**INTRODUCTION**

* 1. **PROBLEM DEFINITION**

The problem that the above code is trying to solve is the management of a collection of currencies. Specifically, the program aims to provide a way for the user to:

* Read in a list of currencies from a text file
* Search for a specific currency by name
* Add new currencies to the collection
* Print the details of all currencies in the collection

This problem can be particularly challenging when dealing with large collections of currencies, as manual management can be time-consuming and error-prone. The use of a doubly linked list data structure and appropriate functions allows for efficient manipulation and of the collection and provides a user-friendly interface for interacting with the collection.

* 1. **OBJECTIVES**

The objectives for this mini-project are as follows:

The use case for the above code could be for someone who has a personal collection of rare currencies and wants to keep track of the details of each currency in the collection. They could use the program to read in a list of currencies from a text file and build up their collection.

They would be able to search for a specific currency by name, and if the currency is found, its details would be printed. They could also add new currencies to the collection and print the details of all the currencies in their list. They could also add new coin to the file or print the details of all currencies in the collection, which will help them in keeping their collection updated and providing them with the necessary information about their collection.

In general, the program would be useful for anyone looking for a tool to manage and keep track of their personal collection of currencies, who wants an easy and efficient way to add, search and manipulate the collection.

## METHODOLOGY TO BE FOLLOWED

1. This struct can be used to create a linked list by allocating new instances of the struct and linking them together via their "next" pointers, forming a chain of nodes. The head of the list is a pointer to the first node, and the tail is a pointer to the last node. Here in this code the structure is used to store different data types. To store currency, dynasty, king name and buy value of the coin. And structure is also used for creating the double linked list.
2. Conditional statements are used in programming to control the flow of execution based on given conditions. The "if" statement allows you to test a condition, and if the condition is true, a block of code is executed. We can also include an "else" clause to execute a different block of code if the condition is false. Additionally, you can use "else if" for checking multiple conditions together.
3. Looping statements like while, Do-while loop is a type of control flow statement that allows you to repeatedly execute a block of code as long as a certain condition is true. The code inside the loop will be executed at least once, even if the condition is false, unlike traditional while loop where loop will not be executed if the condition is false. In this code we used do while loop for login part of the code.
4. A doubly\_linked list is a type of linked list in which each node has a reference not only to the next node, but also to the previous node. Doubly\_linked list node would have two pointers, one for the next node and one for the previous node. Here we used it for doing the operation (insertion, display) with the currency.
5. Certain File handling commands are used to store the values related with the coins in the file. Here we are using it as a data\_base. Command like such as fopen(), fclose(), fread(), fwrite(), fscanf() and fprintf().

## EXPECTED OUTCOMES

1. Secures the collection with help of user\_name and password.
2. Mainly helps in managing the large number of coin collection.
3. Helps in easy access to the coin details, provides operations like knowing the details of the coin, insertion of the new coin and display of all the coins in the collection with the details.
4. And also helps to know the total number of coins in the collection.

## HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

* + 1. Personal Computer/Laptop.
    2. Intel Pentium Processor or later.
    3. RAM 512 MB or more.
    4. Storage of 128GB or more.

Software requirements:

1. Windows 7 32-bit/64-bit or above.
2. C language Compiler.

## CHAPTER 2

**DATA STRUCTURES**

A data structure is a way of arranging and storing. So that it can be accessed and modified efficiently. Different types of data\_structures are suited to different kinds of applications, and some are highly specialized to specific tasks.

**Linear data structure:** Data structure in which data fields are arranged sequentially or linearly, where each element is attached to its previous and next adjacent elements, is called a linear data structure.

**Static data\_structure:** Static data structure has a fixed memory size. It is easier to access the elements in a static data structure.

**Dynamic data structure:** In dynamic data structure, the size is not fixed. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code.

**Non-linear data structure:** Data structures where data\_elements are not placed sequentially or linearly are called non-linear data structures. In a non\_linear data structure, we can’t traverse all elements in a single run only.

* 1. **STRUCTURES**

A structure creates a data type that can be used to group items of possibly different types into a single type.

Syntax example of declaration of a structure:

struct currency

{

char name[50];

char dynasty[100];

char king[50];

int buyvalue;

};

The use case of struct in this code is as follows:

This struct can be used to create a linked list by allocating new instances of the struct and linking them together via their "next" pointers, forming a chain of nodes. The head of the list is a pointer to the first node, and the tail is a pointer to the end node.

Here in this code the structure is used to store different data types. To store currency, dynasty, king name and buy value of the coin. And structure is also used for create the double linked list.

# 2.2 LINKED LIST

A linked-list is a liner data\_structure, in which the elements are not stored in contiguous memory locations. The elements in a linked\_list is connected with the help of pointers.

A doubly-linked list is a type of linked list in which each node has a reference not only to the next node, but also to the previous node. Doubly\_linked list node have 2 pointers, one for the next node and one for prev node. Here we used it for doing the operation (insertion, display) with the currency.

The doubly\_linked list use case in the our code :

typedef struct Currency\_t {

char name[MAX\_NAME\_LENGTH];

char dynasty[MAX\_NAME\_LENGTH];

char king[MAX\_NAME\_LENGTH];

int buyValue;

struct Currency\_t \*next;

struct Currency\_t \*prev;

} Currency;

The benefit of using linked list is mainly due to its advantages over the array like:

1. Dynamic size
2. Efficient insertion and deletion
3. Lower memory overhead.

## CHAPTER 3

**DESIGN**

* 1. **DESIGN GOALS**

*Our design mechanism includes three main things:*

1. Secure and manage user data (coin) collection.
2. Reduce the complexity in excessing and knowing the details of the coin.
3. Helps in easy insertion of the new coins to the collection.

## ALGORITHM / PSEUDOCODE

1. *Start*
2. *At first login to user account.*
3. *During login insert username and password*
4. *If correct you will be logged in successfully or else re-enter the details again until the details are correct.*
5. *Later it asks the user to input the coin name he wants to search for or he wants to add it to its collection.*
6. *It checks for the entered coin name in its collection*

*If present in the collection it will display its details stored.*

*Else it will ask the user whether he wants to add it to his collection.(A new coin will be added).*

1. *If yes then it asks the user to enter the details of the dynasty it belongs, king name who introduced it and the buy value of the coin.*

*Else moves to the next operations.*

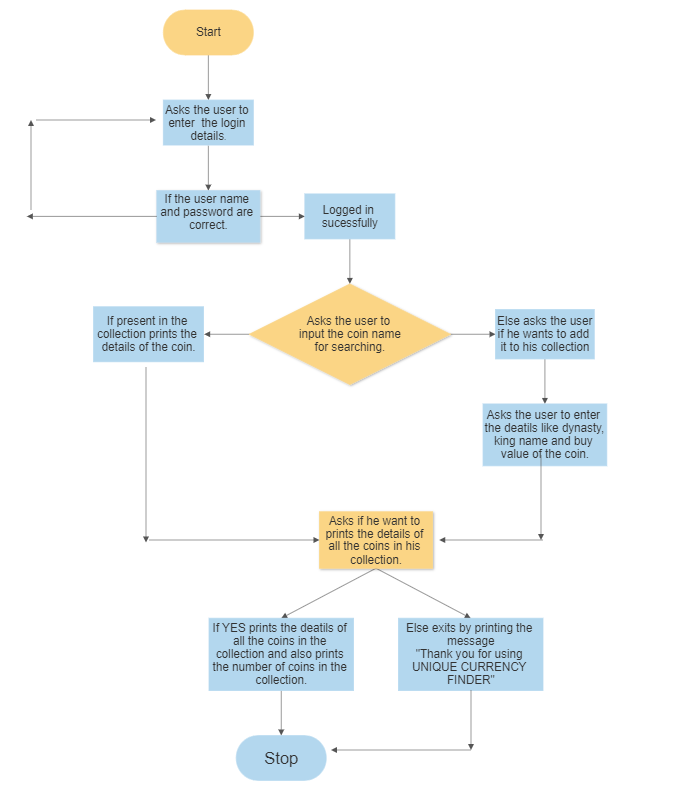
1. *Then later it asks whether the user wants to print the details of all the coins in his collection. ( All the details including name, dynasty, king name and buy value)*
2. *If yes prints the details of the coin and the total count of the coins will also be printed and then terminates*

*Else prints the message “Thank you for using the Unique Currency Finder” and then terminates.*

1. *Stop*

*.*

## 3.3. FLOWCHART

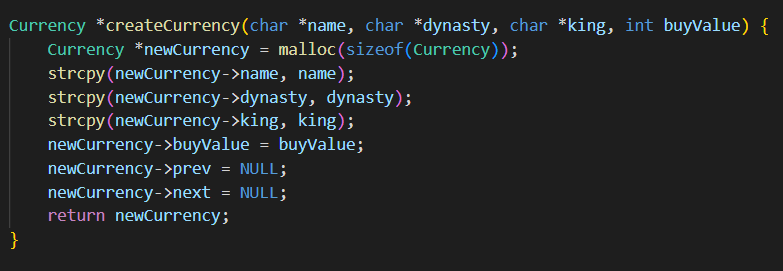


**Fig 3.1: Design flowchart**

**CHAPTER 4**

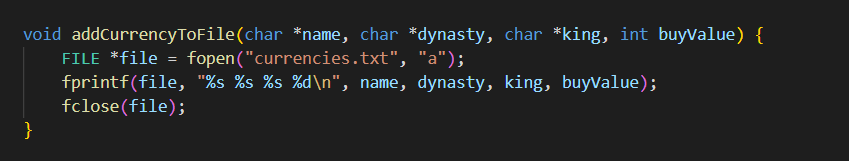
**IMPLEMENTATION**

* 1. **MODULE 1 FUNCTIONALITY**



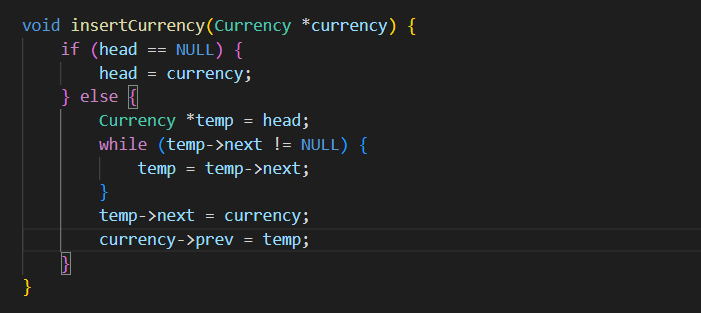
1. **createCurrency** which creates a new node and initializes the values and the next and previous pointers.
2. The **createCurrency** function is defined, which takes four arguments: **name**, **dynasty**, **king**, and **buyValue**, and returns a pointer to a new **Currency** struct. The function uses the **malloc** function to allocate memory for the new struct and sets the values of its fields using the arguments passed to the function. The **strcpy** function is used to copy the strings passed as arguments into the **name**, **dynasty**, and **king** fields. The **prev** and **next** fields are initialized to **NULL**.

**MODULE 2 FUNCTIONALITY**



1. **addCurrencyToFile** is used to add new coin to the file(currencies.txt).
2. The **addCurrencyToFile** function is defined, which takes four arguments: **name**, **dynasty**, **king**, and **buyValue**, and appends a line to the "currencies.txt" file with the values of these arguments. The **fopen** function is used to open the file in "append" mode, and the **fprintf** function is used to write the values to the file. The **fclose** function is used to close the file after writing.

**MODULE 3 FUNCTIONALITY**



1. **insertCurrency** function which adds a new node to the list and links the current last node with the new node.
2. The **insertCurrency** function is defined, which takes a pointer to a **Currency** struct as an argument. The function inserts the struct at the end of the linked-List by updating the **next** and **prev** pointers of the appropriate structs. If the linked list is empty (i.e., **head** = **NULL**), the function sets **head** to the struct passed as an argument.

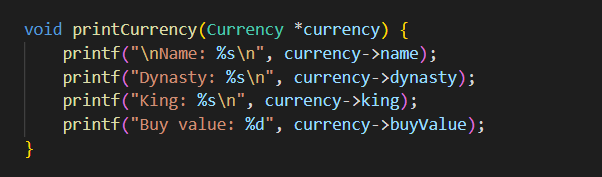
**MODULE 4 FUNCTIONALITY**

A screenshot of a computer

Description automatically generated with medium confidence

1. **findCurrency** function which searches the list for a specific currency, it compares the input name with the name of the nodes in the list.
2. The **findCurrency** function is defined, which takes a string as an argument and searches the linked list for a **Currency** struct with a matching **name** field. If a match is found, the function returns a pointer to the struct. If no match is found, the function returns **NULL**.

**MODULE 5 FUNCTIONALITY**



1. **printCurrency** function which print the details of the specific node (single currency) when called.
2. The **printCurrency** function is defined, which takes a pointer to a **Currency** struct as an argument and prints the values of its fields using the **printf.**

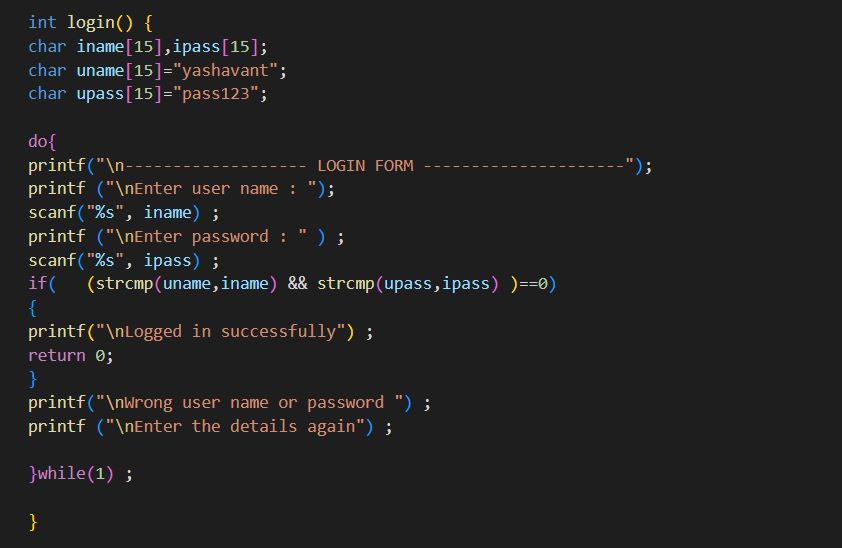
**MODULE 6 FUNCTIONALITY**

Text

Description automatically generated

1. **printCurrencies** is used to display the total details of all the nodes(currency name) present in the list.
2. The **printCurrencies** function is defined, which iterates through the linked list of currencies and calls the **printCurrency** function on each one to print the details of all the currencies. The function also keeps a count of currencies in the linked list and prints this count after all the currencies have been printed.

**MODULE 7 FUNCTIONALITY**

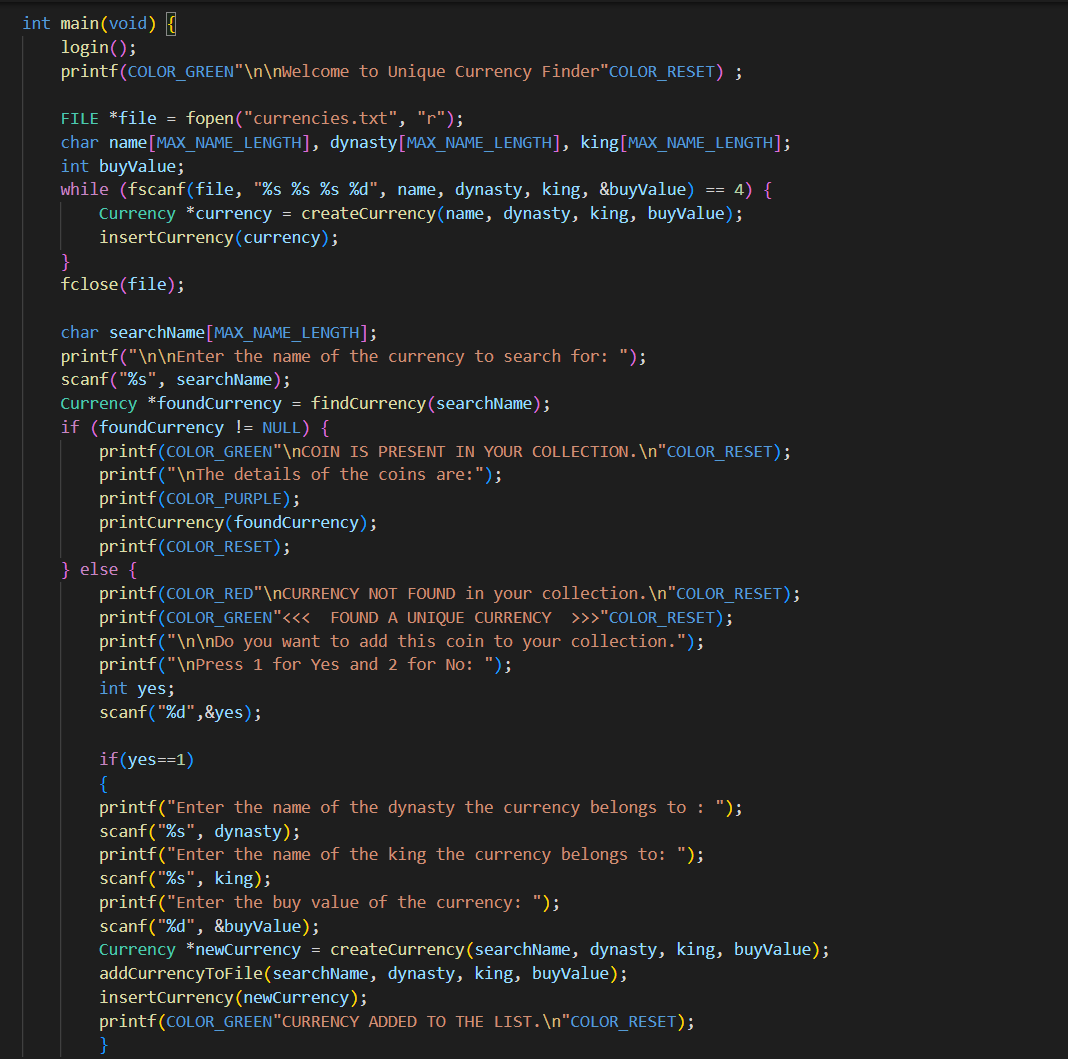


1. login function, which helps to secure the account, asks the user for the username

and password for the entry.

1. The login function is defined, it prompts the user to enter a username and password. It then compares the entered username and password with predefined hardcoded values "yashavant" and "pass123" respectively. If they match, it prints a message "Logged in successfully" and exits the function. If they do not match, it prompts the user to enter the details again, in an infinite loop until the correct details are entered.

**MODULE 8 FUNCTIONALITY**



Text

Description automatically generated

1. The **main** function is defined, which is the entry point of the program. The function first reads in the currencies from the "currencies.txt" file using the code from the original program. This code reads the file line by line, using the **fscanf** function to parse the name, dynasty, king, and buy value of each currency. It then creates a new **Currency** struct for each currency and inserts it into the linked list using the **insertCurrency** function.
2. After reading in the currencies from the file, the **main** function prompts the user to enter the name of a currency to search for. It uses the **scanf** function to read the user input and stores it in a **searchName** variable.
3. The **main** function calls the **findCurrency** function with the **searchName** variable as an argument to search for the currency in the linked list. If the currency is found, the **main** function prints a message indicating that the coin is present in the collection and calls the **printCurrency** function to print the details of the coin. If the currency is not found, the **main** function prints a message indicating that the currency was not found and prompts the user to enter 1 if they want to add the coin to their collection or 2 if they don't want to add the coin.
4. If the user enters 1, the **main** function prompts the user to enter the name of the dynasty the currency belongs to, the name of the king who issued the currency, and the buy value of the currency. It then calls the **createCurrency** function to create a new **Currency** struct with these values and the **searchName** as the **name** field. Finally, it calls the **insertCurrency** function to insert the new struct into the linked list and the **addCurrencyToFile** function to append the details of the new currency to the "currencies.txt" file.
5. After the user has finished adding currencies or if they chose not to add a currency, the **main** function calls the **printCurrencies** function to print the details of all the currencies in the linked list.

## CHAPTER 5

**RESULTS**

**5.1 Snapshots for login:**

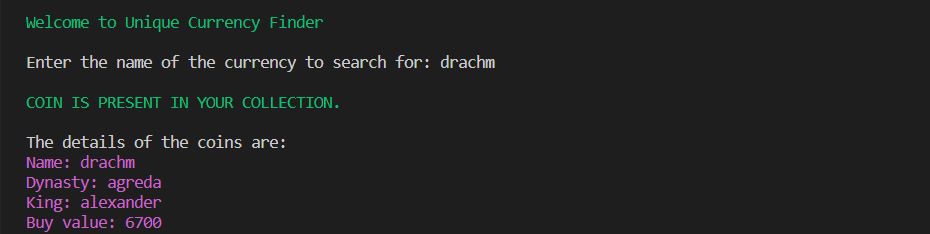
Text

Description automatically generated

**Fig. 5.1: Output for login**

Here we can see that the entry for the user is given only when he enters the correct username and password in case of wrong inputs it asks the user for the details again.

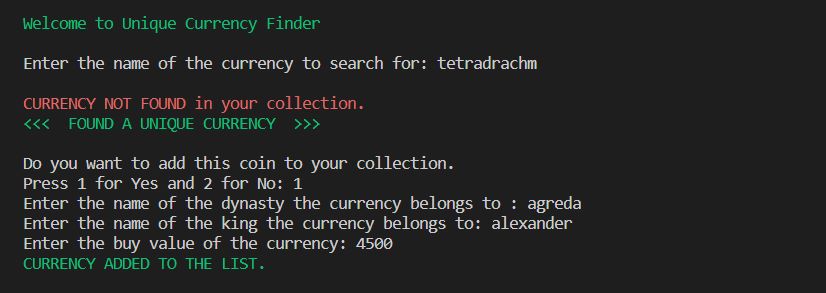
**5.2 Snapshots for coin already present in collection:**



**Fig. 5.2: Output for existing coin**

If the coin is present in the collection, it prints all the details associated with the coin name.

**5.3 Snapshots for coin does not present in collection:**



**Fig. 5.3: Output for new coin entry**

If the coin entered is not present in the users collection it asks for its entry into the collection. It ask the name of the dynasty, king name and buy value of the coin. After entering the details the coin will be added to the collection.

**5.4 Snapshots for printing all the coins:**

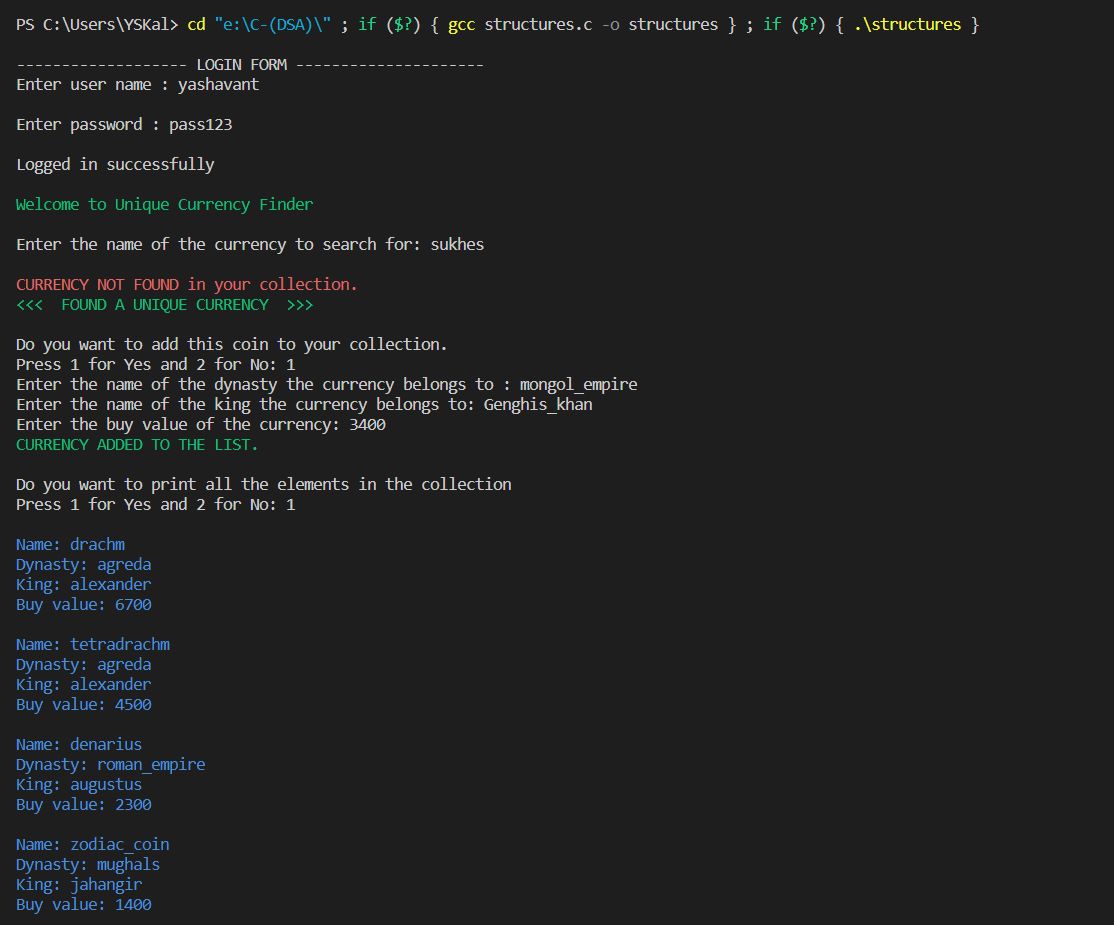
Text

Description automatically generated

**Fig. 5.4: Output for displaying all the coins**

If the user wants to print the entire collection, it can be done. Here in the above output, you can see that all the coins are printed in order and even the count of the coins is also printed at the end.

**5.5 Snapshots of one whole operation:**



Text

Description automatically generated

Here the output gives the one whole operation like inserting new coin and displaying all coins and the coin count.

**CONCLUSION**

* The motive of mini project has successfully completed the goals it had set out in the objectives and design sections of this report.
* The user (numismatic) will be able to arrange or manage his collection easily without much difficulty. The handling of large number of coins is easy now.
* They would be able to search for a specific currency by name, and if the currency is found, its details would be printed. They could also add new currencies to the collection and print the details of all the currencies in their list. They could also add new coin to the file or print the details of all currencies in the collection, which will help them in keeping their collection updated and providing them with the necessary information about their collection.