**Synopsis**



**Grocery Management System**



**Project Id: 4**

Student Name: Graahil Rastogi

University Roll Number: 2025200

Class Roll Number: 34

Section: L

* **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

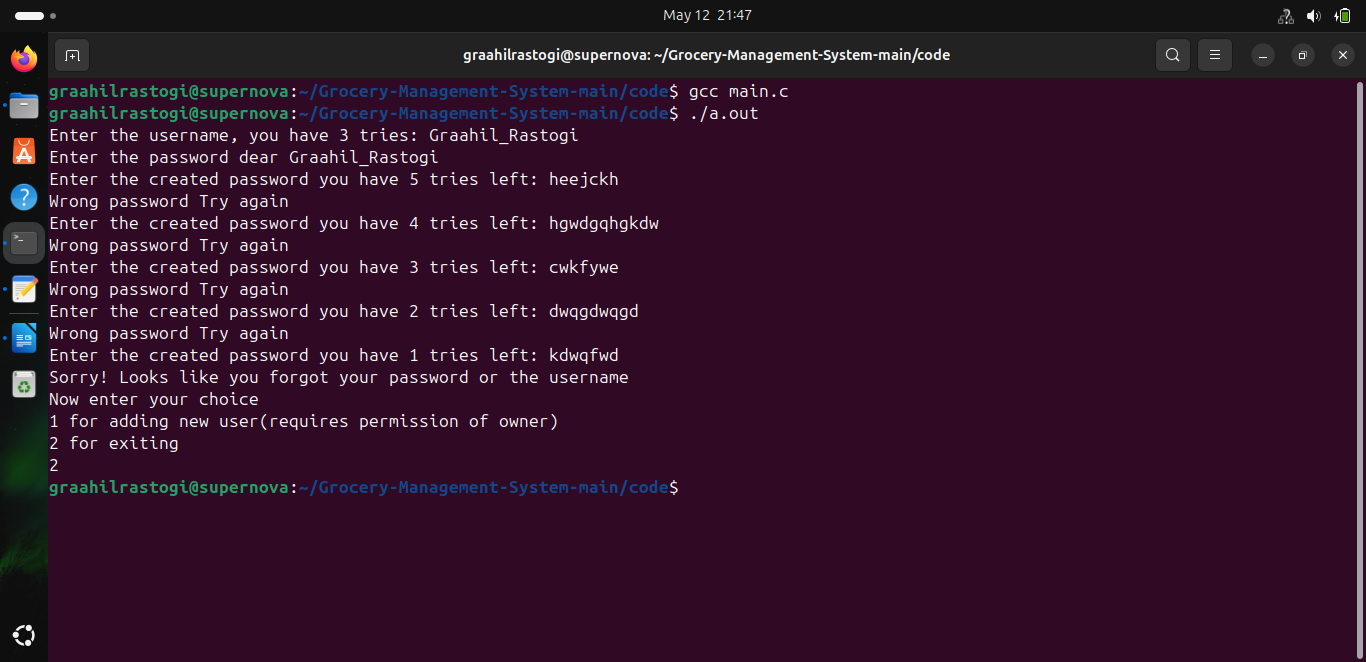
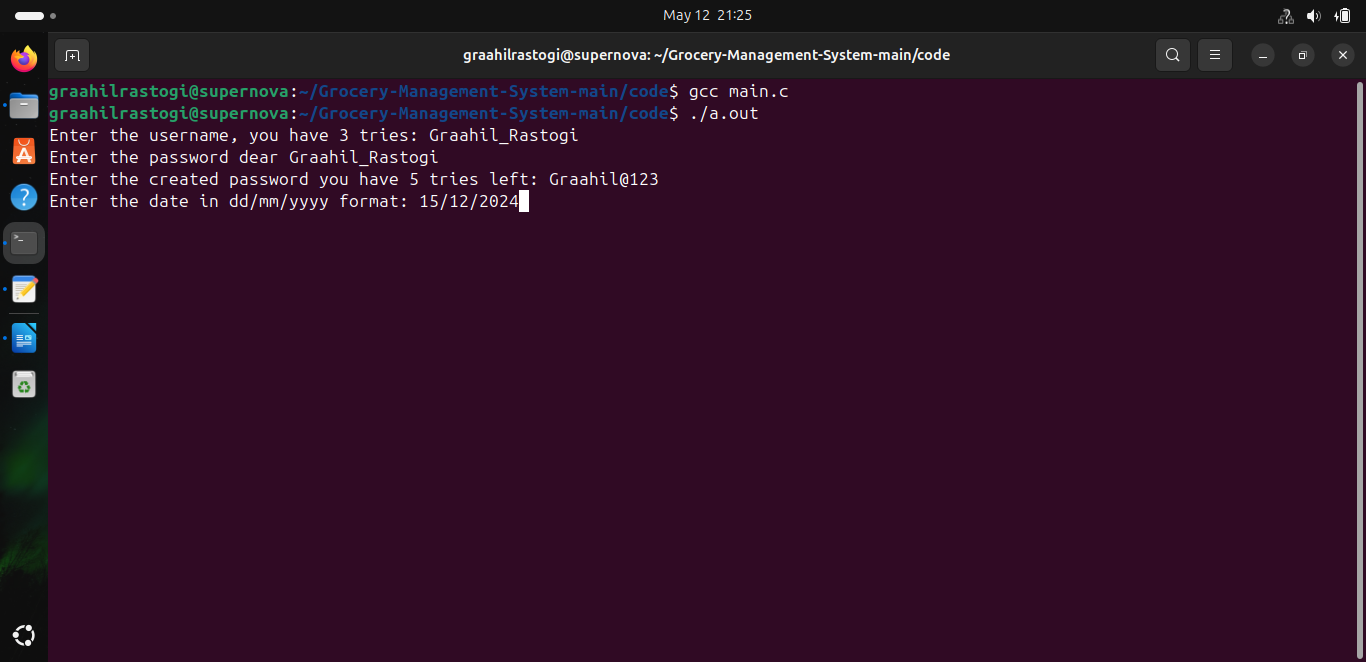
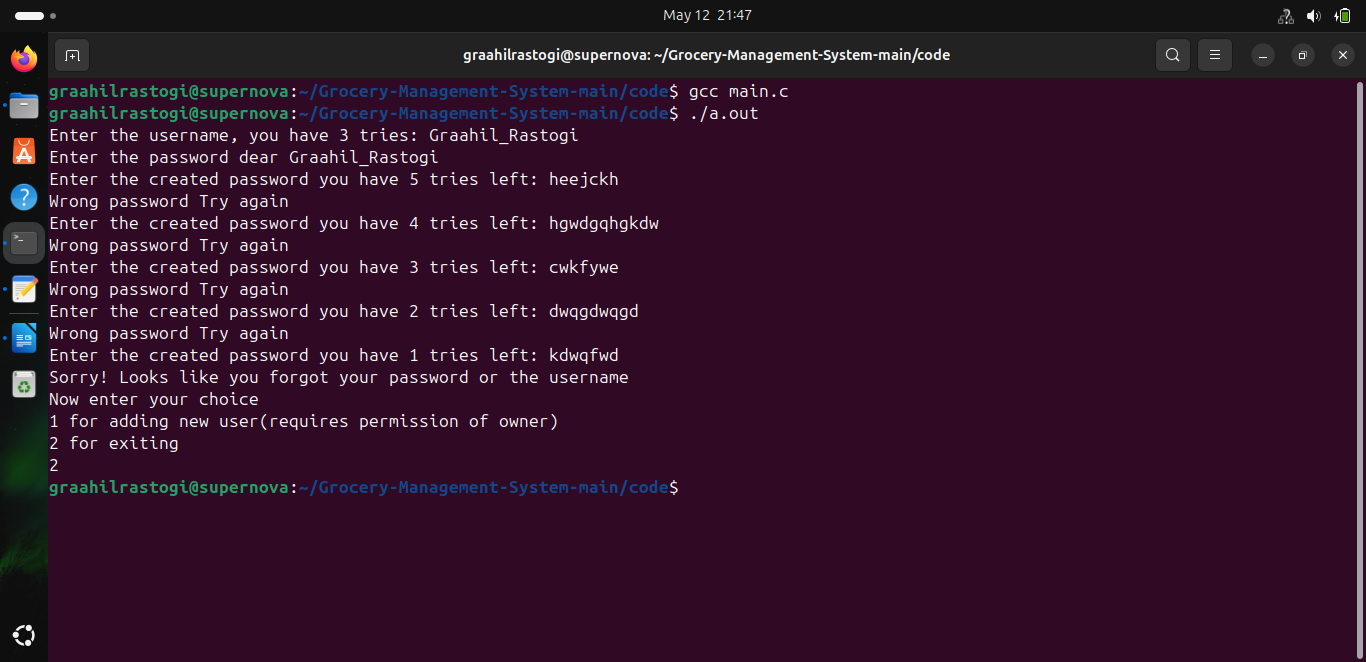
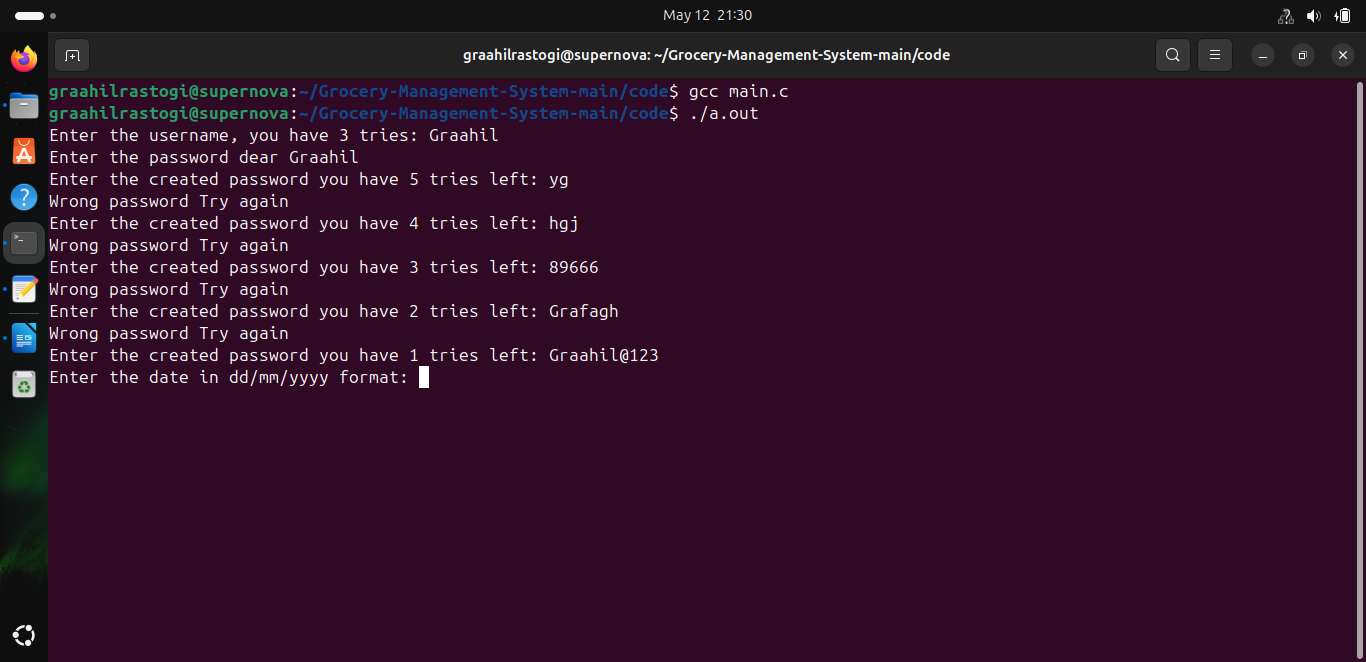
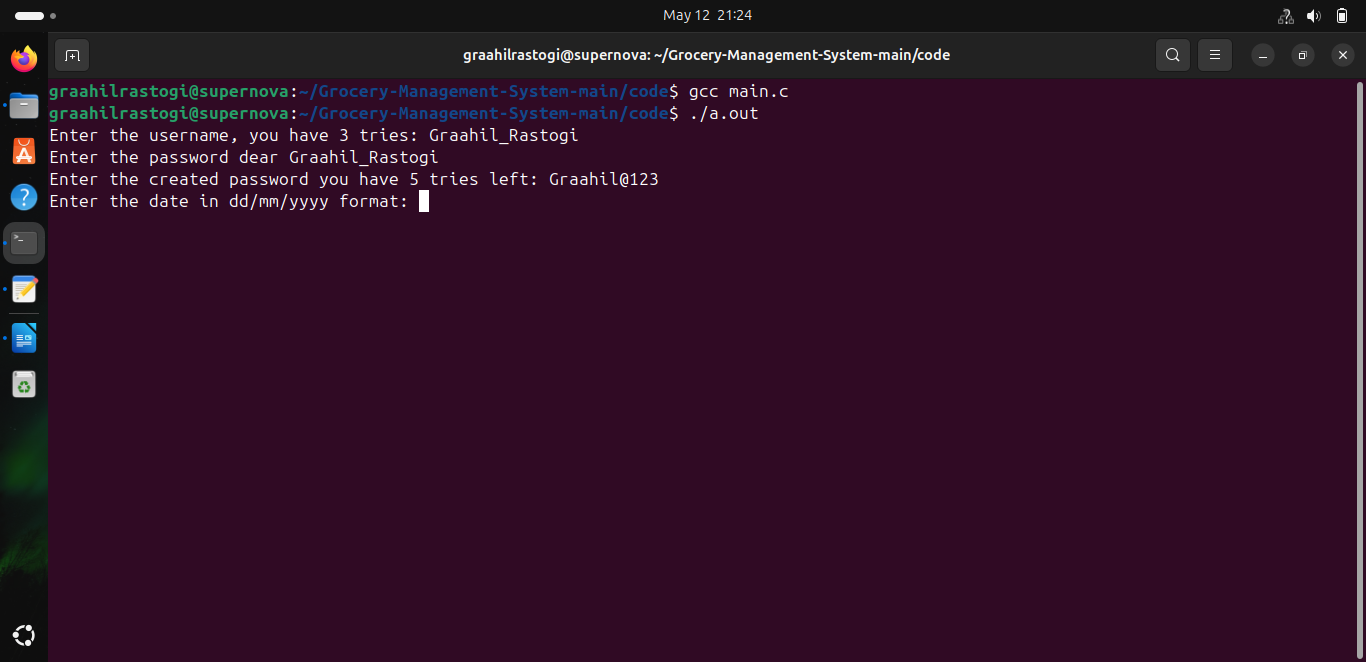
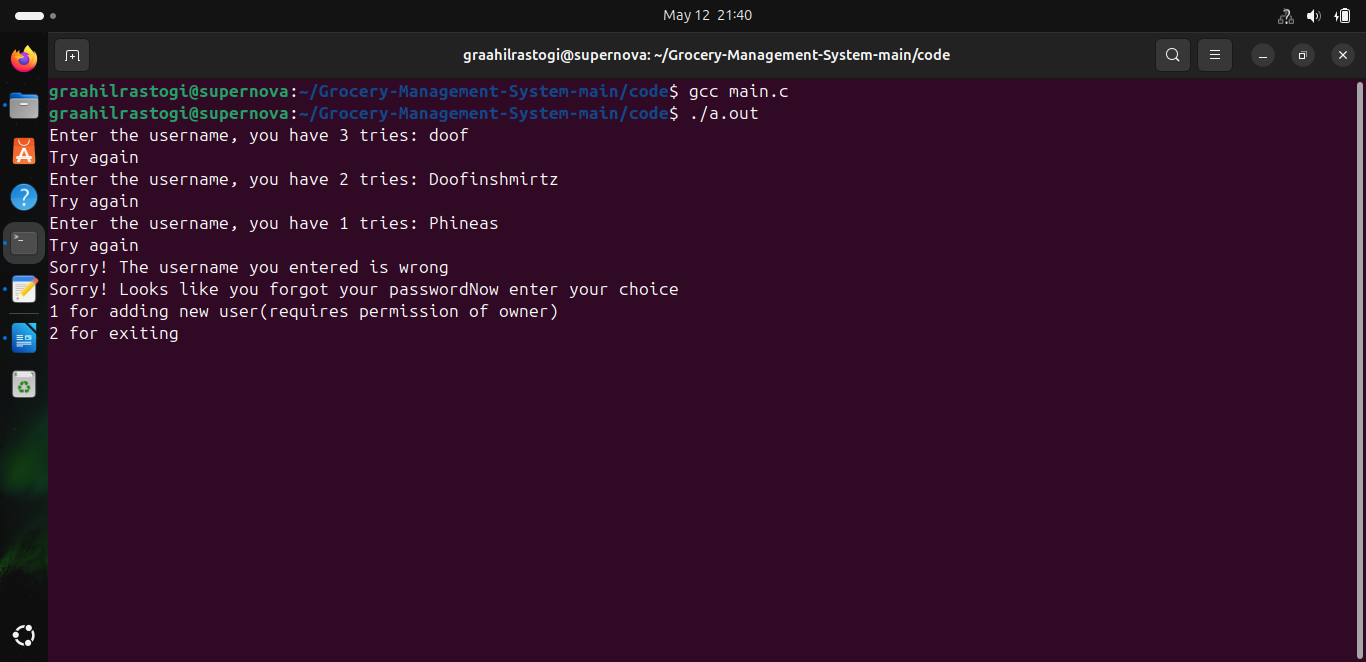
- Monitor sales trends and optimize product placement

- 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 request the user to create a password for logging in later, a file will be created and this function stores it in an encoded version using another function named encode. It will store the password into a separate file. This function will be executed the first time the application is used.
* **void authenticator():** This function will ask the user to input the password he set, if the password matches within 5 attempts the program will exit with a message “sorry!! Looks like you forgot your password”.
* **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, also their percentage, and even the current inventory value. The formulas are given:

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

Where, a[][] stores current stock, purchased quantity, sold quantity of the 10 items b[][] stores cost price, selling price of 10 items respectively.

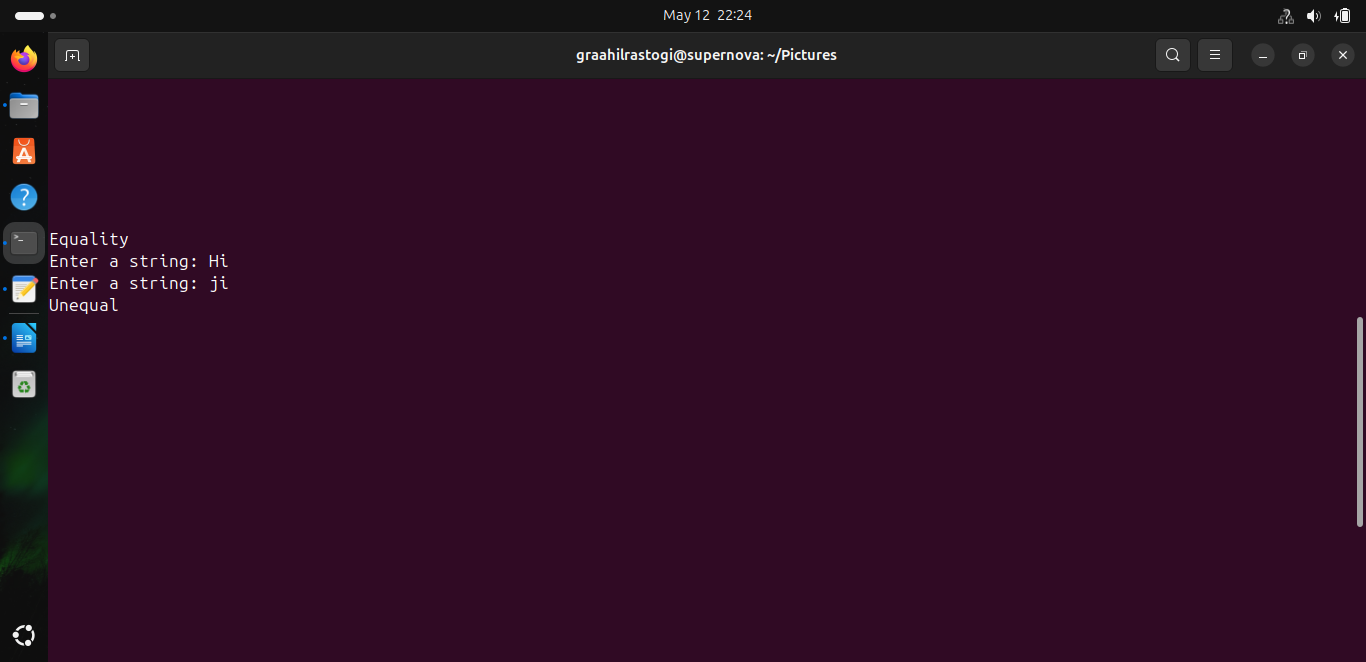
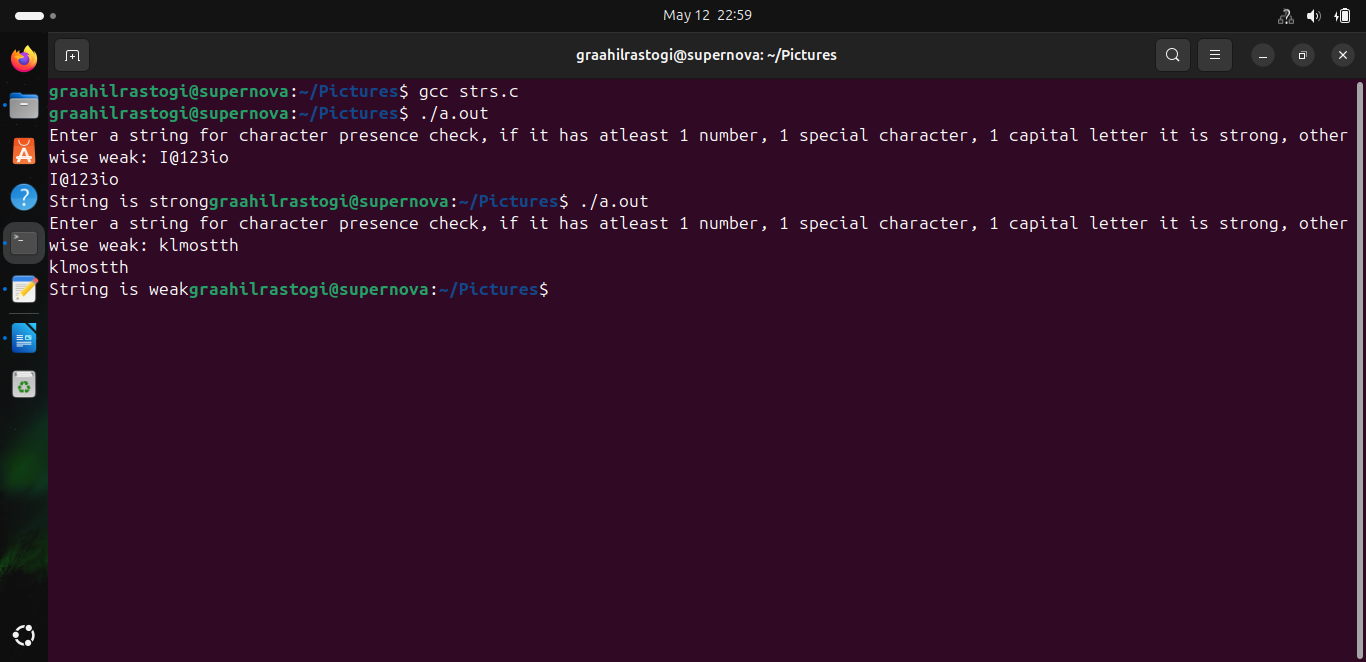
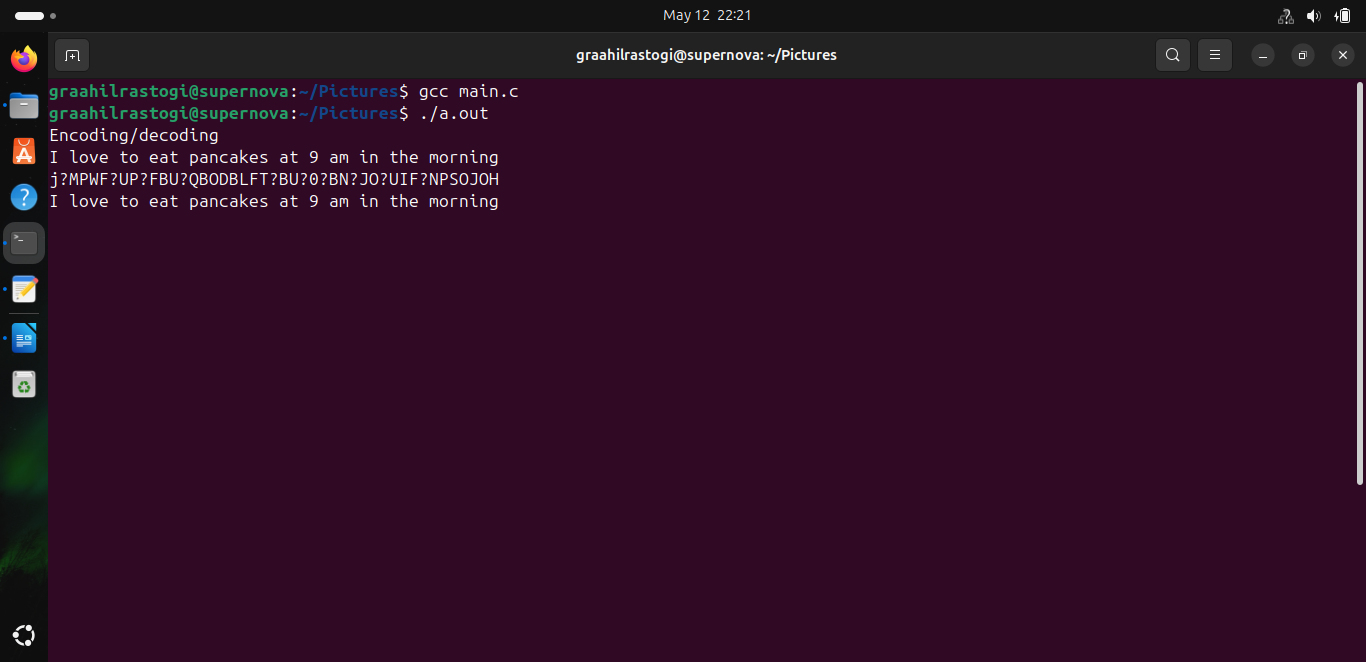
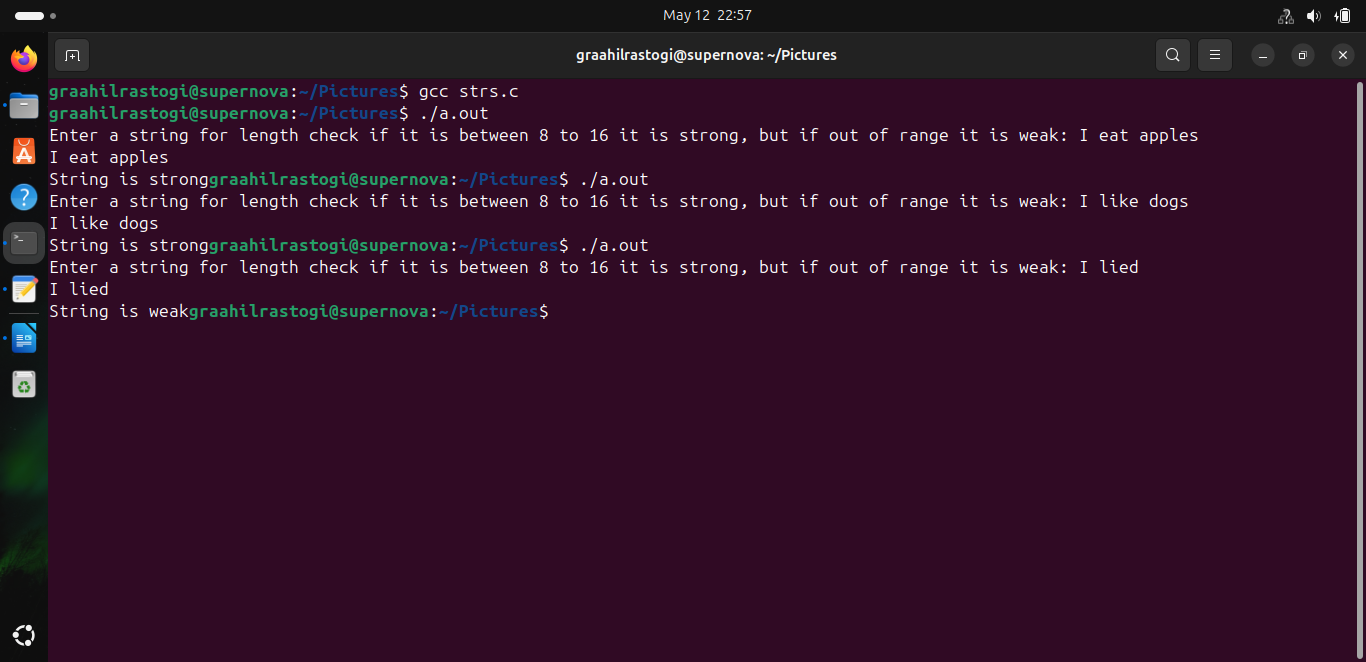
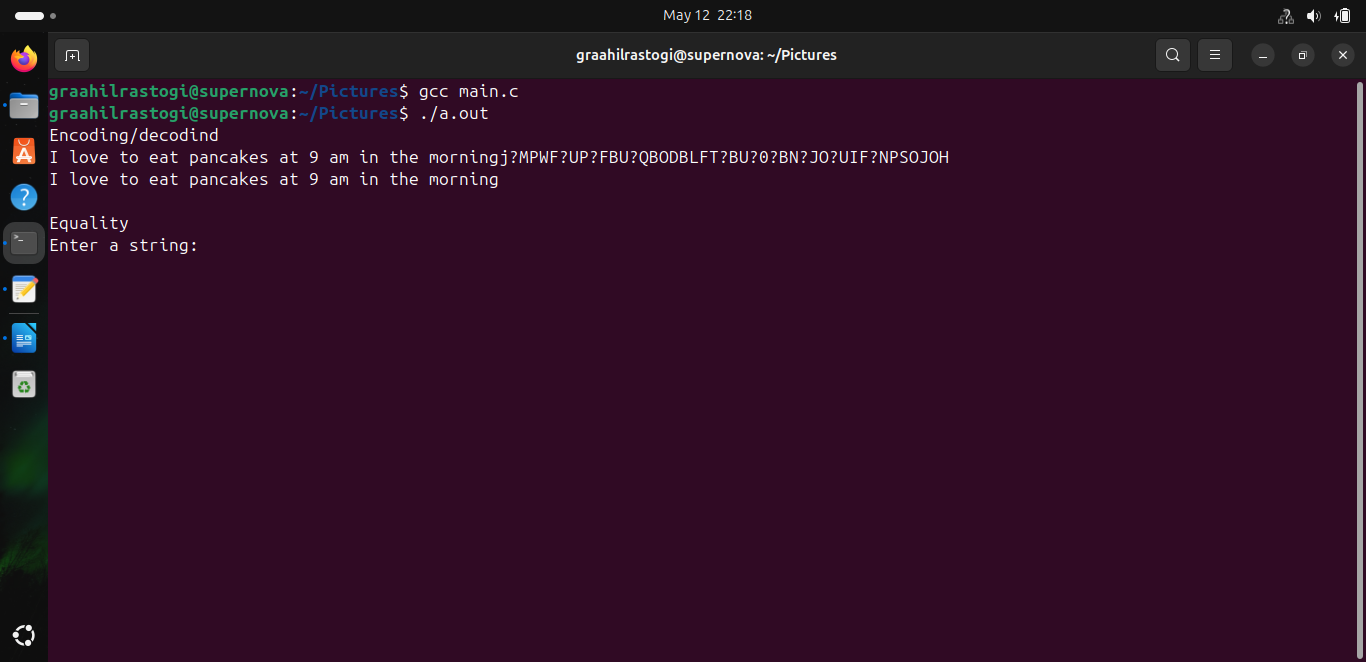
* Data Module
* **void store():** This function will ask the user to input item names, the initial quantity of items he has/ current stock. It will store the names of items in a structure attribute, the quantity in another structure attribute, for easier access. Just like the above function it will work 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

Where current stock, purchase amount, sold goods will be read from the file, converted to integers. And a is in a structure.

* **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, regardless of the platform used.
* **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. This function will be called at the end of the program. This will print the message that will thank the user for using this application.
* 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, this error happened while reading from file, so this function fixes the bug making file reading smooth.



* 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. It also uses a string buffer to turn the date into an integer.
* **void printdata():** This function will print the details of groceries in a particular month and year. This function will allow the user to see the complete log of his purchases in a given month and year. It uses the similar algorithm as dateddate to read the contents of the file.
* String\_Former Module
* **void sTU(char a[]):** This function is used to replace the spaces in a string with underscore. Replacing a space with an underscore facilitates in data reading, writing, and data representation. As many functions in C that read strings terminate at a space, so to turn spaces into underscores this function is used
* **void uTS(char a[]):** On the other hand this function replaces underscores with a space. The aim is to diplay the read string from the file. And this function solves the problem. As it replaces underscores with a space, it makes a string more readable, and better to look at.

* **Required topics from the subject:**

String manipulation:

* Facilitates encoding, decoding of words
* Facilitates conversion of strings into numbers that can be used further

Structures:

* Can be used to group multiple arrays together
* Each one of the array is related under this name, so can be further used
* Also is used to represent dates

File handling:

* Allows permanent data storage
* Custom files that cannot be opened, facilitating password storage
* Facilitation of storing file values into various other data structures

Arrays:

* Group multiple data under a single name
* Facilitates storage of different kinds of quantity under 1 name.

Pointers:

* Facilitate faster access of memory
* Point to an array, or a string to give easy access

Dynamic memory allocation:

* Allows for more customizable storage spaces
* Used in creating the inventory value array.
* **Platform Required:**

CodeBlocks IDE

VS Code

* **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