LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4059NI Fundamentals of Computing

Assessment Weightage & Type

50% Individual Coursework

Year and Semester

2022 Spring

Student Name: Rounak pradhan

London Met ID: 22066975

College ID: NP01NT4A220198

Assignment Due Date: Friday, May 12, 2023

Assignment Submission Date: Friday, May 12, 2023

Word Count: 4459

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of figures

List of tables

1. Introduction

This report's goal is to present an overview and analysis of a system for managing laptop inventories. The system intend to keep track of sales and purchase, manage the inventory of computers, and produce client invoices. The system's design and implementation, including the software tools utilized, the data structures and algorithms used, and the user interface, will be covered in this report. Additionally, it will go through the main aspects and capabilities of the systems, as well as its advantages and disadvantages.

This code is designed to display a welcome message and a menu of the options to the user. The user can either choose to sell a laptop or buy a laptop. If they choose to sell a laptop, they will be asked to enter the name of the buyer and the ID of the laptop they want to sell. The program will then validate the input and update the inventory accordingly. The user can choose "yes" when questioned if they wish to sell more laptops. For every extra laptop they want to sell, the program will then go through the same process again. When the suer is done selling laptops, the application will show the buyer's bill.

If the user decides to purchase a laptop, they will be prompted to provide the seller's name and the laptop's ID. Following input validation, the application will update the inventory as necessary. The user can indicate their want to purchase additional laptops by responding "yes" to the prompt. The process will then be repeated for each additional laptop the users wishes to purchase. Once the user is finished buying laptops, the program will calculate a fine based on the number of days the laptops were in the inventory and display a bill for the seller.

2. Algorithm

Introduction:

A process for solving a problem or completing a computation is referred to as an algorithm. Algorithms are a precise set of instructions that perform specified operations in either hardware or software-based process. Algorithms are composed of a series of steps, like seen below, but they may also include flowcharts and pseudocode to further clarify the code (Alexander S. Gillis, 2022).

Process:

Step 1: Start

Step 2: Display welcome message

Step 3: Display options to sell, buy or exit

Step 4: Enter a value of 1 to sell, 2 to buy, and 3 to exit.

Step 5: If 1 is taken as input, go to step 10,else go to step 6

Step 6: If 2 is taken as input, go to step 39,else go to step 7

Step 7: If 3 is taken as input, then display thankyou message else go to step 9

Step 8: END

Step 9: If an invalid input is taken, display an invalid message and go to step 3

Step 10: create a 2d list “items”

Step 11: Display details related to the laptops

Step 12: Input the ID of the laptops to be sell

Step 13: If a valid id is choosen go to step 16 or go to step 14

Step 14: Invalid input message

Step 15: go to step 11

Step 16: Input the quantity of the laptops to be sell

Step 17: if the laptops is in stock go to step 20 else go to step 18

Step 18: Display Quantity entered exceeds the available stock

Step 19: go to step 16

Step 20:Display the laptops list

Step 21: Append id and quantity of the laptops sold to the item

Step 22: update the value of the laptops in the dictionary

Step 23: Ask if the user wants to sell more laptops

Step 24: If user say yes then go to 11 else go to step 25

Step 25: input the name of the user

Step 26: Display name ,date, price

Step 27: Invoice with users details , laptops sell and grand total

Step 28: Display thank you message

Step 29: go to step 3

Step 30: create a 2d list “items”

Step 31: Display the list of the laptops

Step 32: Input laptops id to buy

Step 33: if a valid id is chosen go to step 36 else go to step 34

Step 34: Invalid input message

Step 35: go to step 31

Step 36: Input the quantity of the laptops to buy

Step 37: Display the updated laptops list

Step 38: Append id and quantity of the laptops buy to the item

Step 39: update the value of the laptops in the dictionary

Step 40: Ask if the user wants to buy more laptops

Step 41: If user say yes then go to step 31 else go to step 42

Step 42: Input the name of the user

Step 43: Display name, date and time , cost

Step 44: Invoice with user details, grand total

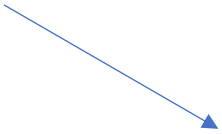
Step 45: go to step 28

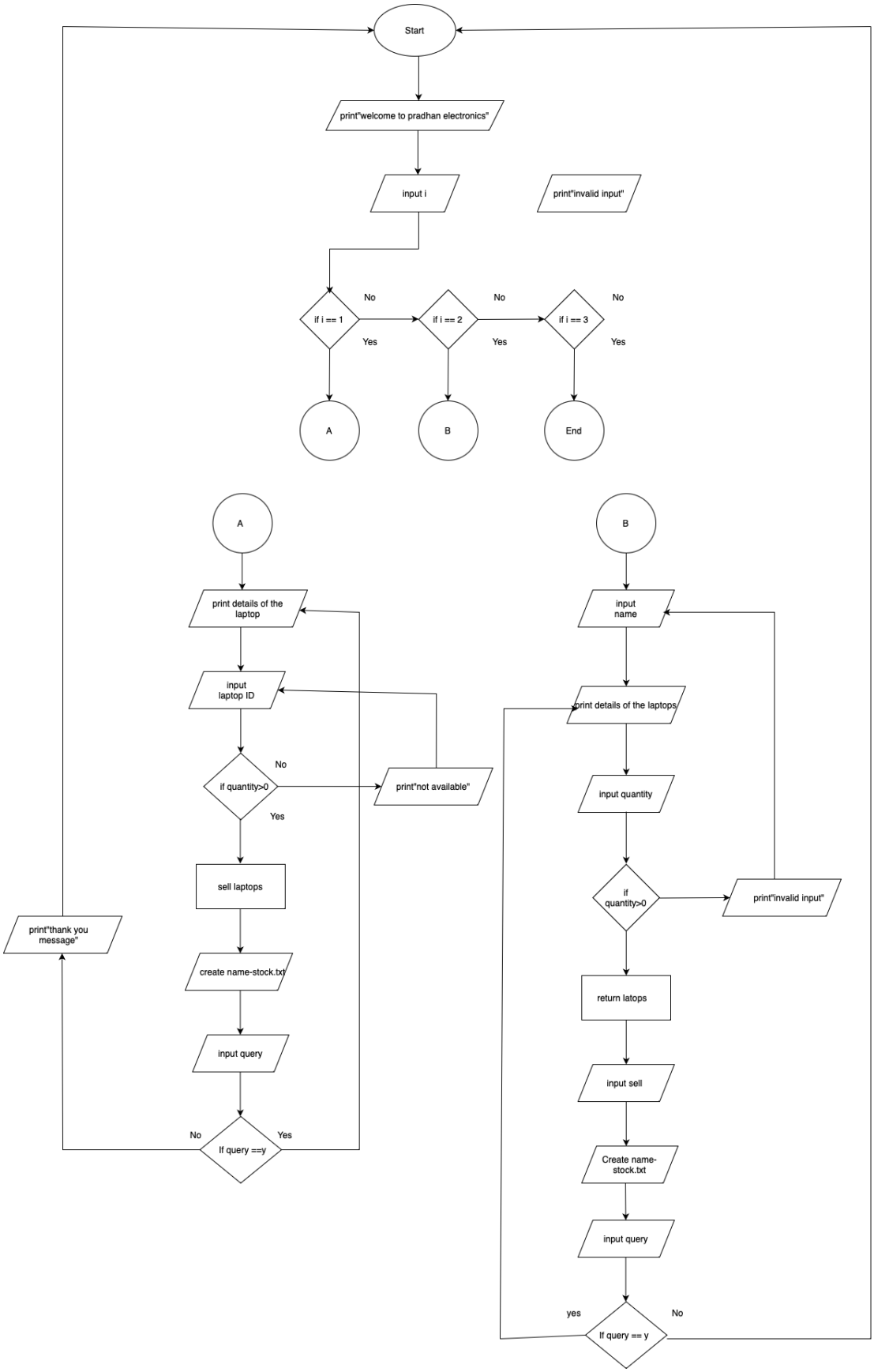
3. Flowchart

A flow chart is a visual or graphical representation of a process. Each process step is represented by a different symbol and includes a brief description of the stage (stage., 2020).

There are five symbols used in this flowchart:

Table 1: Flowchart

Symbol	Name	Function
	Start/end	The oval represents the start or end point.
	Input/output	A parallelogram represents the input or output.
	Decision	A diamond symbol represents a decision.
	Process	A rectangle represents the process.
	Arrow	The arrow shows relationships between different shapes.


Start
print"welcome to pradhan electronics"
input i
print"invalid input"
if i == 1
No
Yes
if i == 2
No
Yes
if i == 3
No
Yes
A
B
End
A
print details of the laptop
input laptop ID
if quantity>0
No
Yes
print"not available"
sell laptops
create name-stock.txt
input query
If query ==y
No
Yes
print"thank you message"
B
input name
print details of the laptops
input quantity
if quantity>0
print"invalid input"
return latops
input sell
Create name-stock.txt
input query
If query == y
yes
No

4. Pseudocode

Pseudocode is a mechanism for producing programming code in a natural language such as English. It is a streamlined programming language that is text-based. It is designed for human reading, not automated reading. It does not call for any rid grammar or programming language (Mahr, study.com, 2022).

Module: main.py

```
IMPORT laptop_read
IMPORT messages
IMPORT update_dict
IMPORT functions
DEFINE start()
    DISPLAY welcome messages
    INITIALIZE choice = True
    WHILE True
    DISPLAY choices

    INITIALIZE False
    TRY:
    WHILE (True)
    INPUT choice
        a = True
    EXCEPT:
        DISPLAY messages
    IF choices is equals to 1:
    CALL messages.choice_one()
    CALL laptop_read.display_table()
    ADD dict_update = dictionary.dict_laptop()
    CALL laptop_Id = functions.checking_laptop_Id(dict_update)
    CALL laptop_list = [ ]
```

CALL laptop_brand = [ ]

IF items is greater than 0

CALL quantity is equals to functions.correct_quantity(dict_update,laptop_Id)

CALL dict_update[laptop_ID] [3] = int(dict_update[laptop_Id] [3]) – quantity

CALL update_dict.laptop_update(dict_update)

ADD to laptop_list

ADD to laptop_brand

CALL totalPrice = functions.price_calc(dict_update[laptop_Id] [2], quantity

WHILE (True):

INPUT name of the seller

IF name!= string

BREAK

DISPLAY messages

DISPLAY messages

INPUT yes or no

INITIALIZE Loop_extension = True

WHILE (True)

IF Loop.lower() is equals to “yes”:

CALL laptop_read.desplay(table()

CALL laptop_Id is equals to functions.checking_laptop_Id(dict_update)

IF int (dict_update[laptop_Id][3]) is greater than 0

CALL quantity = fucntions.correct_quantity(dict_update,laptop_Id)

CALL dict_update[laptop_Id][3] = int(dict_update[laptop_Id][3]) – quantity

FOR i in laptop_list:

IF i != dict_update[laptop_Id] [0]:

CALL laptop_list.append(dict_update[laptop_Id] [0])

FOR j in laptop_brand:

IF j != dict_update[laptop_Id] [1]:

CALL laptop_brand.append(dict_update[laptop_Id] [1])

CALL updateSum = function.dollar_renewed(dict_update[laptop_Id] [2], quantity)

CALL totalPrice = totalPrice + updateSum

DISPLAY message

INPUT yes or no

Loop_extension = True

ELSE

DISPLAY message

DISPLAY message

INPUT yes or no

Loop_extension = True

ELSE

Loop_extension = False

```
        CALL functions.sell_bill(name,laptop_brand,laptop_list,totalPrice)



    ELSE IF choice = 2
        CALL message.choice_two()
        CALL laptop_read.display_table()
        CALL dict_update = laptop_read.dict_laptop()


        CALL laptop_Id = functions.checking_laptop_Id(dict_update)


        CREATE laptop_list = []
        CREATE laptop_brand = []
        IF int(dict_update[laptop_Id] [3])>=0:
        CALL quantity = fucntions.quantity_validation(dict_update,laptop_Id)
        CALL dict_update[laptop_Id] [3] = int(dict_update[laptop_Id] [3]) + quantity


        CALL update_dict.laptop_update(dict_update)


        ADD to laptop_list
        ADD to laptop_brand
        CALL totalPrice = function.dollar_renewed(dict_update[laptop_Id]
[2],quantity)


        WHILE (True):
```

INPUT name of the buyer

IF name! = string

BREAK

DISPLAY message

DISPLAY message

INPUT yes or no

INITIALIZE Loop_extension = True

WHILE (True)

IF Loop.lower() is equals to “yes”:

CALL laptop_read.display_table()

CALL laptop_Id = fucntions.checking_laptop_Id(dict_update)

IF int(dict_update[laptop_Id] [3]) is greater than or equals to 0

CALL quantity = functions.quantity_validation(dict_update,laptop_Id)

CALL dict_update[laptop_Id] [3] = int(dict_update[laptop_Id] [3]) + quantity

CALL update_dict.laptop_update(dict_update)

FOR I in laptop_list:

IF i != dict_update[laptop_Id] [0]:

CALL laptop_list.append(dict_update[laptop_Id] [0])

```
                    FOR j in laptop_brand:
                        IF j != dict_update[laptop_Id] [1]:
                            CALL
laptop_brand.append(dict_update[laptop_Id] [1])

                    CALL update sum =
fucntions.dollar_renewed(dict_update[laptop_Id] [2],quantity)

                    CALL totalPrice = totalPrice + updateSum
                    DISPLAY message
                    INPUT yes or no

                    Loop_extension = True
                ELSE
                    DISPLAY message
                    DISPLAY message
                    INPUT yes or no
                    Loop_extension = True
            ELSE
                Loop_extension = False
        CALL functions.buy_bill(name,laptop_brand,laptop_list,totalPrice)
        INITIALIZE choice = True
ELSE
        DISPLAY message
```

ELSE IF choice == 3

CALL message.choice_three()

CALL choice = false

ELSE

CALL message.inavlid()

CALL choice = True

Module: update_dict.py

DEFINE FUNCTION laptop_update(dictionary)

CALL file = open(“stock.txt”,”w”)

FOR I in dictionary.values():

CALL file.write(i[0]+”,”+i[1]+","+str(i[2])+","+str(i[3])+","+str(i[4])+","+str(i[5]))

CALL file.write(“\n”)

CALL file.close()

Module: messages.py

DEFINE FUCNTION welcome_message()

DISPLAY("----------")

DISPLAY(" Welcome to the pradhan electronics")

DISPLAY("----------")

END FUCNTION welcome_message(0

DEFINE FUNCTION choices()

DISPLAY(“\n”)

DISPLAY("Select your desirable choice")

DISPLAY(“"(1) || Press 1 to sell a laptop.")

DISPLAY(“"(1) || Press 1 to buy a laptop.")

DISPLAY(“(3) || Press 3 to exit.")

DISPLAY(“\n”)

END FUNCTION choices()

DEFINE FUCNTION choice_one():

DISPLAY(“\n”)

DISPLAY(“The items available for sell id displayed below:”)

END FUNCTION choice_one()

DEFINE FUNCTION choice_two()

DISPLAY(“\n”)

DISPLAY(“Please buy accordingly”)

END FUNCTION choice_two()

DEFINE FUNCTION choice_three()

DISPLAY(“\n”)

```
        DISPLAY("Thank You for visiting us.")
END FUNCTION choice_three()
DEFINE FUNCTION invalid()
        DISPLAY("-----------------------------------------------------")
        DISPLAY(" Invalid input, Please choose a valid choice. ")
        DISPLAY("-----------------------------------------------------")
        DISPLAY("\n")
END FUNCTION invalid()
```

Module: functions.py

```
IMPORT datetime
DEFINE FUNCTION dollar_renewed(dollar_sign,quantity):
        CALL price = float(dollar_sign,replace("$",""))
        CALL total = price * quantity
        return total
END FUNCTION dollar_renewed(dollar_sign,quantity):
DEFINE FUNCTION checking_laptop_Id(dictionary):
        INITIALIZE success = False
        WHILE success == False
                TRY
                        Laptop_Id = int(input("Enter the desire laptop Id:"))
```

```
            WHILE laptop_Id<=0 or laptop_Id>len(dictionary):
                DISPLAY message
                Laptop_Id = int(input("Enter laptop Id: "))
            DISPLAY message
            INITIALIZE success = True
        EXCEPT
            DISPLAY message
        return laptop_Id
END FUNCTION checking_laptop_Id(dictionary):


DEFINE FUNCTION correct_quantity(dictionary,laptop_Id):
    INITIALIZE success = False
    WHILE success == False:
        Try
            CALL quantity = int(input("Enter the required quantity: "))
            while quantity<=0 or quantity>int(dictionary[laptop_Id][3]):
            IF quantity <=0:
                Print("Inavlid input for quantity")
            ELSE:
                DISPLAY message
            INPUT quantity
        INITIALIZE success = True
```

```
        EXCEPT
                DISPLAY message
return quantity
END FUNCTION correct_quantity(dictionary,laptop_Id):


DEFINE FUNCTION quantity_validation(dic,update,laptop_Id):
        INITIALIZE success = False
        WHILE success == False:
                Try
                        INPUT quantity
                        WHILE quantity<=0
                        DISPLAY message
                        INPUT quantity
                INITIALIZE success = True
        EXCEPT
                DISPLAY message
return quantity
END FUNCTION quantity_validation(dic,update,laptop_Id):


DEFINE FUNCTION sell_bill(name,list_brand,list_laptop,total_price:
CALL year = datetime.datetime.now().year
CALL month = datetime.datetime.now().month
CALL day = datetime.datetime.now().day
```

```
CALL hour = datetime.datetime.now().hour

CALL minute = datetime.datetime.now().minute

CALL second = datetime.datetime.now().second


CALL unique = str(hour)+str(minute)+str(second)

CALL dateAndTime = str(year)+"/"+str(month)+"/"+str(day)+"
"+str(hour)+":"+str(minute)+":"+str(second)


CALL file = open(f"{name} {unique}.txt","w")

CALL file.write("Name of the customer is: "+name+"\n")

CALL file.write("Date and Time : "+dateAndTime+"\n")

CALL file.write("Total Price:"+str(total_price)+"\n")


FOR i in range(len(list_brand)):

        CALL file.write(list_laptop[i]+": "+list_brand[i]+"\n")

CALL file.close()


DIPLAY message

END FUNCTION sell_bill(name,list_brand,list_laptop,total_price):


DEFINE FUNCTION buy_bill(name,list_brand,list_laptop,total_price:

CALL year = datetime.datetime.now().year

CALL month = datetime.datetime.now().month

CALL day = datetime.datetime.now().day
```

```
CALL hour = datetime.datetime.now().hour

CALL minute = datetime.datetime.now().minute

CALL second = datetime.datetime.now().second


CALL unique = str(hour)+str(minute)+str(second)

CALL dateAndTime = str(year)+"/"+str(month)+"/"+str(day)+"
"+str(hour)+":"+str(minute)+":"+str(second)


CALL fileName = name+unique+"-return.txt"

CALL file = open(fileName,"w")

CALL file.write("Name of the customer is: "+name+"\n")

CALL file.write("Date and Time : "+dateAndTime+"\n")

CALL file.write("Total Price:"+str(total_fine)+"\n")


FOR i in range(len(list_brand)):

        CALL file.write(list_laptop[i]+": "+list_brand[i]+"\n")

CALL file.close()

DISPLAY message

END FUNCTION buy_bill(name,list_brand,list_laptop,total_price:
```

Module:laptop_read.py

```
DEFINE FUNCTION dict_laptop():

        CALL file = open("stock.txt","r")

        CREATE dictionary = {}
```

```
    CALL key = 0


    FOR line in file:
        CALL key = key + 1
        CALL line = line.replaced("\n","")
        CALL line = line.split(",")
        CALL dictionary[key] = line
    CALL file.close()
    return dicionary
END FUNCTION dict_laptop():


DEFINE FUNCTION display_table():
    CALL dictionary = dict_laptop()
    Open the file stock.txt in read mode and store in variable named as files
        DISPLAY message
        DISPLAY message
        DISPLAY message
        FOR sn_num, data_dic in dictionary.items():
            CALL sn_num, data_dic in dictionary.items():
            CALL sn = str(sn_num).rjust(2, ' ')
            CALL company = data_dic [1].ljust(15)
            CALL product = data_dic[0].ljust(18)
            CALL price = data_dic[2].ljust(10)
```

```
        CALL quantity = data_dic[3].ljust(10)
        CALL Processor = data_dic[4].ljust(15)
        CALL graphics = data_dic[5].ljust(10)
        DISPLAY message
END FUNCTION display_table():
```

5. Data Structures:

Data structures are methods of arranging and storing data in a computer's memory so that it may be efficiently accessed and manipulated. There are many different types of data structures, each with their own set of advantages and disadvantages, and selecting the proper one can be important to a program's performance and efficiency.

- Primitive Data structure:
  a. Boolean: A Boolean data type represents logical values like true or false. This has been used in class to help with loops and while conditions.
     Example: choice = True

  b. Integer: An integer is a data type that represents a whole number, that is a number that does not contain any fractions or decimals. Positive, negative, or zero integers are all possible. It is also required for loops and pricing declarations.
     Example: R = 1

  c. String: A string is data type that represents a character sequence.
     It is used to store words, characters, and alphabets in this course. It's also found in printed statements.
     Example: Str = "Rounak"

d. Float: The float data type is used to hold numeric numbers as well as decimal values. It has been utilized to calculate prices in this course.
   Example: Float: 1.2

- Compound Data Structure:
  a. List: A list is a collection of elements, each with a unique data type. A list's element are not always stored in contiguous memory locations and they can be added or removed dynamically. In this course, it is utilized to store components inside square brackets.
     Example: laptop_ = [ ]

  b. Tuple: A tuple is a compound data structure that represents a collection of distinct sorts of elements. This is similar to lists, except that the data entered in a tuple cannot be changed.
     Example: (30, 40, 50)

  c. Dictionary: A dictionary is a data structure that contains a collection of key-value pairs.
     Example: dictionary = {}

  d. Sets: An unordered collection of distinct items. Sets are comparable to mathematical sets in that no duplicates exist and the elements are not arranged.
     Example: my_set = {1, 5, 7}

```
dictionary = {}
key = 0
```

Figure 1: use of dictionary

```
choice = True
while choice == True:
   #displays choices
    messages.choices()
    a = False
    while a == False:
        try:
            # taking an input
            choice = int(input("Enter a choice: "))
            a = True
```

Figure 2: use of Boolean and Integer

6. Program

- Introduction: This code is programmed to be a user-friendly buying systems for laptops. It has multiple laptops in stock. While using the program, the user can access the laptops in stock, buy multiple laptops if they are available, sell laptops they have been bought, charge a fine in case of expiry of the sell date. The buying period is 5 days. Returns any day later will be charged fine accordingly. To keep track of the laptops, buy , and sell .txt files are created which have information of the client, laptops, and the date and time of the transaction.
- Buy and sell process:

  Buy: When the input is set to 2, the buying process begins, and the laptops are displayed in the application. It checks to see if the selected option is available before printing the message. The program requests the buyer's name. The user is then asked if they wish to purchase additional laptops. If the user responds yes, the software prints the display message again and begins a loop if the user says no, the program loops from the beginning until it is excited.

```
Select your desirable choice
(1) || Press 1 to sell a laptop.
(2) || Press 2 to buy a laptop.
(3) || Press 3 to exit.


Enter a choice: 2


Please buy accordingly
==================================================================================================
S.N Laptop name        Company          Price      Quantity   Processor       Graphics
==================================================================================================
 1.  Razer Blade        Razer           $2000     6           i7 7th Gen      GTX 3060
 2.  XPS                Dell            $1976     8           i5 9th Gen      GTX 3070
 3.  Alienware          ALienware       $1978     16          i5 9th Gen      RTX 4070
 4.  Swift 7            Acer            $900      8           i5 9th Gen      RTX 4090
 5.  Macbook Pro 16     Apple           $3500     15          i5 9th Gen      RTX 3090
Enter the desired laptop ID: 5


-----------------------------------------
      The selected laptop is available
-----------------------------------------


Enter the required quantity:1
Enter the name of the buyer: Rounak

 Do you want to buy more laptops?
Type yes if you want to else type no:no
-------------------------------------------------------
                        Invoice for buy
-------------------------------------------------------
Date and Time: 2023/5/12 10:44:57
Name of the customer: Rounak
Brand is: [' Apple']
laptop is: [' Macbook Pro 16']
Total price: 3500.0


Select your desirable choice
(1) || Press 1 to sell a laptop.
```

Figure 3: Buy Process

Sell: When the input 1 is entered, the sell process begins. The laptops are displayed in the program. It determines whether the selected items is for sale and then publishes the appropriate message. An invoice is generated after the user enters his or her name and the dates of purchase.

```
Python 3.10.2 (v3.10.2:a58ebcc701, Jan 13 2022, 14:50:16) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: /Users/ronakgod/Downloads/22066975 Rounak pradhan python coursework/development/main.py
-----------------------------------------------------
       Welcome to the pradhan electronics
-----------------------------------------------------


Select your desirable choice
(1) || Press 1 to sell a laptop.
(2) || Press 2 to buy a laptop.
(3) || Press 3 to exit.


Enter a choice: 1


The items available for sell is displayed below:
==========================================================================================================
S.N Laptop name        Company          Price      Quantity   Processor       Graphics
==========================================================================================================
 1.  Razer Blade        Razer           $2000      3          i7 7th Gen      GTX 3060
 2.  XPS                Dell            $1976      8          i5 9th Gen      GTX 3070
 3.  Alienware          ALienware       $1978      20         i5 9th Gen      RTX 4070
 4.  Swift 7            Acer            $900       8          i5 9th Gen      RTX 4090
 5.  Macbook Pro 16     Apple           $3500      5          i5 9th Gen      RTX 3090
Enter the desired laptop ID: 3


-------------------------------------------
      The selected laptop is available
-------------------------------------------


Enter the  required quantity: 2
Enter the name of the buyer: Rounak

 Do you want to sell more laptops?
Type yes if you want to else type no:no
-----------------------------------------------------
                        Invoice for sell
-----------------------------------------------------
Date and Time: 2023/5/10 16:14:4
Name of the customer: Rounak
Brand Name: [' ALienware']
laptop Name : [' Alienware']
Total Price: 3956.0


Select your desirable choice
(1) || Press 1 to sell a laptop.
(2) || Press 2 to buy a laptop.
(3) || Press 3 to exit.


Enter a choice: |
```

Figure 4: Sell Process

Textfiles: This application has generated a number of text files. A text file is available for recording costumes. After each rent and return transaction, text files are also created. It is critical to construct these text files in order to maintain track of the records. These text files keep track of the stock, the date and time of transactions, and the sale information.

```
Razer Blade, Razer, $2000,3, i7 7th Gen, GTX 3060
XPS, Dell, $1976,8, i5 9th Gen, GTX 3070
Alienware, ALienware, $1978,22, i5 9th Gen, RTX 4070
Swift 7, Acer, $900,8, i5 9th Gen, RTX 4090
Macbook Pro 16, Apple, $3500,5, i5 9th Gen, RTX 3090
```

Figure 5: Stock text file

```
-----------------------------------------------------
                        Invoice for sell
-----------------------------------------------------
Date and Time: 2023/5/10 18:46:20
Name of the customer: Ram
Brand Name: [' Razer']
laptop Name : [' Razer Blade']
Total Price: 4000.0
```

Figure 6: Sell invoice

```
-------------------------------------------------------
                        Invoice for buy
-------------------------------------------------------
Date and Time: 2023/5/10 18:46:55
Name of the customer: Shyam
Brand is: [' Apple']
laptop is: [' Macbook Pro 16']
Total fine: 7000.0
```

Figure 7: Buy Invoice

7. Testing

7.1. Test1:

Table 2: Test 1

Objective	Show the implementation of try and except
Action	Enter a number between 1-3: 5
Expected result	Invalid input, please select one of the options provided.
Actual result	Invalid input, please select one of the options provided.
Conclusion	Test successful

```
-------------------------------------------------------
       Welcome to the pradhan electronics
-------------------------------------------------------


Select your desirable choice
(1) || Press 1 to sell a laptop.
(2) || Press 2 to buy a laptop.
(3) || Press 3 to exit.


Enter a choice: 5
-------------------------------------------------------
   Invalid input, Please choose a valid choice.
-------------------------------------------------------
```

Figure 8: Test 1

7.2. Test 2:

Table 3: Test 2

Objective	To obtain the relevant message, enter negative and non-existed values.
Action	-Input negative value as input. Select an option: -2 -Input non-existent value as input Enter your first name: mac
Expected result	Printing appropriate messages accordingly.
Actual result	Printing appropriate messages accordingly.
Test	Test successful.

```
Select your desirable choice
(1) || Press 1 to sell a laptop.
(2) || Press 2 to buy a laptop.
(3) || Press 3 to exit.


Enter a choice: 1


The items available for sell is displayed below:
==========================================================================================================
S.N Laptop name        Company          Price      Quantity   Processor       Graphics
==========================================================================================================
 1.  Razer Blade        Razer           $2000     7           i7 7th Gen      GTX 3060
 2.  XPS                Dell            $1976     8           i5 9th Gen      GTX 3070
 3.  Alienware          ALienware       $1978     26          i5 9th Gen      RTX 4070
 4.  Swift 7            Acer            $900      8           i5 9th Gen      RTX 4090
 5.  Macbook Pro 16     Apple           $3500     15          i5 9th Gen      RTX 3090
Enter the desired laptop ID: -2
Invalid laptop ID, Please enter a valid ID
Enter laptop ID: mac
Invalid input, Please enter a valid input
Enter the desired laptop ID: |
```

Figure 9: Test 2

```
The items available for sell is displayed below:
==========================================================================================================
S.N Laptop name        Company          Price      Quantity   Processor       Graphics
==========================================================================================================
 1.  Razer Blade        Razer           $2000     7           i7 7th Gen      GTX 3060
 2.  XPS                Dell            $1976     8           i5 9th Gen      GTX 3070
 3.  Alienware          ALienware       $1978     26          i5 9th Gen      RTX 4070
 4.  Swift 7            Acer            $900      8           i5 9th Gen      RTX 4090
 5.  Macbook Pro 16     Apple           $3500     15          i5 9th Gen      RTX 3090
Enter the desired laptop ID: 1


-------------------------------------------
      The selected laptop is available
-------------------------------------------


Enter the  required quantity: -2
Invalid input for quantity
Enter the quantity:mac
Invalid input, Please input the correct data
Enter the  required quantity:
```

Figure 10: Test 2.1

7.3. Test 3:

Table 4: Test 3

Objective	Show buy process
Action	-buy process is choosen Enter a number between 1-4: 1 -The name of buyer is entered Name : Rounak -Bill is generated with an appropriate fine in case of late buy
Expected result	Laptops are expected to be display successfully and a Rounak.unique-return.txt file is expected to be generated.
Actual result	Laptops are expected to be display successfully and a Rounak.unique-return.txt file is expected to be generated.
Test	Test successful.

```
Please buy accordingly
==================================================================================================
S.N Laptop name        Company          Price      Quantity   Processor       Graphics
==================================================================================================
 1.  Razer Blade        Razer           $2000     10          i7 7th Gen      GTX 3060
 2.  XPS                Dell            $1976     8           i5 9th Gen      GTX 3070
 3.  Alienware          ALienware       $1978     26          i5 9th Gen      RTX 4070
 4.  Swift 7            Acer            $900      8           i5 9th Gen      RTX 4090
 5.  Macbook Pro 16     Apple           $3500     15          i5 9th Gen      RTX 3090
Enter the desired laptop ID: 1


--------------------------------------------
      The selected laptop is available
--------------------------------------------


Enter the required quantity:2
Enter the name of the buyer: Rounak

 Do you want to buy more laptops?
Type yes if you want to else type no:no
-------------------------------------------------------
                          Invoice for buy
-------------------------------------------------------
Date and Time: 2023/5/10 21:32:19
Name of the customer: Rounak
Brand is: [' Razer']
laptop is: [' Razer Blade']
Total price: 4000.0
```

Figure 11: Test 3

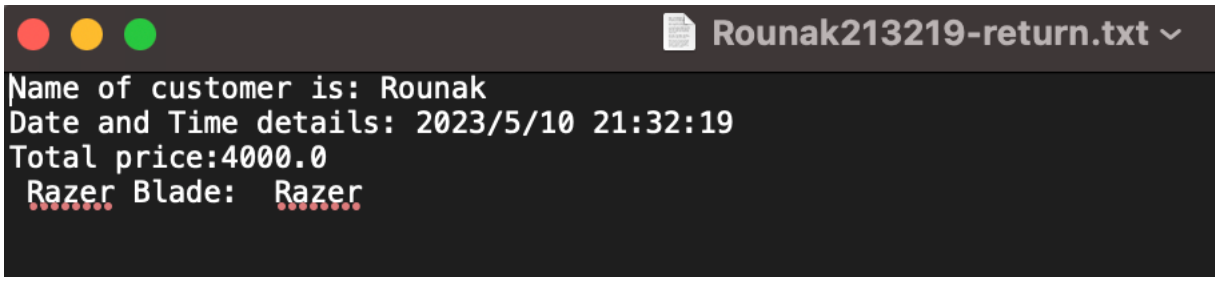


Figure 12: Rounak.unique-return.txt

7.4. Test 4:

Table 5: Test 4

Objective	Show sell process
Action	-Sell process is choosen Enter a number between 1-4: 1 -The name of the seller is entered Name = Ronak -A laptop is sold
Expected result	Laptops are expected to be sold successfully and a Ronak.unique-b
Actual result	
Test	Test successful.

```
The items available for sell is displayed below:
==========================================================================================================
S.N Laptop name        Company          Price      Quantity   Processor       Graphics
==========================================================================================================
 1.  Razer Blade        Razer           $2000     6           i7 7th Gen      GTX 3060
 2.  XPS                Dell            $1976     8           i5 9th Gen      GTX 3070
 3.  Alienware          ALienware       $1978     26          i5 9th Gen      RTX 4070
 4.  Swift 7            Acer            $900      8           i5 9th Gen      RTX 4090
 5.  Macbook Pro 16     Apple           $3500     15          i5 9th Gen      RTX 3090
Enter the desired laptop ID: 1


-------------------------------------------
      The selected laptop is available
-------------------------------------------


Enter the  required quantity: 2
Enter the name of the seller: Ronak

 Do you want to sell more laptops?
Type yes if you want to else type no:no
-------------------------------------------------------
                         Invoice for sell
-------------------------------------------------------
Date and Time: 2023/5/10 21:56:5
Name of the customer: Ronak
Brand Name: [' Razer']
laptop Name : [' Razer Blade']
Total cost: 4000.0
```

Figure 13: Test 4

```
Ronak 21565.txt
Name of the customer is: Ronak
Date and Time : 2023/5/10 21:56:5
Total Price:4000.0
 Razer Blade:  Razer
```

Figure 14: Ronak.unique.txt

7.5. Test 5:

Table 6: Test 5

Objective	Show update in stock
Action	-Sell a laptops -Check stock -Buy the laptops -check the laptops
Expected result	Stock is expected to be updated
Actual result	Stock is updated
Test	Test successful.

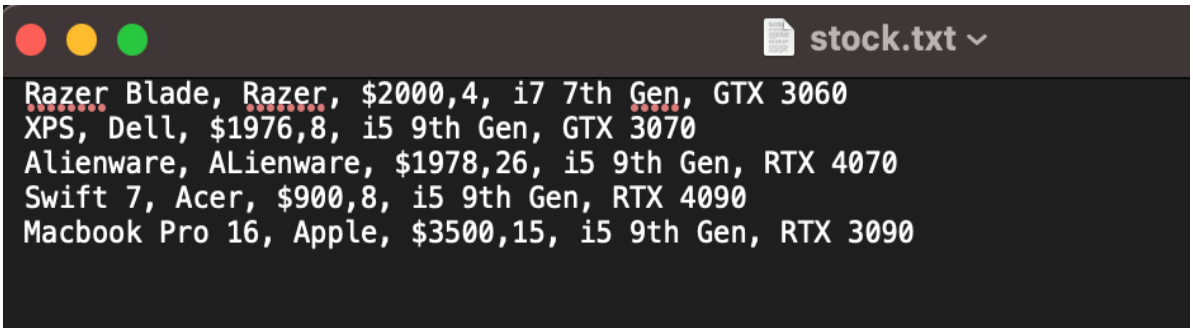


Figure 15: Before selling

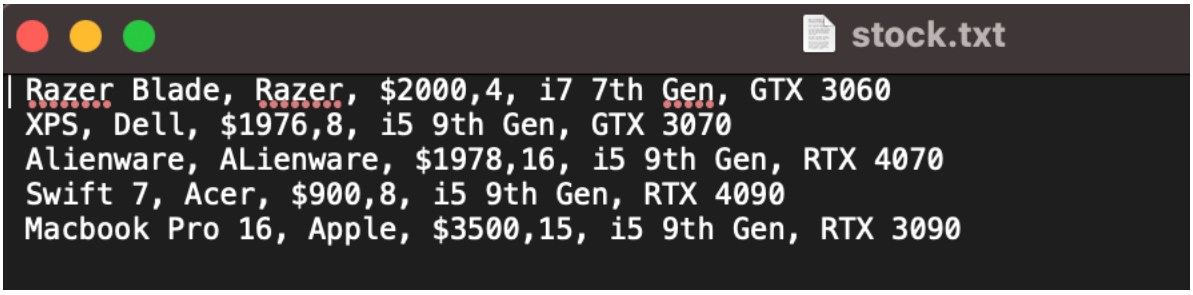


Figure 16: After selling

```
stock.txt
Razer Blade, Razer, $2000,14, i7 7th Gen, GTX 3060
XPS, Dell, $1976,8, i5 9th Gen, GTX 3070
Alienware, ALienware, $1978,16, i5 9th Gen, RTX 4070
Swift 7, Acer, $900,8, i5 9th Gen, RTX 4090
Macbook Pro 16, Apple, $3500,15, i5 9th Gen, RTX 3090
```

Figure 17: After buying

8. Conclusion

- This is essentially a Pradhan electronics course. It maintains track of its inventory, presents it, allows the client to sell laptops if they are available, purchase the laptops, charge a late fee, and calculates the price. It is also meant to generate unique.txt files for each laptop purchased or sold, containing information such as the buyer's name, laptop model, and the date and time of the transaction.

- The code was done in modules. There are a total of five.py files. After the appropriate files have been imported into the main file, the procedure is called. Modularization simplifies code comprehension and writing for both the programmer and the reader. To facilitate comprehension and explanation, the code has been marked with comments.

- I attempted to keep the code as simple and efficient as possible when working on this course. I worked on both the file display and the main software. The display is basic, user-friendly, and easy to understand and apply. The code will loop until you provide acceptable input to exit the program.

- Throughout the semester of the coursework, I learned a lot. Flowcharts, pseudocode, and algorithms were all unfamiliar concepts to me. To illustrate the code's usefulness and adaptability, multiple tests were run as indicated. As a result of flaws and errors, many challenges were faced during the training. This was how the code was created.

- This course contained a substantial amount of research and findings. I discovered a great deal about data structures. I had to conduct extensive study for both the coding and documentation portions. All of the sources I've used are listed here. To help me, I used information provided by the school as well as materials available online.

9. References

Gill, A. S. (2022, May). *TechTarget*. Retrieved from TechTarget: https://www.techtarget.com/whatis/definition/algorithm

Mahr, N. (2022). *Study.com*. Retrieved from Study.com: https://study.com/learn/lesson/pseudocode-examples-what-is-pseudocode.html

Hebb, N. (202). *BreezeTree Software*. Retrieved from BreezeTree: https://www.breezetree.com/articles/what-is-a-flow-chart

Alexander S. Gillis. (2022, May). *TechTarget*. Retrieved from TechTarget: https://www.techtarget.com/whatis/definition/algorithm

stage., A. f. (2020). *Breeze Tree software*. Retrieved from Breeze Tree: https://www.breezetree.com/articles/what-is-a-flow-chart

Mahr, N. (2022). *study.com*. Retrieved from study.com: https://study.com/learn/lesson/pseudocode-examples-what-is-pseudocode.html

10. Appendix

Module: main.py

```
#Importing files
import laptop_read
import messages
import update_dict
import functions

#prints the welcome message
messages.welcome_message()

#displays the given choices
```

```
    choice = True
    while choice == True:
      #displays choices
       messages.choices()
       a = False
       while a == False:
          try:
             # taking an input
             choice = int(input("Enter a choice: "))
             a = True
          except ValueError:
             print("Invalid input, Please choose among the given choices.")


       #choice to sell a laptop
       if choice == 1:
          messages.choice_one()
          laptop_read.display_table()
          dict_update = laptop_read.dict_laptop()


          laptop_Id = functions.checking_laptop_Id(dict_update)
       #creating list
          laptop_list = []
          laptop_brand = []
```

```
        if int(dict_update[laptop_Id][3])>0: #checking if the input item is >0
            quantity = functions.correct_quantity(dict_update,laptop_Id) # validating quantity
            dict_update[laptop_Id][3]= int(dict_update[laptop_Id][3]) - quantity #updating dict


            update_dict.laptop_update(dict_update)


           # adding value to the list
            laptop_list.append(dict_update[laptop_Id][0])
            laptop_brand.append(dict_update[laptop_Id][1])


            totalPrice = functions.dollar_renewed(dict_update[laptop_Id][2],quantity)


            while(True):
                name = input("Enter the name of the seller: ")
                if 1==1: #checking if the entered value only consists of alphabet
                    break
                print("Invalid input. Please input alphabets only.")


            print("\n Do you want to sell more laptops?")
            Loop = input("Type ""yes"" if you want to else type ""no"":")
```

```
Loop_extension = True
while Loop_extension == True:

    if Loop.lower() == "yes":
        print()
        laptop_read.display_table()

        laptop_Id = functions.checking_laptop_Id(dict_update)
        if int(dict_update[laptop_Id][3])>0:
            quantity = functions.correct_quantity(dict_update,laptop_Id)
            dict_update[laptop_Id][3]= int(dict_update[laptop_Id][3]) - quantity

            update_dict.laptop_update(dict_update)



            for i in laptop_list:

                if i != dict_update[laptop_Id][0]:
                    laptop_list.append(dict_update[laptop_Id][0])
            for  j in laptop_brand:
                if j != dict_update[laptop_Id][1]:
                    laptop_brand.append(dict_update[laptop_Id][1])
```

```
                    updateSum =
functions.dollar_renewed(dict_update[laptop_Id][2],quantity)

                    totalPrice = totalPrice + updateSum

                    print("\nDo you want to sell another laptop?")
                    Loop = input("Type ""yes"" if you want to else type ""no"": ")

                    Loop_extension = True
                 else:
                     print(" laptop is not cursellly available ")
                     print("\n Do you want to sell another laptop?")
                     Loop = input("Type ""yes"" if you want to else type ""no"":")
                     Loop_extension = True

              else:
                 Loop_extension = False

           functions.sell_bill(name,laptop_brand,laptop_list,totalPrice)
```

```
    #choice to buy a laptop
    elif choice == 2:
        messages.choice_two()
        laptop_read.display_table()
        dict_update = laptop_read.dict_laptop()


        laptop_Id = functions.checking_laptop_Id(dict_update)
    #creating list
        laptop_list = []
        laptop_brand = []


        if int(dict_update[laptop_Id][3])>=0: #checking if the input item is >0
            quantity = functions.quantity_validation(dict_update,laptop_Id) # validating
quantity
            dict_update[laptop_Id][3]= int(dict_update[laptop_Id][3]) + quantity #updating
dict


            update_dict.laptop_update(dict_update)


           # adding value to the list
            laptop_list.append(dict_update[laptop_Id][0])
            laptop_brand.append(dict_update[laptop_Id][1])
```

```
totalPrice = functions.dollar_renewed(dict_update[laptop_Id][2],quantity)


while(True):
    name = input("Enter the name of the buyer: ")
    if 1==1: #checking if the entered value only consists of alphabet
        break
    print("Invalid input. Please input alphabets only.")


print("\n Do you want to buy more laptops?")
Loop = input("Type ""yes"" if you want to else type ""no"":")


Loop_extension = True
while Loop_extension == True:


    if Loop.lower() == "yes":
        print()
        laptop_read.display_table()


        laptop_Id = functions.checking_laptop_Id(dict_update)
        if int(dict_update[laptop_Id][3])>=0:
            quantity = functions.quantity_validation(dict_update,laptop_Id)
            dict_update[laptop_Id][3]= int(dict_update[laptop_Id][3]) + quantity
```

```
                    update_dict.laptop_update(dict_update)



                    for i in laptop_list:


                        if i != dict_update[laptop_Id][0]:

                            laptop_list.append(dict_update[laptop_Id][0])

                    for  j in laptop_brand:

                        if j != dict_update[laptop_Id][1]:

                            laptop_brand.append(dict_update[laptop_Id][1])


                    updateSum                                                    =
functions.dollar_renewed(dict_update[laptop_Id][2],quantity)


                    totalPrice = totalPrice + updateSum


                    print("\nDo you want to buy another laptop?")

                    Loop = input("Type ""yes"" if you want to else type ""no"": ")


                    Loop_extension = True

                else:

                     print(" laptop is not cursellly available ")

                     print("\n Do you want to buy another laptop?")
```

```
                    Loop = input("Type ""yes"" if you want to else type ""no"":")
                    Loop_extension = True


                else:
                    Loop_extension = False


            functions.buy_bill(name,laptop_brand,laptop_list,totalPrice)
            choice = True
        else:
            print("laptop is not available")



    elif choice == 3:
        messages.choice_three()
        choice = False


    else:
        messages.invalid()
        choice = True
```

Module: functions.py

```
import datetime
#price for sell
def dollar_renewed(dollar_sign,quantity):
    price = float(dollar_sign.replace("$",""))
    total = price * quantity
    return total


def checking_laptop_Id(dictionary):
    success = False
    while success == False:
        try:


            laptop_Id = int(input("Enter the desired laptop ID: "))


            while laptop_Id<=0 or laptop_Id>len(dictionary):
                print("Invalid laptop ID, Please enter a valid ID")


                laptop_Id = int(input("Enter laptop ID: "))


            print("\n")
            print("-------------------------------------------")
            print("      The selected laptop is available    ")
            print("-------------------------------------------")
```

```
            print("\n")


            success = True
        except ValueError:
            print("Invalid input, Please enter a valid input")


    return laptop_Id




def correct_quantity(dictionary,laptop_Id):
    success = False
    while success == False:
        try:
            quantity = int(input("Enter the  required quantity: "))


            while quantity<=0 or quantity>int(dictionary[laptop_Id][3]):
                if quantity <=0:
                    print("Invalid input for quantity")
                else:
                    print("Quantity entered exceeds the available stock!!!")
                quantity = int(input("Enter the quantity:"))
            success = True
```

```
        except ValueError:
            print("Invalid input, Please input the correct data")

    return quantity

def quantity_validation(dict_update, laptop_ID):
    success = False
    while success == False:
        try:
            quantity = int(input("Enter the required quantity:"))

            while quantity<=0:
                print("Invalid input for quantity")

                quantity = int(input("Enter the quantity:"))

            success = True
        except:
            print("Invalid input, Please input the correct data")

    return quantity
```

```
#creating a sell invoice
def sell_bill(name,list_brand,list_laptop,total_price):
    year = datetime.datetime.now().year
    month = datetime.datetime.now().month
    day = datetime.datetime.now().day
    hour = datetime.datetime.now().hour
    minute = datetime.datetime.now().minute
    second = datetime.datetime.now().second



    unique = str(hour)+str(minute)+str(second)
    dateAndTime = str(year)+"/"+str(month)+"/"+str(day)+"
"+str(hour)+":"+str(minute)+":"+str(second)
    # creating a sell text file
    file = open(f"{name} {unique}.txt","w")
    file.write("Name of the customer is: "+name+"\n")
    file.write("Date and Time : "+dateAndTime+"\n")
    file.write("Total Price:"+str(total_price)+"\n")



    for i in range(len(list_brand)):
```

```
        file.write(list_laptop[i]+": "+list_brand[i]+"\n")


    file.close()


    # printing the sell invoice in the terminal
    print("-----------------------------------------------------")
    print("\t\t\t","Invoice for sell ")
    print("-----------------------------------------------------")
    print("Date and Time:",dateAndTime)
    print("Name of the customer:",name)
    print("Brand Name:",list_brand)
    print("laptop Name :",list_laptop)
    print("Total cost:",total_price)



#creating a buy invoice


def buy_bill(name,list_brand,list_laptop,total_fine):
    year = datetime.datetime.now().year
    month = datetime.datetime.now().month
    day = datetime.datetime.now().day
    hour = datetime.datetime.now().hour
    minute = datetime.datetime.now().minute
```

```
    second = datetime.datetime.now().second


    unique = str(hour)+str(minute)+str(second)

    dateAndTime = str(year)+"/"+str(month)+"/"+str(day)+"
"+str(hour)+":"+str(minute)+":"+str(second)


   #creating a buy text file

    fileName = name+unique+"-return.txt"

    file = open(fileName,"w")

    file.write("Name of customer is: "+name+"\n")

    file.write("Date and Time details: "+dateAndTime+"\n")


    file.write("Total price:"+str(total_fine)+"\n")


    for i in range(len(list_brand)):

        file.write(list_laptop[i]+": "+list_brand[i]+"\n")


    file.close()


    # printing the buy invoice in the terminal

    print("-----------------------------------------------------")

    print("\t\t\t","Invoice for buy ")

    print("-----------------------------------------------------")
```

```
    print("Date and Time:",dateAndTime)
    print("Name of the customer:",name)
    print("Brand is:",list_brand)
    print("laptop is:",list_laptop)
    print("Total price:",total_fine)
```

Module: messages.py

```
def welcome_message():
    print("-----------------------------------------------------")
    print("       Welcome to the pradhan electronics       ")
    print("-----------------------------------------------------")

def choices():
    print("\n")
    print("Select your desirable choice")
    print("(1) || Press 1 to sell a laptop.")
    print("(2) || Press 2 to buy a laptop.")
    print("(3) || Press 3 to exit.")
    print("\n")

def choice_one():
    print("\n")
```

```
    print("The items available for sell is displayed below:")


def choice_two():
    print("\n")
    print("Please buy accordingly")


def choice_three():
    print("\n")
    print("Thank You for visiting us.")


def invalid():
    print("-----------------------------------------------------")
    print("   Invalid input, Please choose a valid choice. ")
    print("-----------------------------------------------------")
    print("\n")
```

Module: laptop_read.py

```
def dict_laptop():
    file = open("stock.txt","r")
    dictionary = {}
    key = 0


    for line in file:
```

```
        key = key+1
        line = line.replace("\n","")
        line = line.split(",")
        dictionary[key] = line

    file.close()
    return  dictionary

def display_table():
    dictionary = dict_laptop()
    with open ("stock.txt","r") as files:

print("==========================================================
=================================================")
        print(f"S.N  {'Laptop  name':<18}  {'Company':<16}  {'Price':<10}  {'Quantity':<10}
{'Processor':<15} {'Graphics':<10}")

print("==========================================================
=================================================")
        for sn_num, data_dic in dictionary.items():
            sn = str(sn_num).rjust(2, ' ')
            company = data_dic[1].ljust(15)
            product = data_dic[0].ljust(18)
            price = data_dic[2].ljust(10)
```

```
        quantity = data_dic[3].ljust(10)
        Processor = data_dic[4].ljust(15)
        Graphics = data_dic[5].ljust(10)
        print(f"{sn}. {product} {company} {price} {quantity} {Processor} {Graphics}")
```

Module: update_dict.py

```
def laptop_update(dictionary):
    file = open("stock.txt","w")
    for i in dictionary.values():
        file.write(i[0]+","+i[1]+","+str(i[2])+","+str(i[3])+","+str(i[4])+","+str(i[5]))
        file.write("\n")
    file.close()
```