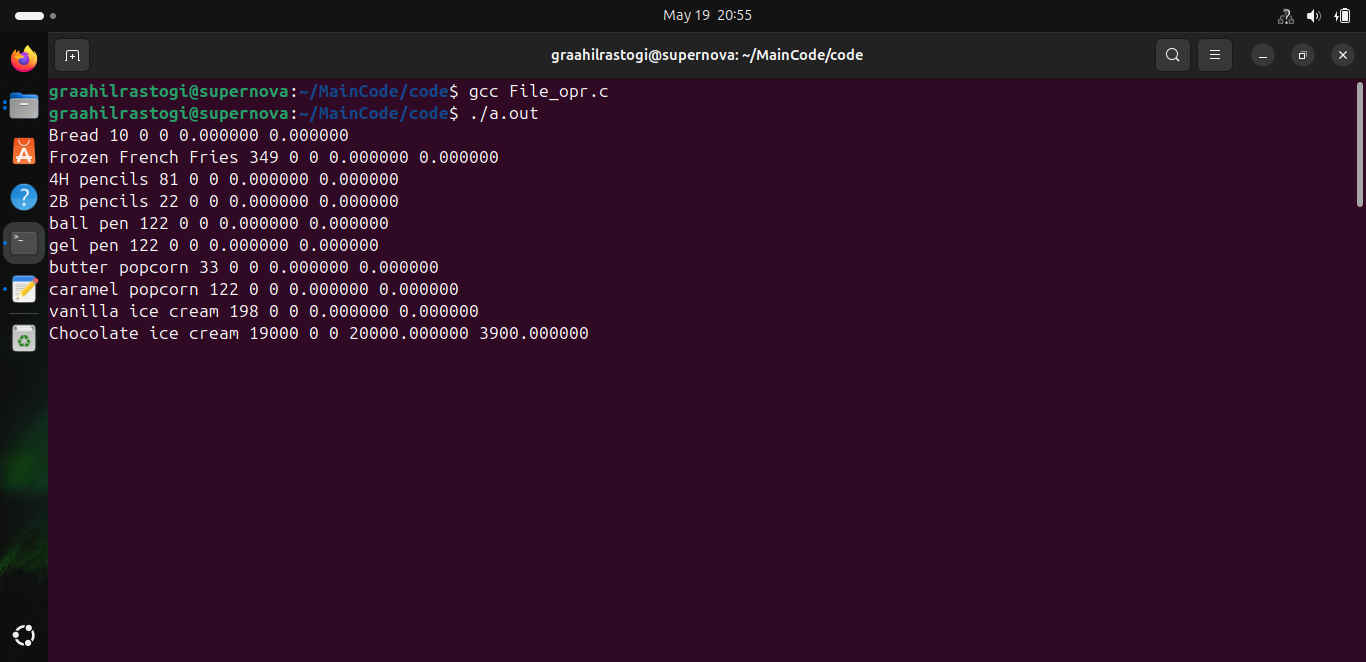


* **Date Module**

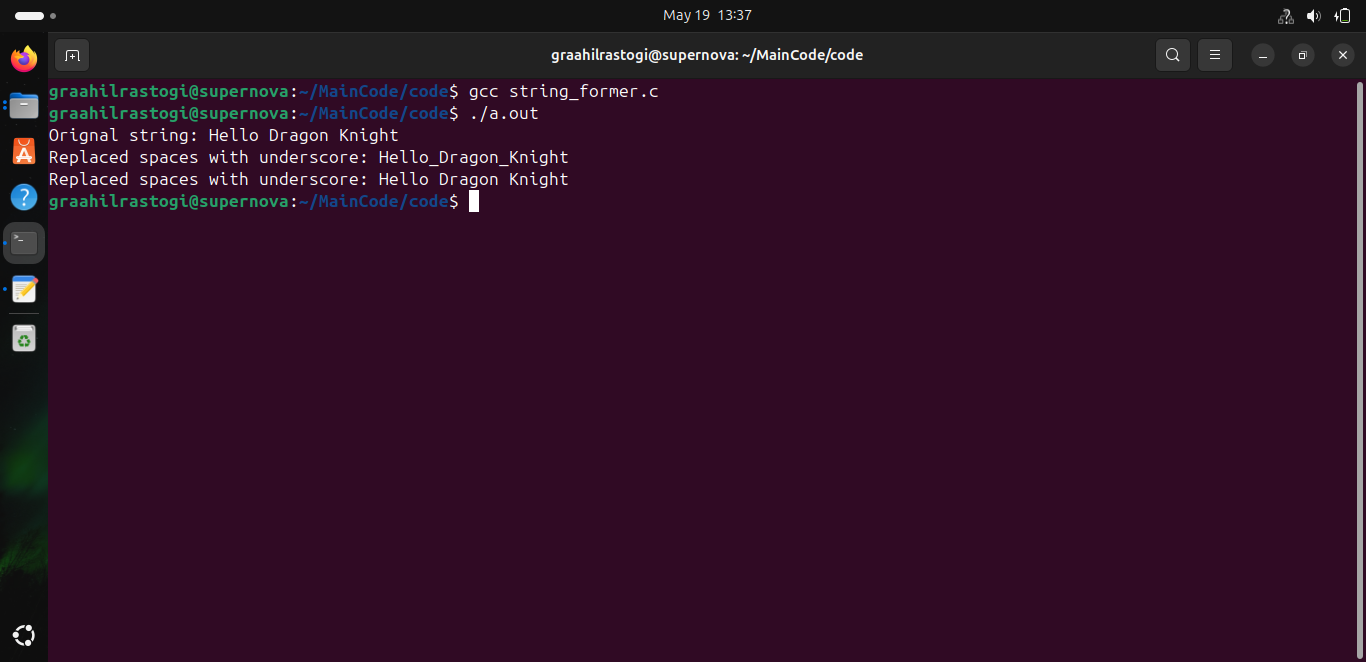




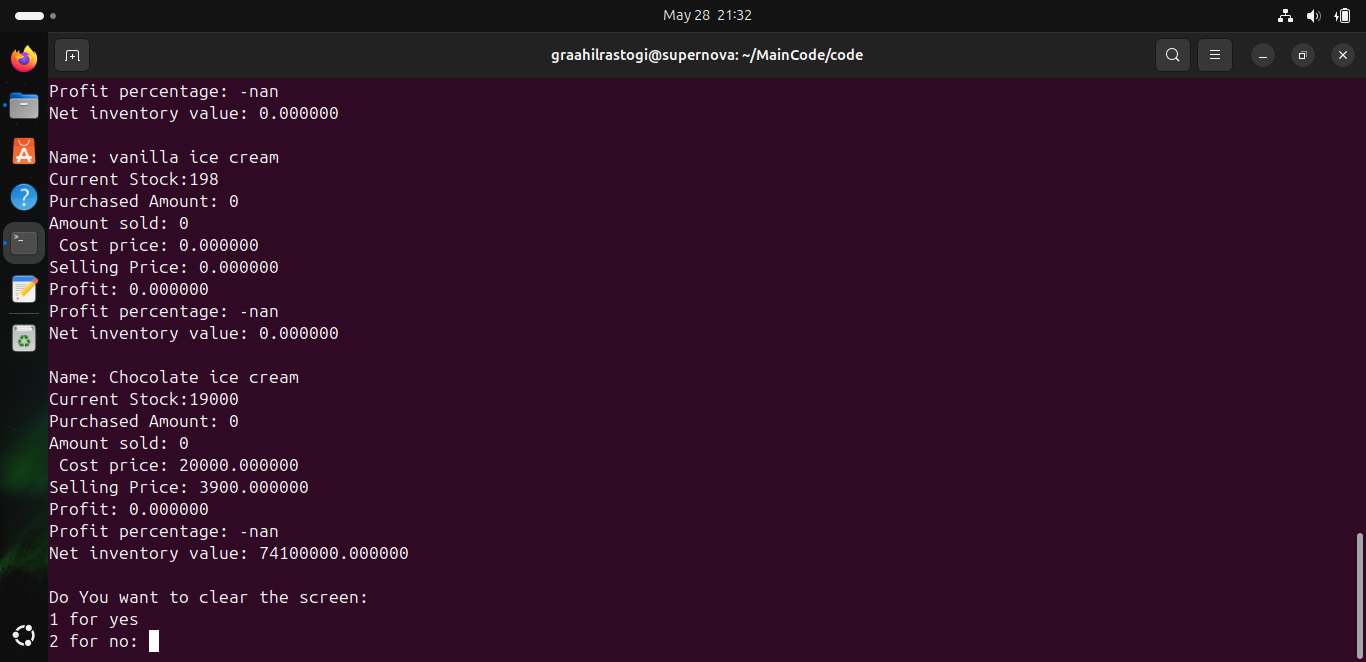
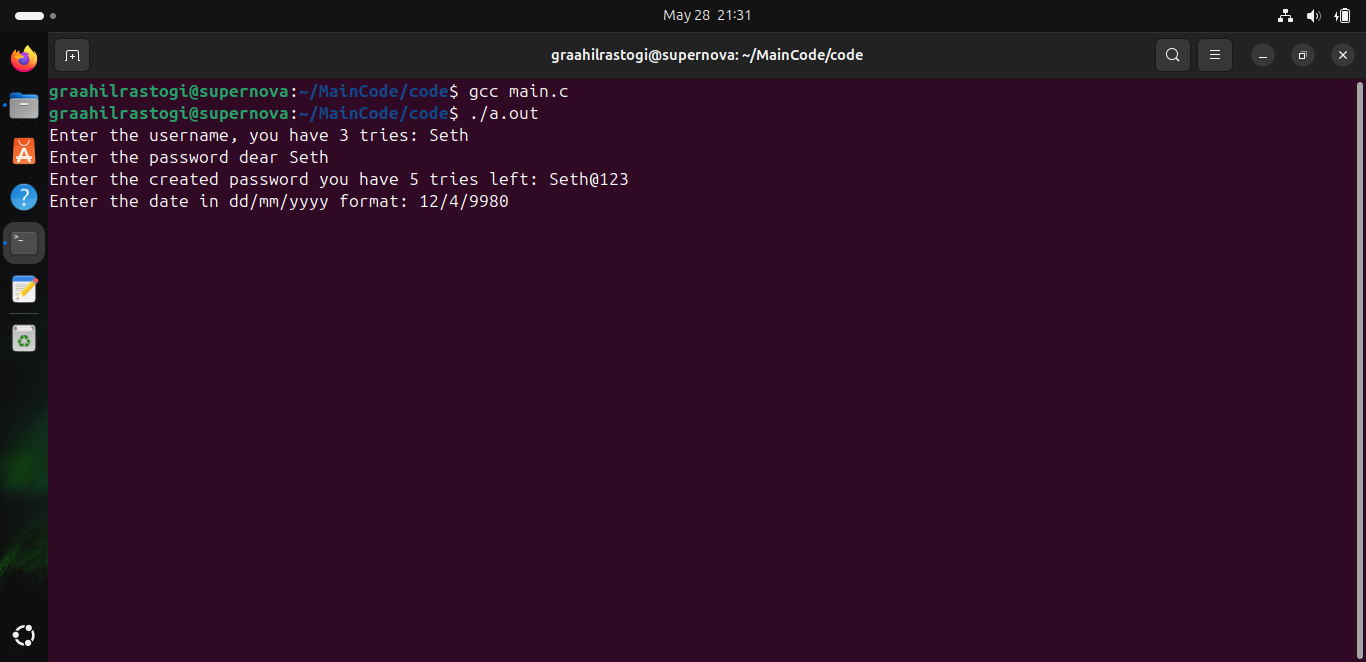
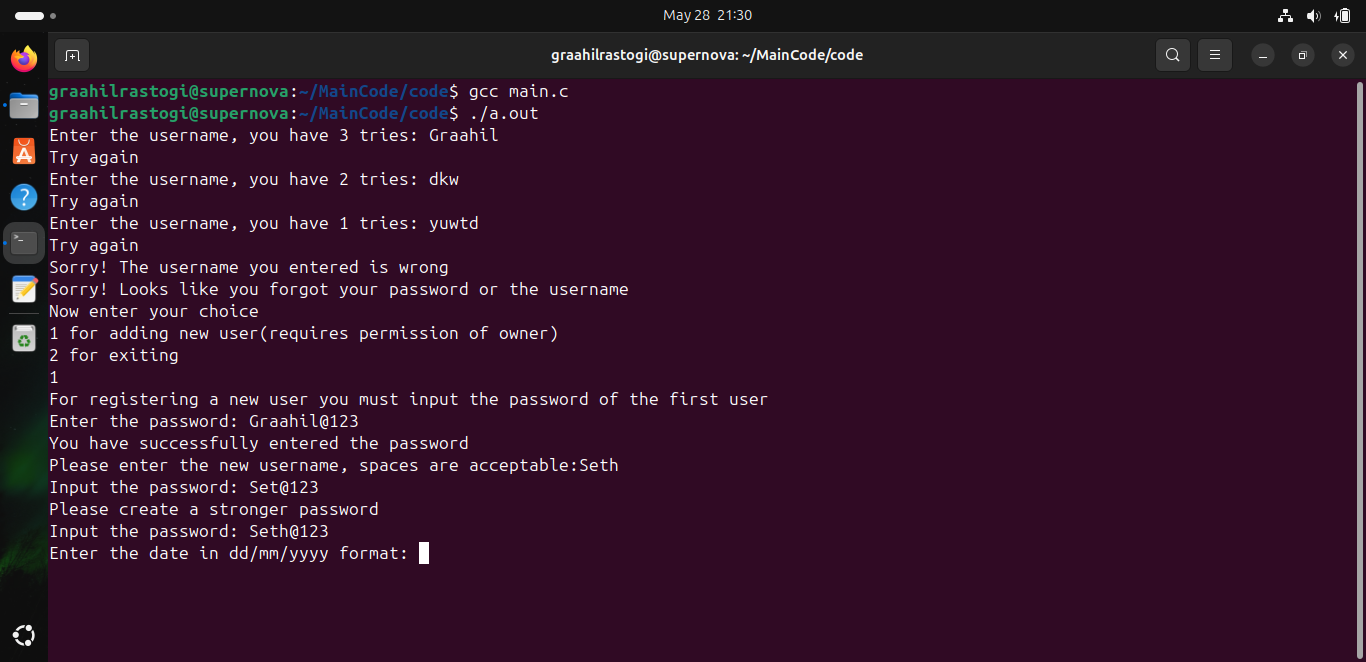
* **File\_opr Module**

****

* **String\_former Module**



* **Main Module**



* **Conclusion and Future Scope:** In this module I used various functions, to make a menu, combine functionality to my code, and perform the most basic operations required for a grocery store owner and add authentication for allowing only owner and his subordinates to use this application. This application uses various topics in C like arrays, strings, pointers, dynamic memory allocation, etc. And through this project the process of effective grocery management is achieved. However some advancements can be made like the following:
* Allow supermarket owners to assign different areas of the store to be handled by each employee where each area will contain 10 different items.
* Making separate file for each month inside a folder with the name as the year, it will make grocery management easier for looking afterwards.
* Adding functions that will be able to add graphics to the interface, for example using some special symbols print a room to the terminal representing a place for item storage, adding a special character to represent the player, etc. Ensuring to not deviate from the sole purpose of effective grocery management.
* **Books and Link Sources:**

<https://www.geeksforgeeks.org/getting-started-with-c/>

<https://stackoverflow.com/questions>

“C: A Reference Manual" by Samuel P. Harbison III and Guy L. Steele Jr

"C in a Nutshell: The Definitive Reference" by Peter Prinz and Tony Crawford