LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC4051NI Fundamentals of Computing

Assessment Weightage & Type

60% Individual Coursework

Year and Semester

2020-21 Spring

Student Name: Aashna Shrestha

Group: C13

London Met ID: 20048800

College ID: NP01CP4S210103

Assignment Due Date: 10th September 2021

Assignment Submission Date: 10th September 2021

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

Table of Contents

List of Figures

List of Tables

1. Introduction

Libraries have been using written approach to record the transaction details. This method is inefficient when the user needs to find each transaction detail manually. In addition to that, management systems are going digital. So, in order to make the process of recording and accessing the transaction details easier and more efficient, this coursework has been assigned to us on the module Fundamentals of Computing.

As a more appropriate method, the coursework assigned us to develop a management system for a library following the file-based approach. File based approach is a process of storing and organizing data in a file or multiple files. Each file in this system follows a particular format. These systems are built through applications programmed to perform certain tasks such as retrieving an old data, or updating any data without a hassle.

The programming for this application has been done using Python as the programming language and was written in IDLE (Integrated Development and Learning Environment). IDLE is an IDE which is simple and designed for beginners. It has features such as Python shell, auto completion of code, auto indentation and so on (Datacamp, 2020). Along with the program code, the entire library management system has been built through the implementation of algorithm, flowchart and pseudocode. The flowchart has been drawn in an application called Drawio – Diagrams.net. It is a software to draw flowcharts, Entity Relation Diagrams, design databases and create UML. The algorithm and flowchart have been written in this report prepared along with the objectives of the coursework in MS Word.

1.1 Goals and Objectives

The main goal of this coursework is to provide a convenient interface for the user to record the details of transactions when a customer borrows or returns any books in a file, access the records to check the stock and update the details after each transaction.

As for the ones who build the application, the coursework intends to strengthen the concept of Python including modular programming, use of data structures, exception handling, loops and appropriate user interface. It also aims to build an idea of file-based approach to store, retrieve and update data. Furthermore, it makes us work with algorithm and flowchart which helps to provide step by step description about how the program works. Besides, it should help to improve research skills as the design of the management system does not have any limitations and different built-in functions can be used for it. This should help to develop the application creatively as well.

2. Algorithm

Procedure

Step 1: Start

Step 2: Assign input value to select:

- 1. Start/Continue
- 2. End

Step 3: If select == 1 Go to Step 4 Else if select == 2 Go to Step 49 Else Print Error Message

Step 4: Input customer_name

Step 5: Assign input value to option:

- 1. Borrow
- 2. Return
- 3. Display books

Step 6: If option == 1 Go to Step 7 Else Go to Step 24

Step 7: Open file “Library.txt” in read mode

Step 8: Store all items from the file in the list data

Step 9: Display data

Step 10: Get borrow_date and return_date

Step 11: Input num_of_books

Step 12: If number of books is a positive number Go to Step 13 Else Go to Step 11

Step 13: Set books = 0

Step 14: If books < num_of_books Go to Step 15 Else Go to Step 23

Step 15: Input book_name

Step 16: If book_name is found in data and stock of book is available Go to Step 17 Else Go to Step 2

Step 17: Calculate total_amount

Step 18: Decrease the stock in data

Step 19: Set transaction_id = 1

Step 20: If transaction_id exists increment the value of transaction_id Else Go to Step 21

Step 21: Write customer_name, borrow_date, return_date, book_name, total_amount to the file "Transaction_{transaction_id}.txt"

Step 22: Update data in the file "Library.txt"

Step 23: Increment books and Go to Step 14

Step 24: If option == 2 Go to Step 25 Else Go to Step 45

Step 25: Open file "Library.txt" in read mode and store in library_file

Step 26: Store all items from the file in the list library_data

Step 27: Input num_of_books

Step 28: If num_of_books is a positive number Go to Step 29 Else Go to Step 27

Step 29: Set books = 0

Step 30: If books < num_of_books Go to Step 31 Else Go to Step 2

Step 31: Input transaction_id

Step 32: Print data from the file "Transaction_{transaction_id}.txt"

Step 33: Input book_name

Step 34: If book has been returned Go to Step 2 Else Go to Step 35

Step 35: Get current_date

Step 36: If current_date <= return_date Go to Step 37 Else Go to Step 38

Step 37: Calculate amount and Go to Step 39

Step 38: Calculate amount with fine

Step 39: Increment the stock

Step 40: Calculate total_amount

Step 41: If book is found in the transaction file, Go to Step 42 Else Go to Step 2

Step 42: Append customer_name, book_name, return_date, fine, total_amount to the file “Transaction_{transaction_id}.txt”

Step 43: Update data in the file “Library.txt”

Step 44: Increment books and Go to Step 30

Step 45: If option == 3 Go to Step 46 Else Go to Step 2

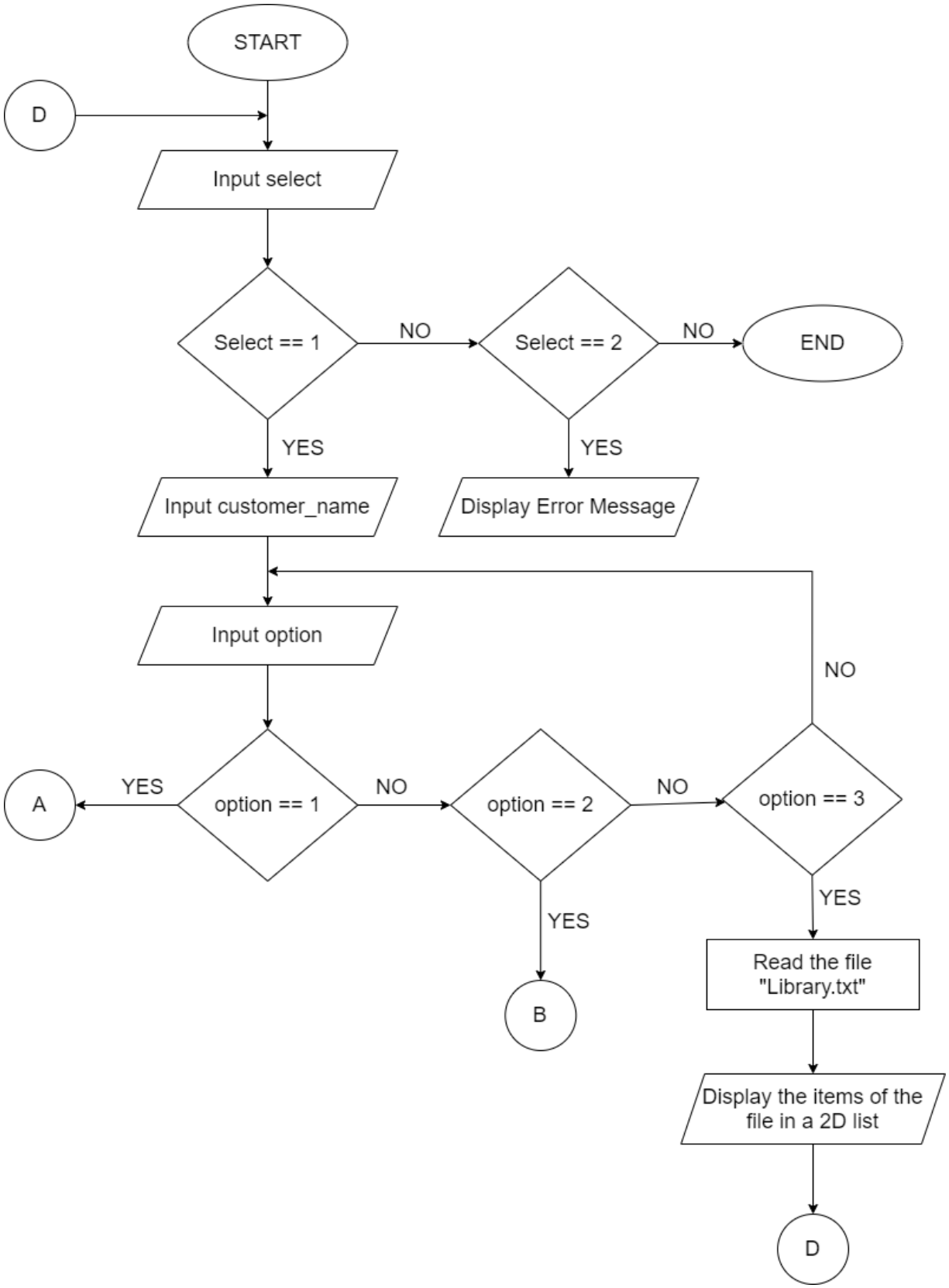
Step 46: Read data from a file

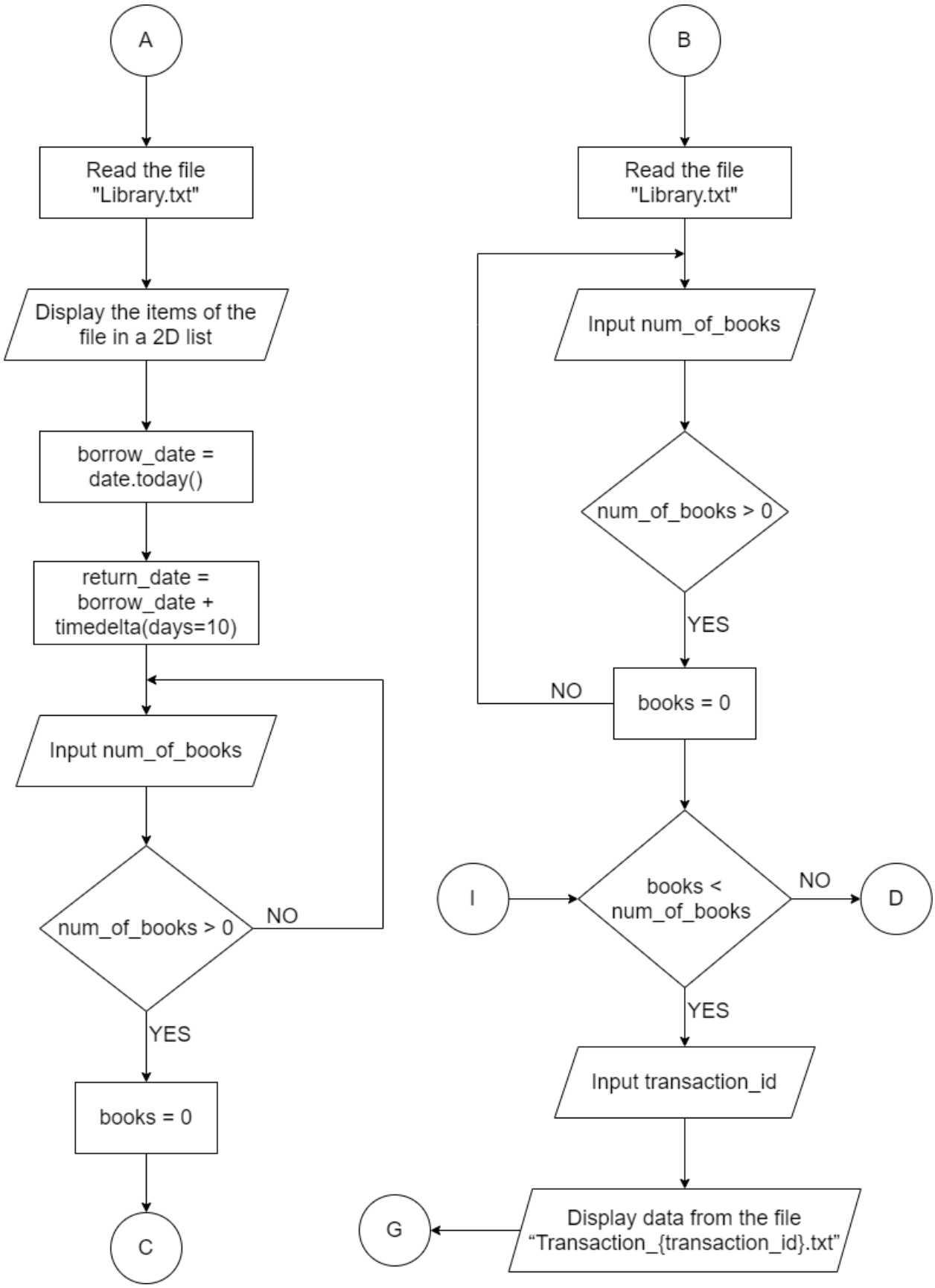
Step 47: Store the data in a 2D list

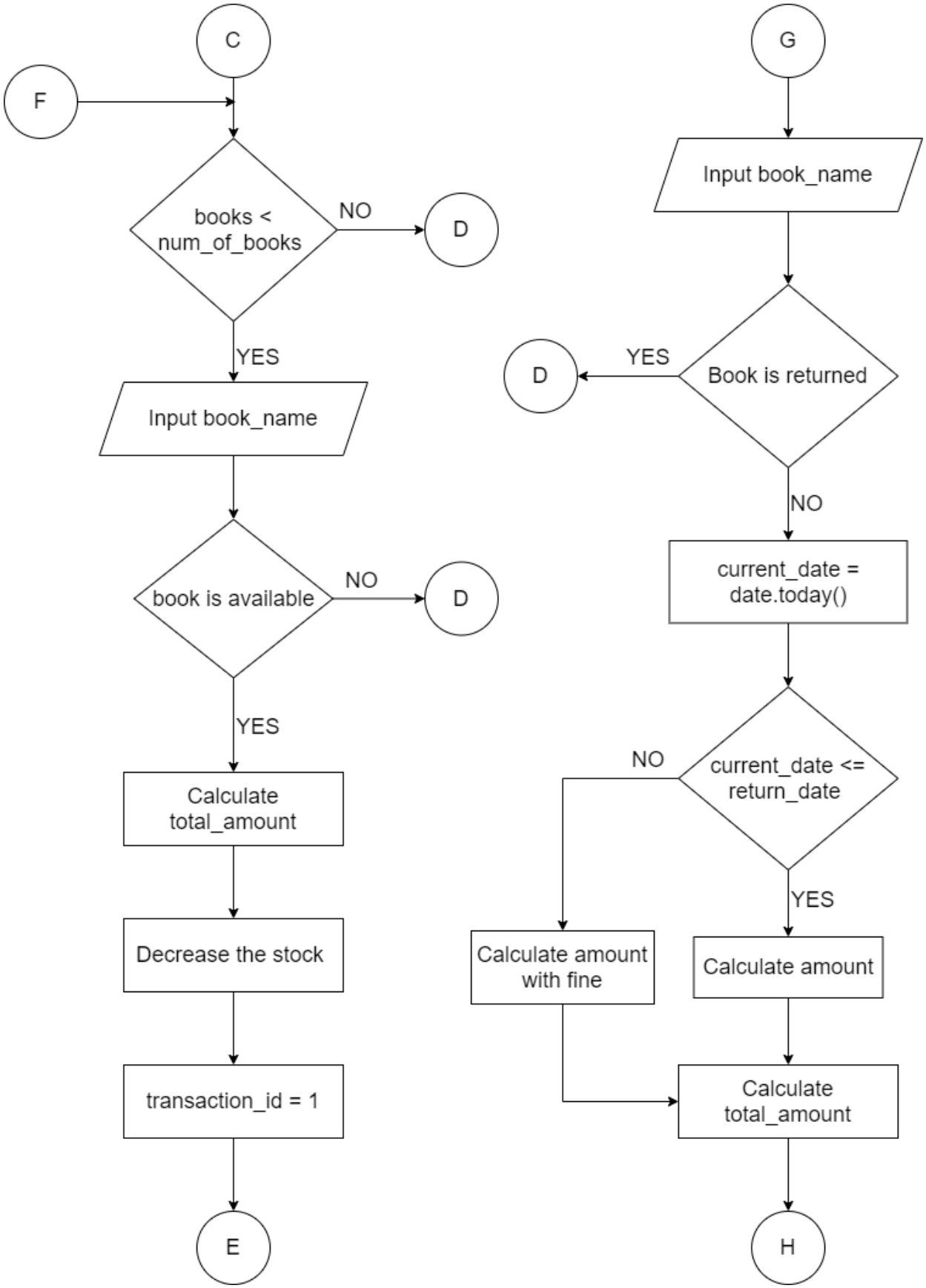
Step 48: Display the list

Step 49: End

3. Flowchart


START
D
Input select
Select == 1
NO
Select == 2
NO
END
YES
YES
Input customer_name
Display Error Message
Input option
NO
A
YES
option == 1
NO
option == 2
NO
option == 3
YES
YES
B
Read the file "Library.txt"
Display the items of the file in a 2D list
D


A
Read the file "Library.txt"
Display the items of the file in a 2D list
borrow_date = date.today()
return_date = borrow_date + timedelta(days=10)
Input num_of_books
num_of_books > 0
NO
YES
books = 0
C
B
Read the file "Library.txt"
Input num_of_books
num_of_books > 0
YES
NO
books = 0
I
books < num_of_books
NO
D
YES
Input transaction_id
Display data from the file "Transaction_{transaction_id}.txt"
G


C
F
books < num_of_books
NO
D
YES
Input book_name
book is available
NO
D
YES
Calculate total_amount
Decrease the stock
transaction_id = 1
E
G
Input book_name
Book is returned
YES
D
NO
current_date = date.today()
current_date <= return_date
NO
YES
Calculate amount with fine
Calculate amount
Calculate total_amount
H

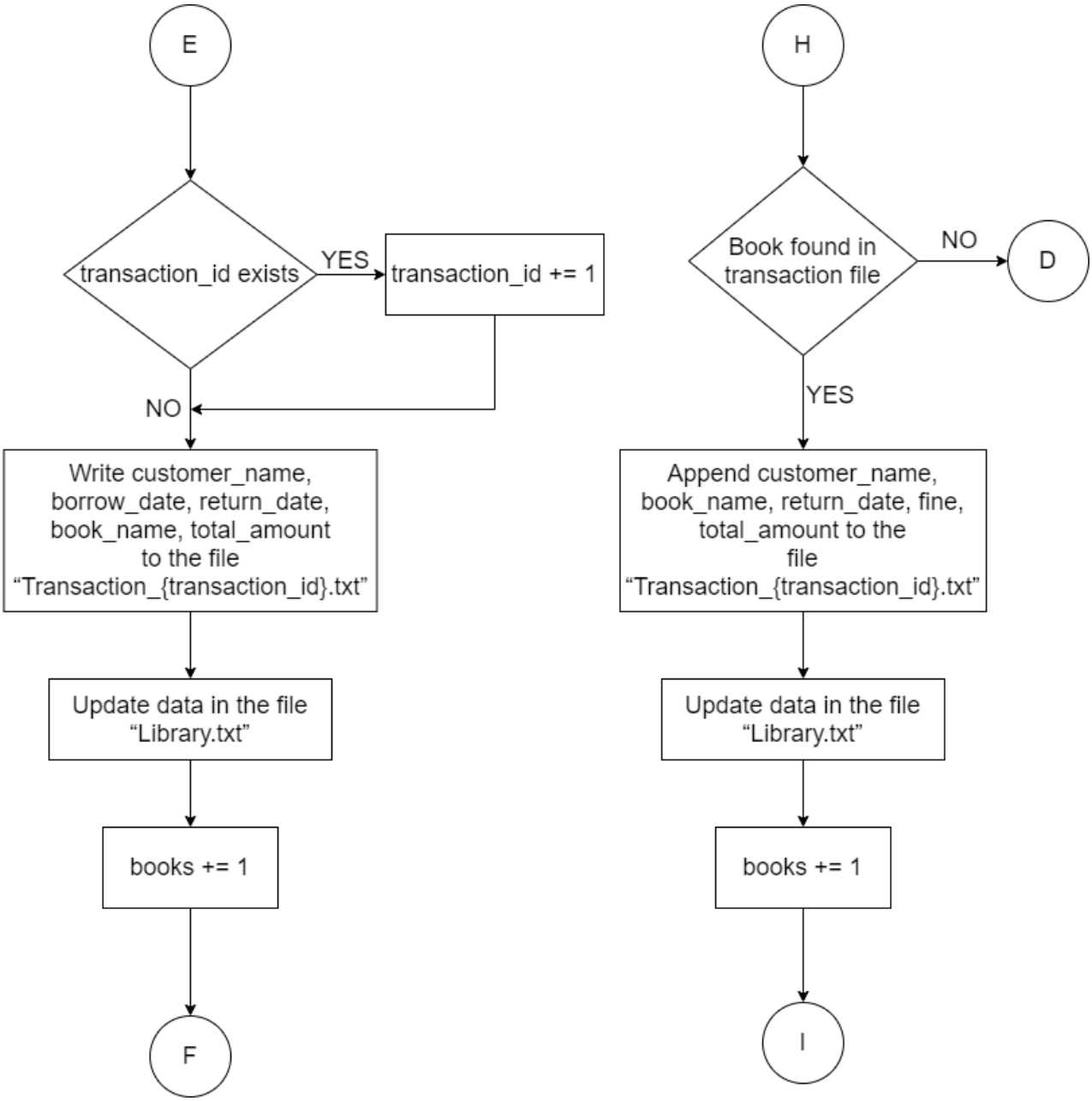


Figure 1 Flowchart of Library Management System

4. Pseudocode

4.1 Module 1 - read_data.py

```
DEFINE Function file_data(file_name)
    OPEN file_name in WRITE mode and store it in file
    READ data from file and store it in data
    CLOSE file
    RETURN data

DEFINE Function data_list(file)
    INITIALIZE a list to data
    FOR each in file
        REPLACE "\n" with ""
        SPLIT ","
        APPEND each by replacing "\n" with "' and splitting "," to data
    END FOR
```

4.2 Module 2 - library_menu.py

```
START
IMPORT * from read_data
DEFINE Function display(data)
    FOR i in range(len(data))
        PRINT (data[i][0])
DEFINE Function borrow(customer_name):
    INITIALIZE the function file_data with the parameter as "Library.txt" to fileData
    INITIALIZE the function data_list with the parameter as fileData to data
    FOR i in range (len(data))
        FOR j in range(2, 4)
            INITIALIZE the integer value of data[i][j] to data[i][j]
        END FOR
```

```
END FOR
PRINT "Book List: "
PRINT ""
PRINT "[Book's Name, Author, Quantity, Price in Rs.] "
FOR row in data
    PRINT row
PRINT ""
INITIALIZE a dictionary to transaction
IMPORT date and timedelta to datetime
INITIALIZE current date to borrow_date
INITIALIZE the date 10 days after the borrow_date to return_date
INITIALIZE the key as "Name", "Borrow Date", "Return Date" and the value as
customer_name, borrow_date, return_date respectively to the dictionary
transaction
WHILE count == 0
    TRY
        INPUT num_of_books
        IF num_of_books > 0
            INITIALIZE the value 1 to count
        ELSE IF num_of_books < 0
            PRINT "Negative numbers are not valid. Please try again."
    EXCEPT
        PRINT "Invalid Input. Please enter a valid number."
END WHILE
PRINT ""
INITIALIZE False to book_available
INITIALIZE the value 0 to total_amount
INITIALIZE the value 0 to books
WHILE books < num_of_books
    INPUT book_name
    INCREMENT books
```

```
        FOR I in range(len(data))
            FOR j in range(1)
                IF book_name == data[i][j] and data[i][2] > 0
                    INITIALIZE the value of data[i][3] to price
                    INITIALIZE price as the value of the dictionary
                    transaction with the key data[i][0]
                    INITIALIZE the sum of total_amount and price to
                    total_amount
                    INITIALIZE the difference of data[i][2] and 1 to stock
                    INITIALIZE stock to data[i][2]
                    INITIALIZE True to book_available
                END IF
            END FOR
        END FOR
    END WHILE
    INITIALIZE "Total Amount" as the key and total_amount as value of the
    dictionary transaction
    IMPORT os
    INITIALIZE the value 1 to transaction_id
    END WHILE os.path.exists(Transaction_{ transaction_id }.txt)
        INCREMENT the value of transaction_id
    END WHILE
    INITIALIZE "Transaction ID" as the key and I as the value of the dictionary
    transaction
    IF book_available == True
        OPEN the file "Transaction_{i}.txt" in writing mode and store it in file
        WRITE "Borrow Transaction: " to file
        PRINT ("Book Details")
        FOR key, value in transaction.items()
            INITIALIZE key and string of value by concatenating to
            transaction_details
```

```
            WRITE transaction_details to file
            WRITE a newline to file
            PRINT(transaction_details)
            END FOR
            CLOSE file
            FOR i in range (len(data))
                FOR j in range(2, 4)
                    REPLACE the integer value of data[i][j] with its string value
                END FOR
            END FOR
            OPEN the file "Library.txt" in write mode and store it in main_file
            FOR items in data
                INITIALIZE the items in data by separating them with commas to
                data_update
                WRITE data_update to main_file
            END FOR
            CLOSE main_file
            PRINT ("Thank you for borrowing")
        ELSE IF book_available == False
            PRINT (book_name + "is not available")
        END IF

DEFINE Function return_book(customer_name)
        INITIALIZE the function file_data with the parameter "Library.txt" to library_file
        INITIALIZE the function data_list with the parameter library_file to library_data
        FOR i in range(len(library_data))
            FOR j in range(2, 4)
                INITIALIZE the integer value of data[i][j] to data[i][j]
            END FOR
        END FOR
        INTIALIZE the value of 0 to count
```

```
WHILE count == 0
    TRY
        INPUT num_of_books
        IF num_of_books > 0
            INITIALIZE the value 1 to count
        ELSE IF num_of_books < 0
            PRINT "Negative numbers are not valid. Please try again."
    EXCEPT
        PRINT "Please enter a valid number"
END WHILE
INITIALIZE the value of 0 to total_amount
INITIALIZE the value of 0 to fine
INITIALIZE a dictionary to transaction
INITIALIZE "Customer's Name" as the key and customer_name as the value of
the dictionary transaction
INITIALIZE the value 0 to books
WHILE books < num_of_books
    INITIALIZE False to transaction_found
    INCREMENT books
    WHILE transaction_found == False
        TRY
            INPUT transaction_id
            INITIALIZE the function file_data with the parameter
            "Transaction_{transaction_id}.txt" to transaction_file
            INTIALIZE the function data_list with the parameter
            transaction_file to transaction_data
            CALL the function display with the parameter as
            transaction_data
        EXCEPT
            PRINT ("Transaction ID not found")
    INPUT book_name
```

```
INITIALIZE False to book_found
INITIALIZE False to book_returned
FOR i in range(len(library_data))
    FOR j in range(1)
        IF book_name == library_data[i][j]
            INITIALIZE the key as “Book” and the value as library_data[i][j] to the dictionary transaction
            INITIALIZE library_data[i][3] to price
            INITIALIZE book_name + “ : Returned” to book_status
            FOR x in range(len(transaction_data)-1)
                IF transaction_data[x][0] == book_status
                    PRINT ("The book has been returned already")
                    INITIALIZE True to book_returned
                ELSE
                    INITIALIZE book_name + “ : ” + str(price) to book_detail
                    IF transaction_data[x][0] == book_detail
                        INITIALIZE True to book_found
                        INITIALIZE transaction_data[3][0] to return_date
                        INITIALIZE the characters from the 14th to 24th index of return_date to return_date_string
                        IMPORT datetime, date, timedelta from datetime
                        INITIALIZE the object of return_date_string in the format '%Y-%m-%d' to return_date_object
```

INITIALIZE the current date to current_date

IF current_date <= return_date_object

INITIALIZE price to amount

ELSE IF current_date > return_date_object

INITIALIZE the difference between current date and return date to date_passed

INITIALIZE the date_passed * 50 to fine

INITIALIZE the sum of price and fine to amount

END IF

INITIALIZE library_data[i][2] to stock

INITIALIZE stock to library_data[i][2]

ADD amount to total_amount

INITIALIZE “Return Date”, “Fine”, “Total Amount”, book_name as the keys and current_date, fine, total_amount, “Returned” as the values to the dictionary transaction

END IF

END IF

END FOR

END IF

```
        END FOR
    END FOR
    IF book_found == True and book_returned == False
        PRINT(“Book Details”)
        OPEN the file Transaction_{transaction_id}.txt in append mode and
        store it in file
        WRITE “Return Transaction:” to file
        FOR key, value in transaction.items()
            INITIALIZE key+" : "+str(value) to transaction_details
            WRITE transaction_details to file
            WRITE a new line to file
            PRINT transaction_details
        END FOR
        CLOSE file
        FOR i in range(len(library_data))
            FOR j in range(2,4)
                INITIALIZE the string value of library_data[i][j] to
                library_data[i][j]
            END FOR
        END FOR
        OPEN the file “Library.txt” in write mode and store it in main_file
        FOR items in library_data
            INITIALIZE items to data_update
            WRITE data_update to main_file
        END FOR
        CLOSE main_file
        PRINT(“Book Returned”)
    ELSE IF book_found == False
        PRINT ("The transaction list does not match. \n Please check the
        book's name and the Transaction ID")
    END IF
```

```
        END FOR
```

4.3 Module 3 – main.py

```
START
IMPORT * from library_menu
INITIALIZE 0 to start
DEFINE FUNCTION transaction()
        INPUT customer_name
        PRINT("")
        PRINT ("Options:")
        PRINT ("1. Borrow")
        PRINT ("2. Return")
        PRINT ("3. Display books")
        INITIALIZE False to option_valid
        WHILE option_valid == False
                TRY
                        INPUT option
                        IF option == 1 or option == 2 or option == 3
                                INITIALIZE True to option_valid
                        ELSE IF option < 0 or option > 3:
                                PRINT("Please enter a valid option")
                        END IF
                EXCEPT
                        PRINT("Please enter a valid number")
        END WHILE
        IF option == 1
                CALL the function borrow with the parameter customer_name
        ELSE IF option == 2
                CALL the function return with the parameter customer_name
        ELSE IF option == 3
                INITIALIZE the function file_data with the parameter as "Library.txt" to
                library_file
```

```
            INITIALIZE the function data_list with the parameter as library_file to data
            CALL the function display with the parameter as data
        END IF
PRINT("Welcome to the Library")
PRINT("")
WHILE start == 0
        PRINT("Press 1 to start/continue")
        PRINT("Press 2 to end")
        INPUT select
        IF select == 1
            CALL the function transaction
        ELSE IF select ==2
            PRINT("Please visit again!")
            INITIALIZE 1 to start
            BREAK
        END IF
END WHILE
```

5. Data Structure

Data structures are the different ways in which data can be stored and arranged. Data needs to be organized so that it can be accessed whenever required. However, it will not be effective to store all the data in a similar manner. Hence, there are various types of data structures which can be used according to the need of the program. (Programiz, n.d.)

Some commonly used data structures in Python are lists, sets, tuples and dictionaries. The data structures used in the program are mentioned below:

5.1 List

A list is a data structure with “ordered collection of items”. Lists can contain elements which are not necessarily of the same data type. A single list can contain Strings,

Objects and Integers. It allows users to add, remove and update elements when required without any replacement to the list itself. The elements of a list must be stored in square brackets with commas to separate each element. (CFI Education Inc., 2015).

For example:

List_1 = [“Apple”, “Banana”, “Cherry”]

List used in the program:

```
def data_list(file):
    """
    Creates a list data to store the data from file
    Returns the data from the file in 2D list
    """
    data = []
    for each in file:
        data.append(each.replace("\n", "").split(","))
    return data
```

Figure 2 List used in the program

The module read_data.py, consists of a function data_list() which returns the data from file passed in the parameter as a 2D list. The variable data stores a list where the information in each file is stored as the elements. Here, data.append() appends the list and adds the elements to the list.

5.2 Dictionary

Unlike other data structures, dictionaries do not store individual elements. Groups of data can be stored in a dictionary as a pair of key: value. Keys are unique and mapped to their values. The keys must be either a string or an integer data type whereas the values may have any data type. The elements of a dictionary are written inside curly braces { } in pairs (Key:value) (Geeks For Geeks, 2021).

For example:

Dict = {Name : “Aashna”, Marks: 80}

Elements can also be added to a dictionary individually. For example:

Dict[“Name”] = “Aashna”

Dict[“Marks”] = 80

Dictionary used in the program:

```
transaction["Name"] = customer_name
transaction["Borrow Date"] = borrow_date
transaction["Return Date"] = return_date
```

Figure 3 Dictionary used in the program

The module library_menu.py consists of a function which allows the user to borrow books from the library. In the code above, elements are being added to the dictionary transaction with “Name”, “Borrow Date”, “Return Date” as keys and customer_name, borrow_date and return_date as values.

5.3 Strings

Although strings are commonly known as data types, they can be considered as data structures as well. Strings store each of their character in an array. The characters in a string may be a combination of alphabets, numbers or special characters (Geeks For Geeks, 2021). In Python, strings are written in single or double quotes (‘ ’ or “ ”).

For example:

Course = “Fundamentals of Computing”

String used in the program:

```
print("Options:")
print("1. Borrow")
print("2. Return")
print("3. Display books")
```

Figure 4 Printing the strings

```
for i in range(len(library_data)):
    for j in range(2, 4):
        library_data[i][j] = str(library_data[i][j])
```

Figure 5 Converting integer to string

```
return_date = transaction_data[3][0]
return_date_string = return_date[14:24]
```

Figure 6 Extracting certain characters from a string

The codes are written in the main.py module. In figure 4, a sequence of strings has been printed to provide options to borrow or return book/s. In figure 5, the 2nd and 3rd elements of a 2D list library_data are being converted into string. In figure 6, the characters from the 14th to 24th index of return_date is being stored in return_date_string.

Some data structures which are not used in the program are:

5.4 Tuples

Tuples are built in data structures of Python. They store collection of objects. They have some restrictions as compared to lists. Once a tuple is created, data cannot be added to it or removed from it. They cannot be appended either. The elements of a tuple are stored using parentheses; however the use of parentheses can be optional. (CFI Education Inc., 2015)

For example:

Tuple_1 = (Dog, Cat, Horse)

5.5 Sets

A set is a collection of unique items. They do not have any particular order. Sets are needed to check if an element is the member of the group (set) or not. It is even used to check how two or more sets are related. Sets can be replaced, appended, added to or even removed. The elements of a set are stored in curly braces. (CFI Education Inc., 2015)

For example:

Set_1 = {“John”, “Bob”, “Anne”}

6. Program

The coursework includes a program developed for a library to record the details of the transaction when a customer borrows or returns a book. The program has been written in IDLE (Integrated Development Learning Environment) version 3.9.6 using the programming language Python. Python is a dynamically typed programming language with high level data structures. It is a suitable language to build applications. (Python Software Foundation, 2021)

The program consists of three modules i.e., main.py, read_data.py and library_menu.py. Each of these modules consists of functions which are individually responsible for certain actions such as reading data from a file, executing a course of events when the user borrows or returns any books and interact with the user for their requirements from the library. The data of the library are stored in the file "Library.txt".

6.1 Read_data.py module

The module read_data.py consists of two functions. The first function file_data(file_name) takes the name of the file as its parameter. The file is then read and the data from that file is returned as a collection data type. The second function data_list(file) takes a list as its parameter. It replaces all the "\n" with an empty string and splits all the string after a comma as an element of a list. It then returns the list. (Refer to Appendix A for the program code.)

6.2 Library_menu.py module

The module library_menu.py consists of three functions:

- **display(data)**

  The function display(data) accepts a list as the parameter. It iterates through the file and prints the 1st elements from each row. In figure 7, the user is asked to input the option. If they choose to display books, the display(data) function will be called. The elements of the "Library.txt" file will be passed in the parameter and the name of the books will be displayed.

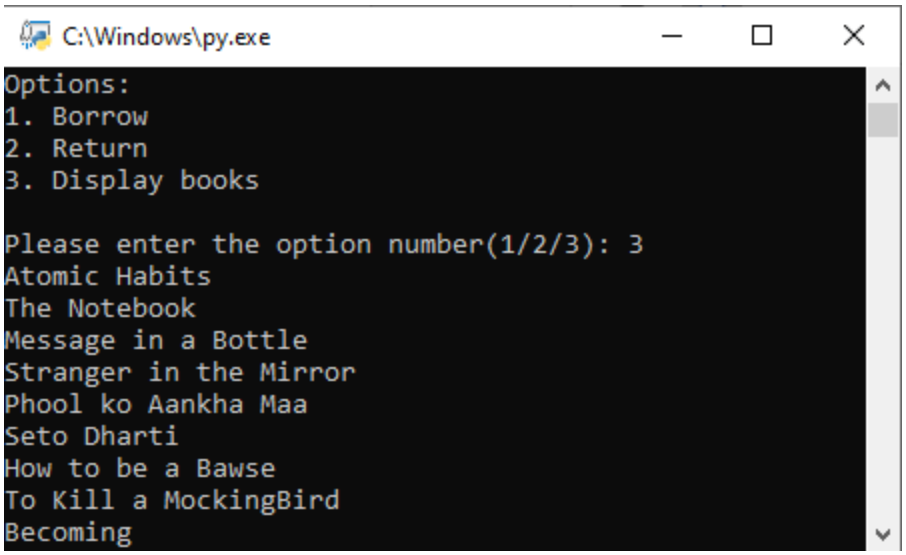


Figure 7 Displaying the books in Library.txt

- **borrow(customer_name)**

  The function borrow(customer_name) accepts the name of the customer as its parameter. When this function is called, it calls the method file_data("Library.txt") from read_data.py module. The data from "Library.txt" is read and displayed as a 2D list. The user is asked to input the number of books they want to borrow and the name of the books. The availability of the book is checked in the text file. If the book is available the total amount will be calculated and the stock will be decreased. Finally, the transaction details will be printed out stored in a new text file. (Refer to Appendix B for the program code.)

```
C:\Windows\py.exe
Book List:

[Book's Name, Author, Quantity, Price in Rs.]
['Atomic Habits', ' James Clear', 50, 1280]
['The Notebook', ' Nicholas Sparks', 30, 640]
['Message in a Bottle', ' Nicholas Sparks', 40, 560]
['Stranger in the Mirror', ' Sidney Sheldon', 25, 560]
['Phool ko Aankha Maa', ' Ani Choying Dolma', 35, 300, '']
['Seto Dharti', ' Amar Neupane', 40, 400]
['How to be a Bawse', ' Lilly Singh', 45, 1120]
['To Kill a MockingBird', ' Harper Lee', 50, 640]
['Becoming', ' Michelle Obama', 60, 1120]

Number of books to be borrowed: 2

Name of the book: Message in a Bottle
Name of the book: How to be a Bawse

Book Details:
Name : John Smith
Borrow Date : 2021-09-08
Return Date : 2021-09-18
Message in a Bottle : 560
How to be a Bawse : 1120
Total Amount : 1680
Transaction ID : 3

Thank you for borrowing
```

Figure 8 Output Result when borrow() function is called

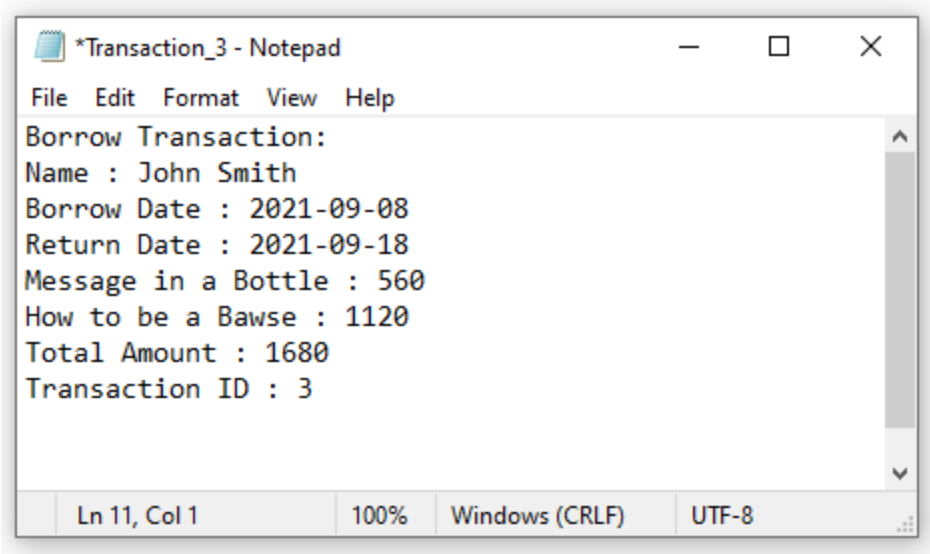


Figure 9 Text file generation when a user borrows books

- **return_book(customer_name)**

  Similar to the previous function, this functions also accepts the name of the customer as its parameter. When this function is called, it asks the user to input the number of books that they want to return and the name of the books. It also asks for the transaction id and checks the file with the matching transaction id to see if the details are correct and also see if the book has been returned already. If the required details are correct, it checks if the due date has passed. A late fine of Rs 50 will be charged if the due date has passed (Figure 9). The total amount will be calculated and the stock will be increased. Once again, the transaction details will be printed out and stored in a new text file (Figure 8). The users can even return multiple books each borrowed in different days. The details of such transactions will be stored in the respective files. (Refer to Appendix B for the program code.)

```
Number of books to be returned: 1
Enter Transaction ID: 3
['Borrow Transaction: ']
['Name : John Smith']
['Borrow Date : 2021-09-08']
['Return Date : 2021-09-18']
['Message in a Bottle : 560']
['How to be a Bawse : 1120']
['Total Amount : 1680']
['Transaction ID : 3']
Name of the book: How to be a Bawse

Book Details:
Customer's Name : John Smith
Book : How to be a Bawse
Return Date : 2021-09-08
Fine : 0
Total Amount : 1120
How to be a Bawse : Returned

Book Returned
```

Figure 10 Output result when return_book() function is called

Transaction_3 - Notepad

File Edit Format View Help

```
Return Transaction:
Customer's Name : John Smith
Book : How to be a Bawse
Return Date : 2021-09-08
Fine : 0
Total Amount : 1120
How to be a Bawse : Returned
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Figure 11 Text file when a book is being returned

```
Number of books to be returned: 1
Enter Transaction ID: 2
['Borrow Transaction: ']
['Name : Aashna']
['Borrow Date : 2021-08-26']
['Return Date : 2021-09-05']
['Atomic Habits : 1280']
['Total Amount : 1280']
['Transaction ID : 2']
['']
['Return Transaction: ']
["Customer's Name : Aashna"]
['Book : Atomic Habits']
['Return Date : 2021-09-08']
['Fine : 150']
['Total Amount : 1430']
['Atomic Habits : Returned']
['']
Name of the book: Atomic Habits
The book has been returned already
```

Figure 12 Trying to return the book which has been returned already

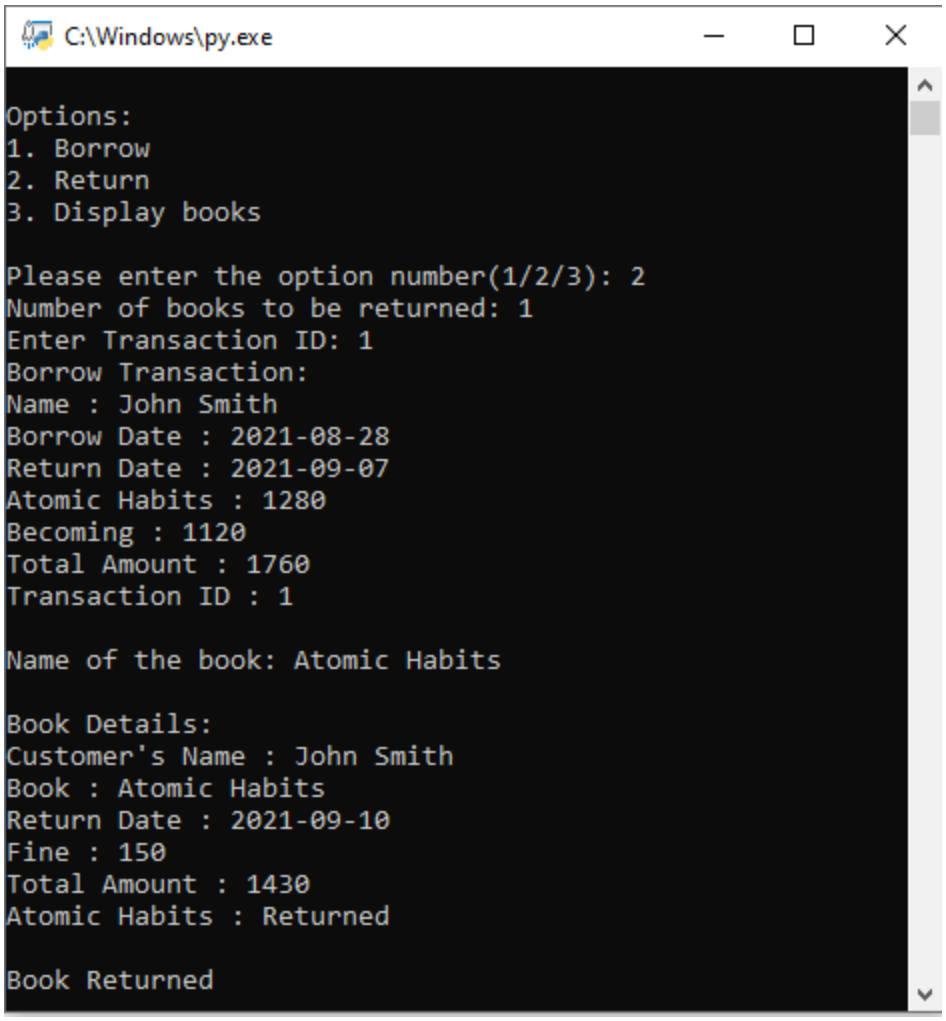


Figure 13 Fine being charged for late return

6.3 Main.py module

The module main.py consists of a function named transaction(). It starts by asking the user to input their name. It then asks the user if they want to borrow or return a book. If the user chooses to borrow, the function borrow(customer_name) will be called from the library_menu.py module. If the user chooses to return the function return_book(customer_name) will be called from the same module. Both of these functions will pass the name entered by the user as the parameter.

At the beginning of the program, the user will be asked if they want to start/continue or end the program. If they choose to continue, the function transaction() will be called and once the transaction completes, the same user will be given the options to continue or end again. If they choose to end, the program will terminate. (Refer to Appendix C for the program code.)

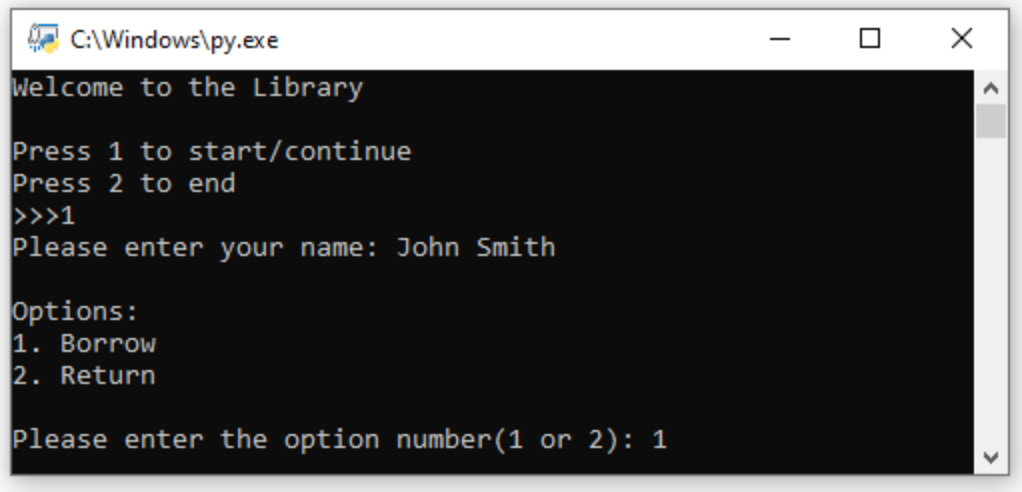


Figure 14 Output from main.py module

7. Testing

7.1 Test 1 – Implementation of Try, Except

Test No.	1
Objective	To handle an exception when user inputs the wrong transaction id.
Action	• Ask the user to input the transaction ID • Search the directory for the file with the name “Transaction_{transaction id}. txt” • If the file is not found print an error message and ask the user to input the ID again.
Expected Result	The message "Transaction ID not found” should be printed and the user should be asked to enter the ID again.

Output	The message "Transaction ID not found" was printed and the user was asked to enter the ID again.
Conclusion	The test has been completed successfully.

Table 1Testing the implementation of Try Except

Output Result:

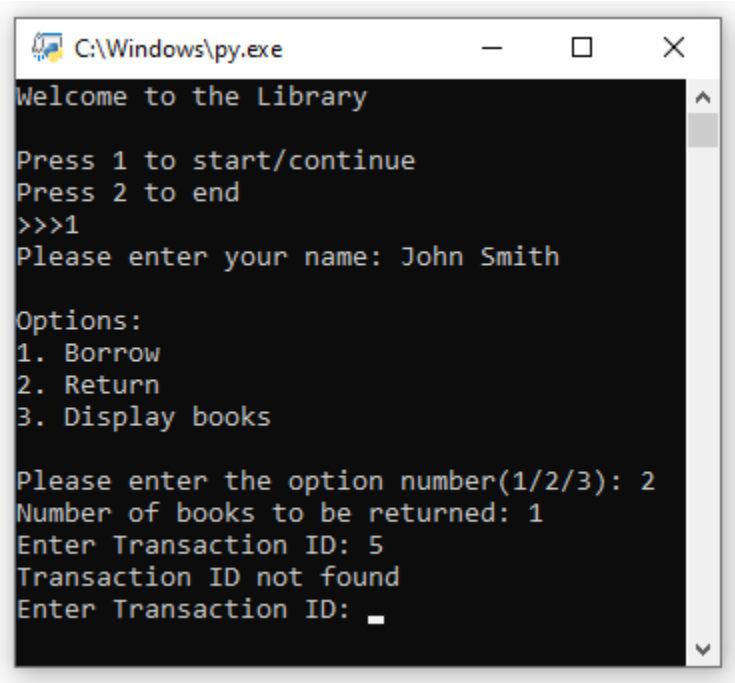


Figure 15 Output of Test 1

7.2 Test 2 – Selection of borrow and return option

Test No.	2
Objective	To input an error message when user inputs any other value or a noon-integer when asked to choose the borrow or return options.
Action	• Ask the user to input option number 1 or 2 or 3 to choose if they want to borrow, return or display the books. • If the user inputs a non-integer value, print an error message and ask them to try again.
Expected Result	The message "Invalid Input. Choose 1 to borrow and 2 to return." should be printed and the user should be asked to input the option number again.
Output	The message "Invalid Input. Choose 1 to borrow and 2 to return." was printed and the user was asked to input the option number again.
Conclusion	The test has been completed successfully.

Table 2 Testing the borrow and return process

Output Result:

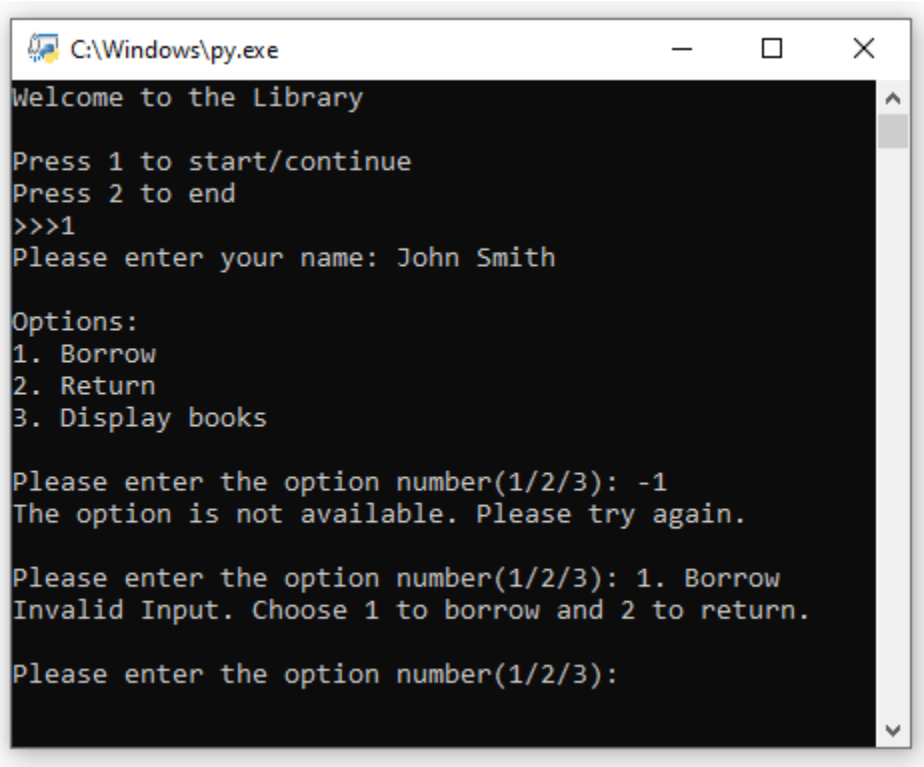


Figure 16 Output of Test 2

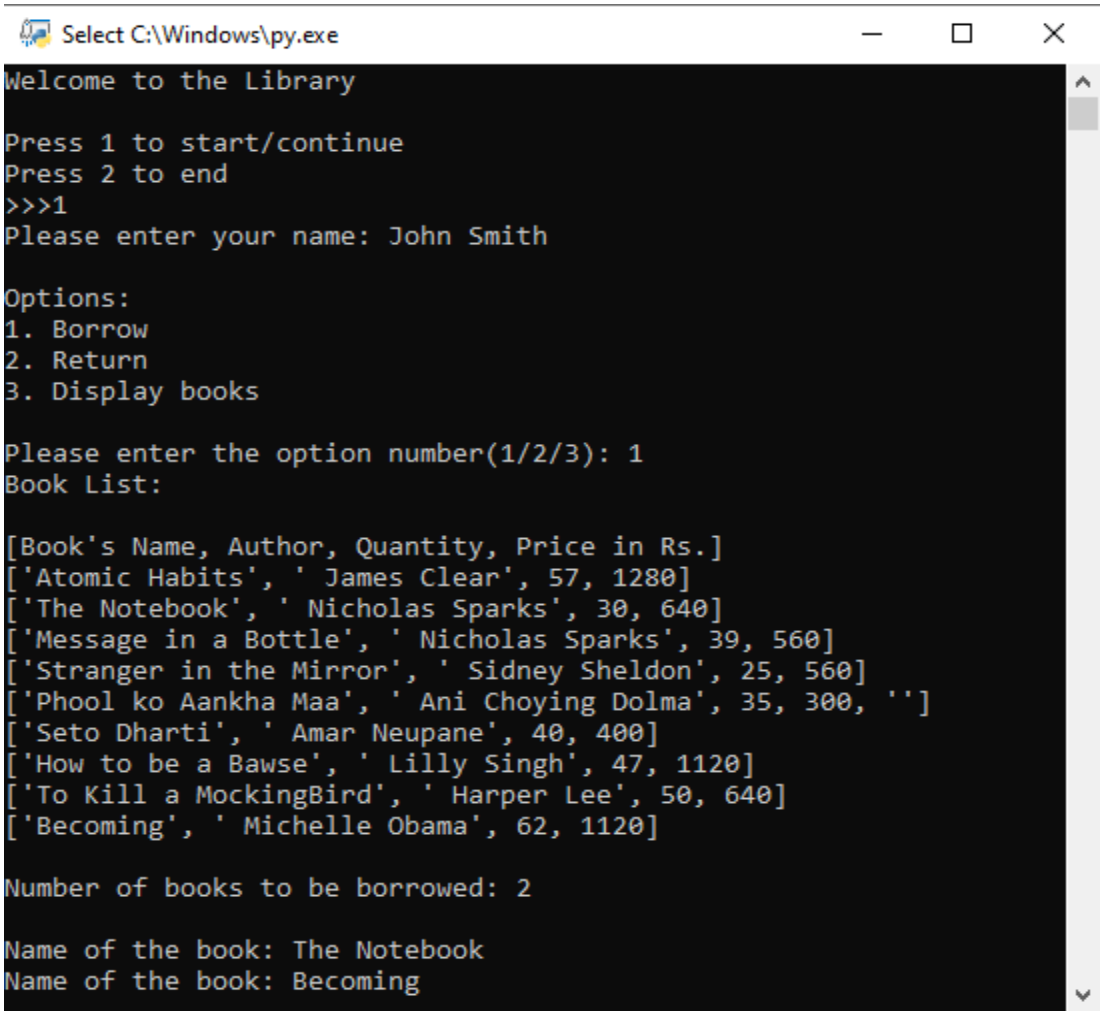
7.3 Test 3 – File generation of borrow

Test No.	3
Objective	To generate a file when user borrows a book.
Action	• Ask the user to input if they want to borrow or return any books. • If they choose to borrow show the complete book list and ask the user for the number of books that they want to borrow. • Ask the user to input the name of the books they want to borrow. • Calculate the total amount. • Write the transaction details in a text file.
Expected Result	The transaction details which include the

	customer's name, borrow date, return date, name of the books and the total amount to be paid should be displayed and written into a new text file.
Output Result	All the details were displayed and written into a new text file.
Conclusion	The test has been completed successfully.

Table 3 Testing the file generation of borrow

Output Result:



```
Select C:\Windows\py.exe
Welcome to the Library

Press 1 to start/continue
Press 2 to end
>>>1
Please enter your name: John Smith

Options:
1. Borrow
2. Return
3. Display books

Please enter the option number(1/2/3): 1
Book List:

[Book's Name, Author, Quantity, Price in Rs.]
['Atomic Habits', ' James Clear', 57, 1280]
['The Notebook', ' Nicholas Sparks', 30, 640]
['Message in a Bottle', ' Nicholas Sparks', 39, 560]
['Stranger in the Mirror', ' Sidney Sheldon', 25, 560]
['Phool ko Aankha Maa', ' Ani Choying Dolma', 35, 300, '']
['Seto Dharti', ' Amar Neupane', 40, 400]
['How to be a Bawse', ' Lilly Singh', 47, 1120]
['To Kill a MockingBird', ' Harper Lee', 50, 640]
['Becoming', ' Michelle Obama', 62, 1120]

Number of books to be borrowed: 2

Name of the book: The Notebook
Name of the book: Becoming
```

Figure 17 Borrow Process

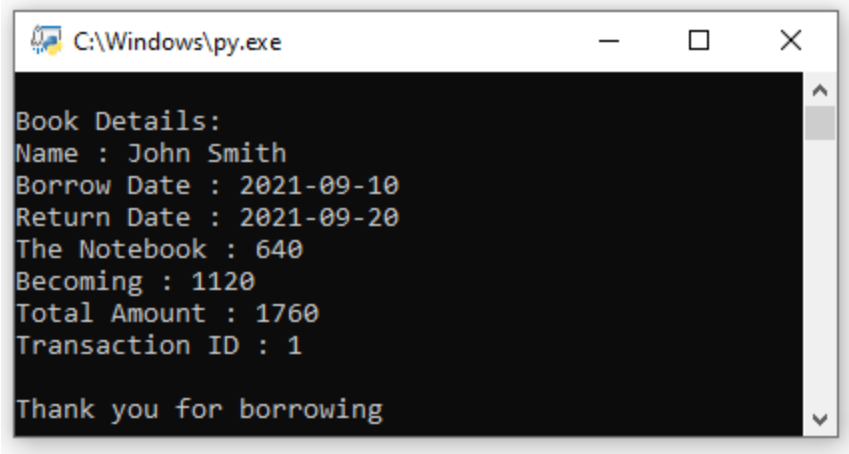


Figure 18 Output from borrow in shell

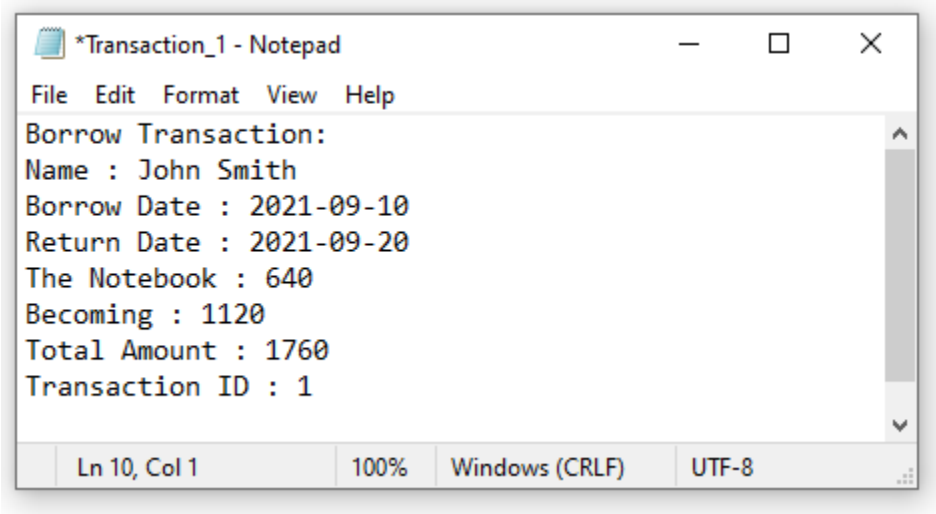


Figure 19 Transaction details in file when book has been borrowed

7.4 Test 4 – File generation from return

Test No.	4
Objective	To generate a file when user borrows a book.
Action	• Ask the user to input if they want to borrow or return any books. • If they choose return, ask for the transaction ID. • Search the directory with the file name Transaction_transaction_id.txt • If the directory is found, check if the return date has passed. • If the return date has not passed get the total amount that they need to pay, else add Rs 50 fine per day. • Display the transaction details and write them in the same file in which the book was borrowed.
Expectation	The transaction details which include the customer's name, return date, name of the books and the total amount to be paid including the fine (if needed) should be displayed and written into a new text file.
Output Result	All the details were displayed and written into a new text file.
Conclusion	The test has been completed successfully.

Table 4 Testing the file generation of return

Output Result:

```
Select C:\Windows\py.exe
Welcome to the Library

Press 1 to start/continue
Press 2 to end
>>>1
Please enter your name: John Smith

Options:
1. Borrow
2. Return
3. Display books

Please enter the option number(1/2/3): 2
Number of books to be returned: 1
Enter Transaction ID: 1
Borrow Transaction:
Name : John Smith
Borrow Date : 2021-09-10
Return Date : 2021-09-20
The Notebook : 640
Becoming : 1120
Total Amount : 1760
Transaction ID : 1
Name of the book: Becoming

Book Details:
Customer's Name : John Smith
Book : Becoming
Return Date : 2021-09-10
Fine : 0
Total Amount : 1120
Becoming : Returned

Book Returned
```

Figure 20 Return Process

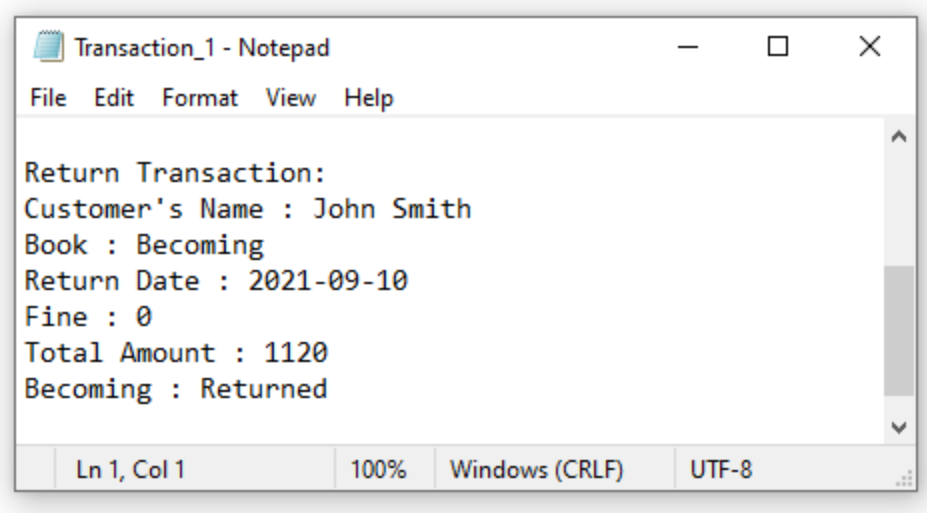


Figure 21 Transacation details in a file when a book has been returned

7.5 Test 5 – Update the stock after a transaction

Test No.	5
Objective	To decrease the stock of the book when a customer borrows a book and increase it when they return it.
Actions	• Once a customer borrows a book successfully, decrease the stock of the book and update it to the file "Library.txt". • Once a customer returns a book successfully, increase the stock of the book and update it to the file "Library.txt".
Expected Result	The stock of a book that was borrowed should decrease and that of the book that was returned should increase.
Output Result	The stock decreased when the book was borrowed and increased when it was returned.
Conclusion	The test has been completed successfully.

Table 5 Testing the stock update after a transaction

Output Result

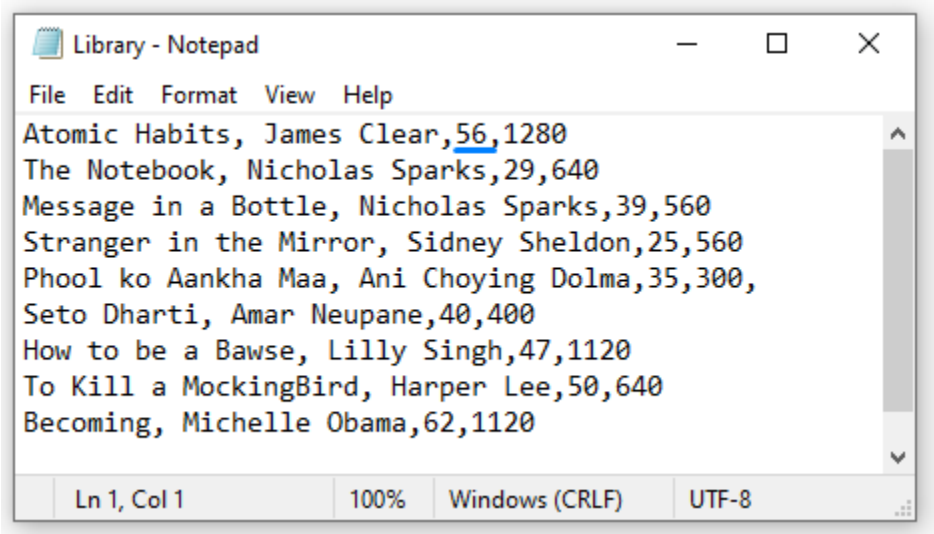


Figure 22 Stock before book was borrowed

```
C:\Windows\py.exe
1. Borrow
2. Return
3. Display books

Please enter the option number(1/2/3): 1
Book List:

[Book's Name, Author, Quantity, Price in Rs.]
['Atomic Habits', ' James Clear', 56, 1280]
['The Notebook', ' Nicholas Sparks', 29, 640]
['Message in a Bottle', ' Nicholas Sparks', 39, 560]
['Stranger in the Mirror', ' Sidney Sheldon', 25, 560]
['Phool ko Aankha Maa', ' Ani Choying Dolma', 35, 300, '']
['Seto Dharti', ' Amar Neupane', 40, 400]
['How to be a Bawse', ' Lilly Singh', 47, 1120]
['To Kill a MockingBird', ' Harper Lee', 50, 640]
['Becoming', ' Michelle Obama', 62, 1120]

Number of books to be borrowed: 1

Name of the book: Atomic Habits

Book Details:
Name : Bob Parker
Borrow Date : 2021-09-10
Return Date : 2021-09-20
Atomic Habits : 1280
Total Amount : 1280
Transaction ID : 2

Thank you for borrowing
```

Figure 23 Borrow Process

```
Library - Notepad
File  Edit  Format  View  Help
Atomic Habits, James Clear,55,1280
The Notebook, Nicholas Sparks,29,640
Message in a Bottle, Nicholas Sparks,39,560
Stranger in the Mirror, Sidney Sheldon,25,560
Phool ko Aankha Maa, Ani Choying Dolma,35,300,
Seto Dharti, Amar Neupane,40,400
How to be a Bawse, Lilly Singh,47,1120
To Kill a MockingBird, Harper Lee,50,640
Becoming, Michelle Obama,62,1120
Ln 1, Col 1   100%   Windows (CRLF)   UTF-8
```

Figure 24 Stock after book was borrowed

```
C:\Windows\py.exe

Options:
1. Borrow
2. Return
3. Display books

Please enter the option number(1/2/3): 2
Number of books to be returned: 1
Enter Transaction ID: 2
Borrow Transaction:
Name : Bob Parker
Borrow Date : 2021-09-10
Return Date : 2021-09-20
Atomic Habits : 1280
Total Amount : 1280
Transaction ID : 2
Name of the book: Atomic Habits

Book Details:
Customer's Name : Bob Parker
Book : Atomic Habits
Return Date : 2021-09-10
Fine : 0
Total Amount : 1280
Atomic Habits : Returned

Book Returned
```

Figure 25 Return Process

```
Library - Notepad
File  Edit  Format  View  Help
Atomic Habits, James Clear,56,1280
The Notebook, Nicholas Sparks,29,640
Message in a Bottle, Nicholas Sparks,39,560
Stranger in the Mirror, Sidney Sheldon,25,560
Phool ko Aankha Maa, Ani Choying Dolma,35,300,
Seto Dharti, Amar Neupane,40,400
How to be a Bawse, Lilly Singh,47,1120
To Kill a MockingBird, Harper Lee,50,640
Becoming, Michelle Obama,62,1120
Ln 1, Col 1   100%   Windows (CRLF)   UTF-8
```

Figure 26 Stock after book was returned

8. Conclusion

This report is based on a coursework which assigned to design a library management system through an application on Python programming language. The program has been built with a modular approach where each module consists of functions with different purpose such as reading data from a file and carrying out a number of consecutive steps to borrow a return a book. Tests have been carried out to check if all the procedures work properly. While carrying out the tests, some bugs were discovered in the program which led to exceptions. For instance, in the initial code, exception occurred when a user entered a string value where integer should have been the input and the program would come to a halt due to this. The exception has been handled by displaying an error message and asking the user to provide the input again. A number of such obstacles have been tackled and it has been concluded that the functions run smoothly and the user interface is handled well.

Besides the program, an algorithm with the step-by-step process of how the system works has been written with a diagrammatic representation in a flowchart. Although these steps make the system's concept clear, working with the steps of the algorithm was a bit difficult due to all the repeated tasks (functions that work in loops, such as asking the user to input as many books as they want to borrow), exceptions, and conditions of the system. However, with some research and breaking down the procedures, better understanding of algorithm has been established.

Despite having to face some difficulties, the coursework has been completed successfully. Since there were no restrictions on the working mechanism of the management system, designing it required some research and creativity. It has helped to understand how the management system of a library works, problems that could arise in the system and the different solutions to those problems. It has also helped to establish a better knowledge of the concepts of programming with Python, data structures, exception handling, algorithm, flowchart and file-based system.

9. Appendix

9.1 Appendix A – read_data.py

```
def file_data(file_name):
    """Reads data from the file passed in the parameter.
    Returns a collection data type with each element from the file.
    """
    file = open(file_name, "r")
    data = file.readlines()
    file.close()
    return data


def data_list(file):
    """Creates a list data to store the data from file
    Returns the data from the file in 2D list
    """
    data = []
    for each in file:
        data.append(each.replace("\n", "").split(","))
    return data
```

9.2 Appendix B – library_menu.py

```
from read_data import *


def borrow(customer_name):
    """Actions to be performed when a book is to be borrowed
    The function takes the parameter customer_name.
    Takes input from user for the book to be borrowed.
    Generates a txt file with transaction details.
    Updates the stock of the books once they are borrowed.
    """
```

```
#Calling the function file_data() and data_list() from read_data.py
fileData = file_data("Library.txt")
data = data_list(fileData)

for i in range(len(data)):
    for j in range(2, 4):
        data[i][j]  = int(data[i][j])

print("Book List: ")
print("")
print("[Book's Name, Author, Quantity, Price in Rs.]")
for row in data:
    print(row)     #Printing the booklist
print("")

transaction = {}   #Dictionary to store transaction details

#Import date and time
from datetime import date, timedelta
borrow_date = date.today()
return_date = borrow_date + timedelta(days=10)

#Data for dictionary
transaction["Name"] = customer_name
transaction["Borrow Date"] = borrow_date
transaction["Return Date"] = return_date

#Getting the number of books to be borrowed input by the user
count = 0
while count == 0:
```

```
        try:
            num_of_books = int(input("Number of books to be borrowed: "))
            if num_of_books > 0:
                count = 1
            elif num_of_books <= 0:
                print("Negative numbers are not valid. Please try again.")
        except:
            print("Invalid Input. Please enter a valid number.")
    print("")

    book_available = False
    total_amount = 0
    books = 0
    while books < num_of_books:
        book_name = input("Name of the book: ")
        books+=1
        for i in range(len(data)):
            for j in range(1):
                '''Checks if the book is available.
                Calculates the total amount and updates the stock.
                '''
                if book_name == data[i][j] and data[i][2] > 0:
                    price = data[i][3]
                    transaction[data[i][0]] = price
                    total_amount += price
                    stock = data[i][2] - 1
                    data[i][2] = stock
                    book_available = True


    transaction["Total Amount"] = total_amount
```

```
'''Checks if the file with the transaction id exists.
Increments the transaction id if the id exists already.
'''
import os
transaction_id = 1
while os.path.exists(f"Transaction_{transaction_id}.txt"):
    transaction_id += 1
transaction["Transaction ID"] = transaction_id


if book_available == True:
    '''Opens a txt file with the name as Transaction_transaction_id in write mode.
        Writes the details of the transaction in the file.'''
    file = open(f"Transaction_{transaction_id}.txt", "w")
    file.write("Borrow Transaction: \n")
    print("\nBook Details:")
    for key, value in transaction.items():
        transaction_details = key+" : "+str(value)
        file.write(transaction_details)
        file.write("\n")
        print(transaction_details)
    file.close()

    for i in range(len(data)):
        for j in range(2, 4):
            data[i][j]  = str(data[i][j])

    #Updates the data with new stock in the file "Library.txt"
    main_file = open("Library.txt", "w")
    for items in data:
```

```
            data_update = ",".join(items)
            main_file.write(data_update+"\n")
        main_file.close()

        print("\nThank you for borrowing")
    elif book_available == False:
        print(book_name, "is not available")  #Error message in case the book entered by the user is not available.


def return_book(customer_name):
    """Actions to be performed when a book is to be returned.
    The function takes the parameter customer_name.
    Takes input from user for the book to be returned.
    Finds the txt file with the details when the book was borrowed and writes the details of return in it
    Updates the stock of the books once they are returned.
    """

    #Calling the function file_data() and data_list() from read_data.py
    library_file = file_data("Library.txt")
    library_data = data_list(library_file)

    #Converts the price and range into integer
    for i in range(len(library_data)):
        for j in range(2, 4):
            library_data[i][j] = int(library_data[i][j])

    #Getting the number of books to be returned input by the user
    count = 0
    while count == 0:
```

```
        try:
            num_of_books = int(input("Number of books to be returned: "))
            if num_of_books > 0:
                count = 1
            elif num_of_books <= 0:
                print("Negative numbers are not valid. Please try again.")
        except:
            print("Please enter a valid number")

    total_amount = 0
    fine = 0
    transaction = {}
    transaction["Customer's Name"] = customer_name
    books = 0

    while books < num_of_books:
        transaction_found = False
        books +=1
        while transaction_found == False:
            '''Print the details of the transaction with the id input by the user.
            Display and error message if the file is not available.
            '''
            try:
                transaction_id = input("Enter Transaction ID: ")
                transaction_file = file_data(f"Transaction_{transaction_id}.txt")
                transaction_data = data_list(transaction_file)
                for row in transaction_data:
                    print(row)
                transaction_found = True
            except:
                print("Transaction ID not found")
```

```
        book_name = input("Name of the book: ")
        book_found = False
        book_returned = False

        for i in range(len(library_data)):
            for j in range(1):
                if book_name == library_data[i][j]:
                    transaction["Book"] = library_data[i][0]
                  price = library_data[i][3]    #Getting the price of the book entered by the
user

                    book_status = book_name + " : Returned"
                    for x in range(len(transaction_data) - 1):
                     #Prints an error message if the book entered by the user has been
returned already
                        if transaction_data[x][0] == book_status:
                            print("The book has been returned already")
                            book_returned = True

                        else:
                            book_detail = book_name + " : " + str(price)
                            if transaction_data[x][0] == book_detail:
                                book_found = True

                                return_date = transaction_data[3][0]
                                return_date_string = return_date[14:24]

                                from datetime import datetime, date, timedelta
                                #Converts return_date_string to yyyy/mm/dd format
```

```
                return_date_object = datetime.strptime(return_date_string, '%Y-%m-%d').date()

                current_date = date.today()  #Getting the current date

                '''Checks if the date to return the book has passed.
                Adds a fine to the total amount if the date has passed.
                '''
                if current_date <= return_date_object:
                    amount = price
                elif current_date > return_date_object:
                    date_passed = (current_date - return_date_object).days
                    fine = 50 * date_passed
                    amount = price + fine

                stock = library_data[i][2] + 1
                library_data[i][2] = stock

                total_amount += amount

                #Data for dictionary
                transaction["Return Date"] = current_date
                transaction["Fine"] = fine
                transaction["Total Amount"] = total_amount
                transaction[book_name] = "Returned"

        if book_found == True and book_returned == False:
            '''Opens a txt file with the name as Transaction_transaction_id in append mode.
            Updates the return details of the transaction in the file.
            '''
```

```
            print("\nBook Details:")
            file = open(f"Transaction_{transaction_id}.txt", "a")
            file.write("\nReturn Transaction: \n")
            for key, value in transaction.items():
                transaction_details = key+" : "+str(value)
                file.write(transaction_details)
                file.write("\n")
                print(transaction_details)
            file.write("\n")
            file.close()

            #Updates the data with new stock in the file "Library.txt"
            for i in range(len(library_data)):
                for j in range(2, 4):
                    library_data[i][j] = str(library_data[i][j])
            main_file = open("Library.txt", "w")
            for items in library_data:
                data_update = ",".join(items)
                main_file.write(data_update+"\n")
            main_file.close()
            print("\nBook Returned")

        elif book_found == False:
            #Error message to be displayed if the book's name entered by the user is not found in the file
            print("The transaction list does not match. \n Please check the book's name and the Transaction ID")
```

9.3 Appendix C – main.py

```
from library_menu import *

start = 0
def transaction():
    """Asks users to choose if they want to borrow or return any book.
    If the user choosed to borrow the function borrow() is called
    else if they choose to return the function return_book will be called.
    Both the functions pass customer_name input by the user as the parameter.
    """
    customer_name = input("Please enter your name: ")
    print("")
    print("Options:")
    print("1. Borrow")
    print("2. Return")

    option_valid = False
    while option_valid == False:
        try:
            option = int(input("\nPlease enter the option number(1 or 2): "))
            if option == 1 or option == 2:
                option_valid = True
            elif option < 1 or option > 2:
                print("The option is not available. Please try again.")
        #Prints an error message if the input is a non_integer value
        except:
            print("Invalid Input. Choose 1 to borrow and 2 to return.")
    if option == 1:
        borrow(customer_name)
    elif option == 2:
        return_book(customer_name)
```

```
print("Welcome to the Library")
print("")

#Asks the user if they want to start/continue or close the application after each transaction is completed
while start == 0:
    print("Press 1 to start/continue")
    print("Press 2 to end")
    select = int(input(">>>"))
    if select == 1:
        transaction()
    if select == 2:
        print("Please visit again!")
        start = 1
        break
```

10. Bibliography

CFI Education Inc., 2015. *Python Data Structures.* [Online]
Available at: https://corporatefinanceinstitute.com/resources/knowledge/other/python-data-structures/
[Accessed 2 September 2021].

Datacamp, 2020. *Introduction to Python IDLE.* [Online]
Available at: https://www.datacamp.com/community/tutorials/python-IDLE?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377095&utm_targetid=aud-299261629
[Accessed 28 August 2021].

Geeks For Geeks, 2021. *Python Dictionary.* [Online]
Available at: https://www.geeksforgeeks.org/python-dictionary/
[Accessed 2 September 2021].

Geeks For Geeks, 2021. *String Data Structure.* [Online]
Available at: https://www.geeksforgeeks.org/string-data-structure/#Python
[Accessed 2 September 2021].

Programiz, n.d. *Data Structure and Types.* [Online]
Available at: https://www.programiz.com/dsa/data-structure-types
[Accessed 2 September 2021].

Python Software Foundation, 2021. *Python Tutorial.* [Online]
Available at: https://docs.python.org/3/tutorial/index.html
[Accessed 27 August 2021].