LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4051NI Fundamentals of Computing

60% Individual Coursework

2023 Summer

Student Name: Shirshak Aryal

London Met ID: 22085620

College ID: NP01CP4S230122

Assignment Due Date: Friday, August 25, 2023

Assignment Submission Date: Thursday, August 24, 2023

Word Count: 12669

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher 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

Table of Figures

Table of Tables

1. Introduction

This project is the coursework of the module CS4051NI Fundamentals of Computing, and accounts for 60% of the final grade of this module. It involves the use of Python to develop a program for an equipment rental shop. The program is supposed to carry out all the necessary processes that occur when an item/multiple items are rented and/or returned, which include updating the existing stock, generating invoices, etc.

Python is a high level, interpreted, object-oriented programming language with several high-level built-in features to enhance its versatility and areas of use. It also supports procedural programming. It has a dynamic and relatively simple syntax, which is one of the many reasons it is one of the most widely used programming languages in the world. (Anon., 2023)

1.1. Goals and objectives

The goal of this project is to develop an equipment rental system using Python, achieving the following objectives:

a. Dividing the program into 4 different modules
b. Defining separate functions/methods for purposes like:
    - reading and writing to a text file containing the product stock information,
    - renting and returning of products
    - generating invoices for either process
    - writing invoices to the respective new text files
    - generating unique file names for each text file storing the invoices
c. Using different types of collection data structures like lists and dictionaries to store the product information

This program was developed using an object-oriented approach to reduce the overall volume of code and increase its efficiency as compared to procedural programming. It also helps achieve a higher degree of code modularity, reusability, and ease of maintenance.

1.2. Tools Used

1.2.1. IDLE

IDLE is an Integrated Development and Learning Environment developed by Python. The software is coded 100% using Python, and has features like shell window, editor window, colorized text for keywords, error messages, input, etc., and a built-in debugging tool. (Anon., 2023)

IDLE was used as the software to develop and interact with the program.

1.2.2. draw.io

draw.io is a technology framework designed to create diagramming applications. It holds itself as the most-used browser-based diagramming software among end users worldwide. (Anon., 2023)

draw.io was used in this project to create a flowchart that represents the working of the whole program.

1.2.3. MS Word

MS-Word is a word-processing software developed by Microsoft Corporation (Anon., 2023). It is the most widely used word-processing program in the world.

MS-Word was used to formulate this report for the project.

2. Discussion and Analysis

The development of this program is illustrated through a few different ways below:

2.1. Algorithm

An algorithm is a set of rules or steps that is to be followed while solving problems. Algorithms are described in informal, natural language to be understood by anyone regardless of their field of expertise. Some examples of algorithms are recipes, finding books in the library, etc.

The algorithm for this program is given below:

algorithm Equipment Rental System

Step 1: **Output** shop information.

Step 2: **Output** welcome message.

Step 3: **Output** the 3 options, i.e., rent, return, or exit.

Step 4. **Input option**.

Step 5. **If option** == 1, go to Step 6. **Else if option** == 2, go to Step 24. **Else,** go to Step 44.

Step 6. **Read** from stock text file and display available stocks.

Step 7. **Input prod_id.**

Step 8. **If prod_id** is invalid, **output** error message, then go to Step 7. **Else**, go to Step 9.

Step 9. **Input qty**.

Step 10. **If qty** is invalid, **output** error message, then go to Step 9. **Else**, go to Step 11.

Step 11. **Input rented_for**.

Step 12. **If rented_for** is invalid, **output** error message, then go to Step 11. **Else**, go to Step 13.

Step 13. **Update** the stock text file.

Step 14. **Input rent_choice**.

Step 15. **If rent_choice** == ‘y’, go to Step 6. **Else if rent_choice** == ‘n’, go to Step 16. **Else, output** error message, then go to Step 14.

Step 16. **Output** invoice message.

Step 17. **Input cust_name**.

Step 18. **If cust_name** is invalid, **output** error message, then go to Step 17. **Else**, go to Step 19.

Step 19. **Input cust_num.**

Step 20. **If cust_num** is invalid, **output** error message, then go to Step 19. **Else**, go to Step 21.

Step 21. **Generate** rent invoice.

Step 22. **Write** invoice to text file.

Step 23. Go to Step 3.

Step 24. **Read** from stock text file and display available stocks.

Step 25. **Input prod_id.**

Step 26. **If prod_id** is invalid, **output** error message, then go to Step 25. **Else**, go to Step 27.

Step 27. **Input qty**.

Step 28. **If qty** is invalid, **output** error message, then go to Step 27. Else, go to Step 29.

Step 29. **Input days_rented**.

Step 30. **If days_rented** is invalid, **output** error message, then go to Step 29. Else, go to Step 31.

Step 31. **Input returned_after.**

Step 32. **If returned_after** is invalid, **output** error message, then go to Step 31. **Else,** go to Step 33.

Step 33. **Update** the stock text file.

Step 34. **Input return_choice**.

Step 35**. If return_choice** == ‘y’, go to Step 24. **Else if return_choice** == ‘n’, go to Step 35. **Else,** output error message, then go to Step 34.

Step 36. **Output** invoice message.

Step 37. **Input cust_name**.

Step 38. **If cust_name** is invalid, **output** error message, then go to Step 37. **Else**, go to Step 39.

Step 39. **Input cust_num**.

Step 40. **If cust_num** is invalid, **output** error message, then go to Step 39. **Else**, go to Step 41.

Step 41. **Generate** return invoice.

Step 42. **Write** invoice to text file.

Step 43. Go to Step 3.

Step 44. **Output** exit message.

End Equipment Rental System

2.2. Flowchart

A flowchart is an informal, diagrammatic representation of the separate steps of solving a problem in a sequential manner. It is a generic way of describing the flow of the solution and can be applied to almost any problem. For example, the recipe of a dish can again be represented in a flowchart, step by step, similar to an algorithm but in a pictorial form. It explains the steps in a concise manner.

The flowchart of this program is illustrated below:

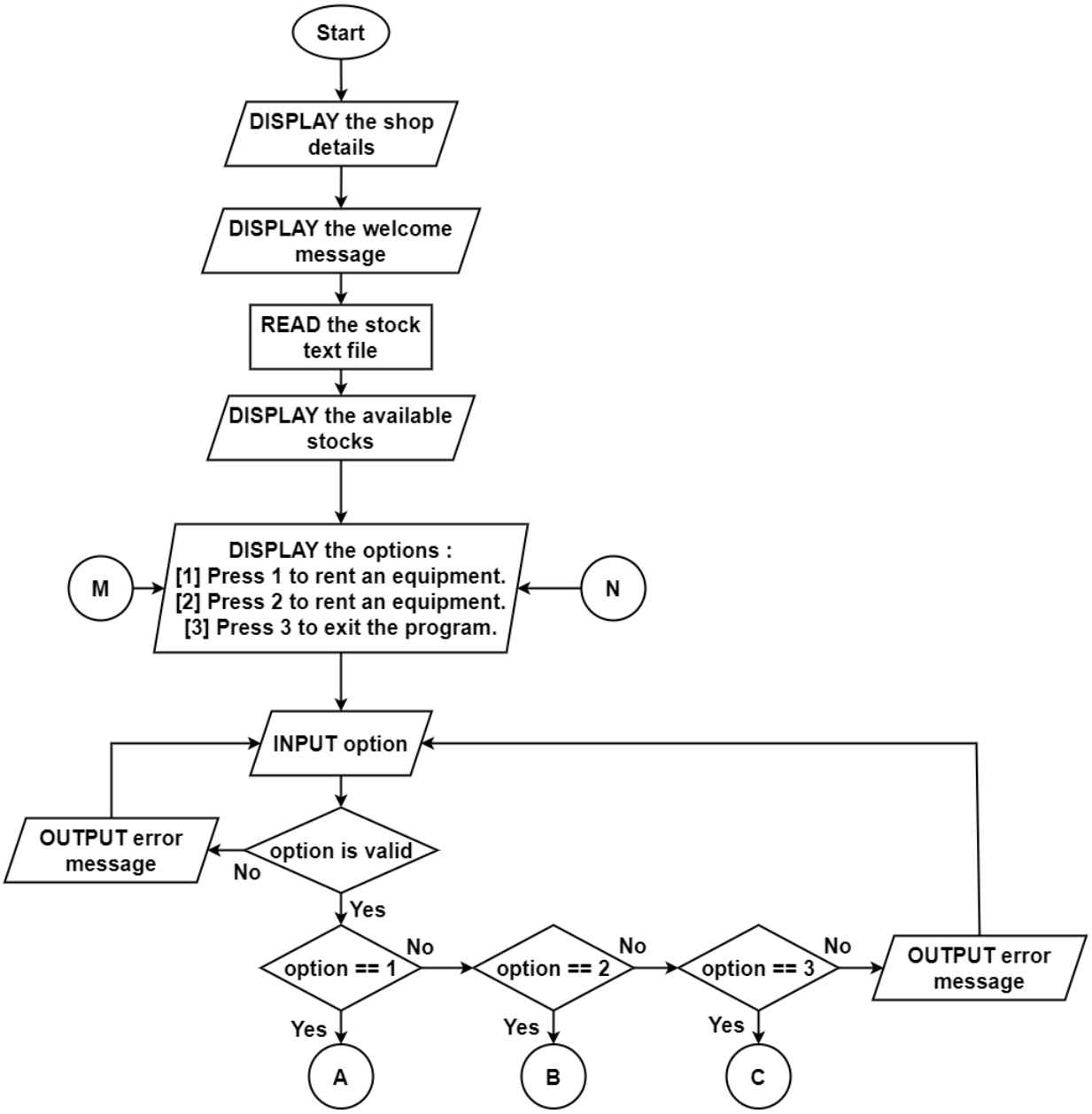


Figure 1: Flowchart of Main Menu

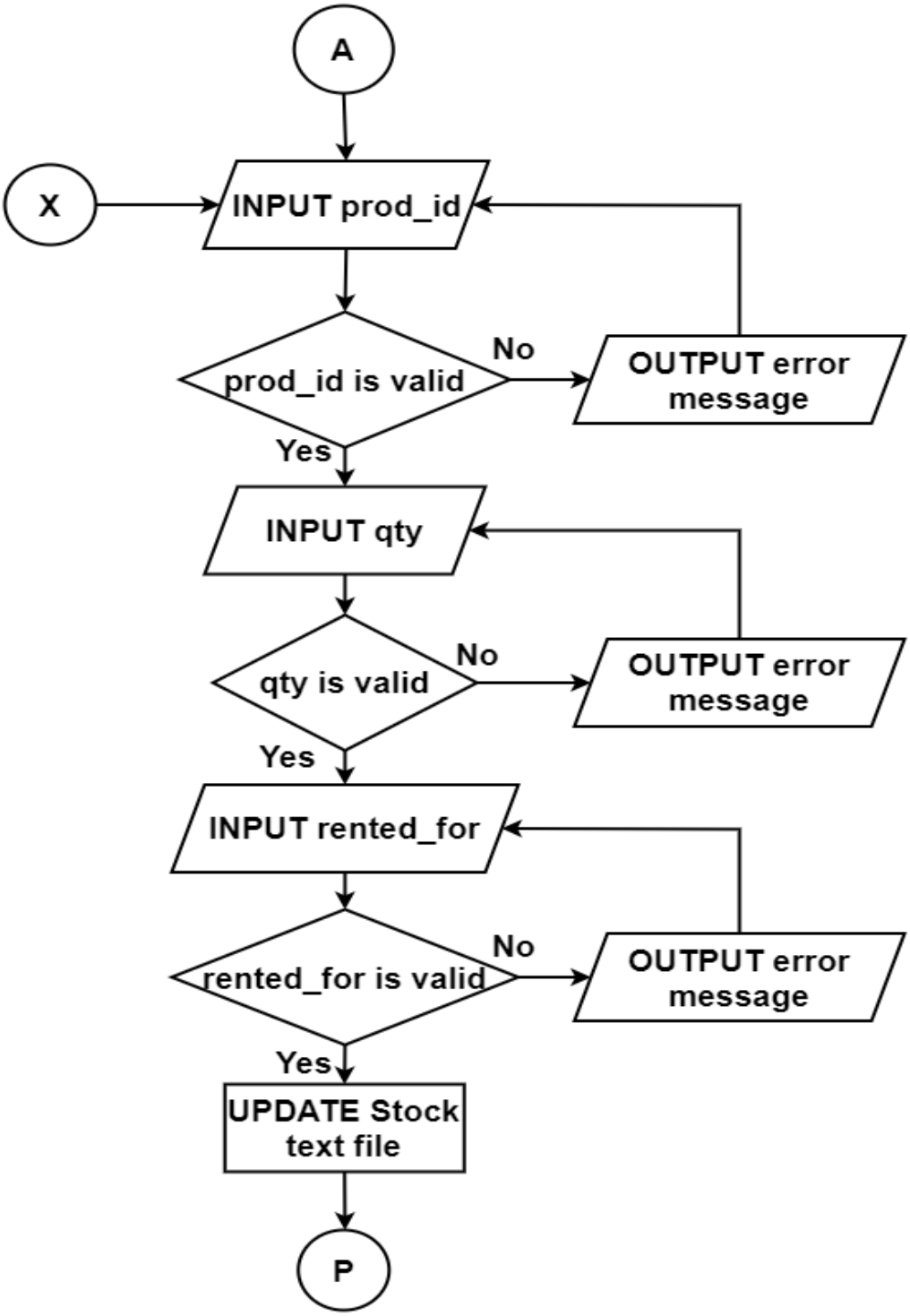


Figure 2: Flowchart of Rent process (part 1)

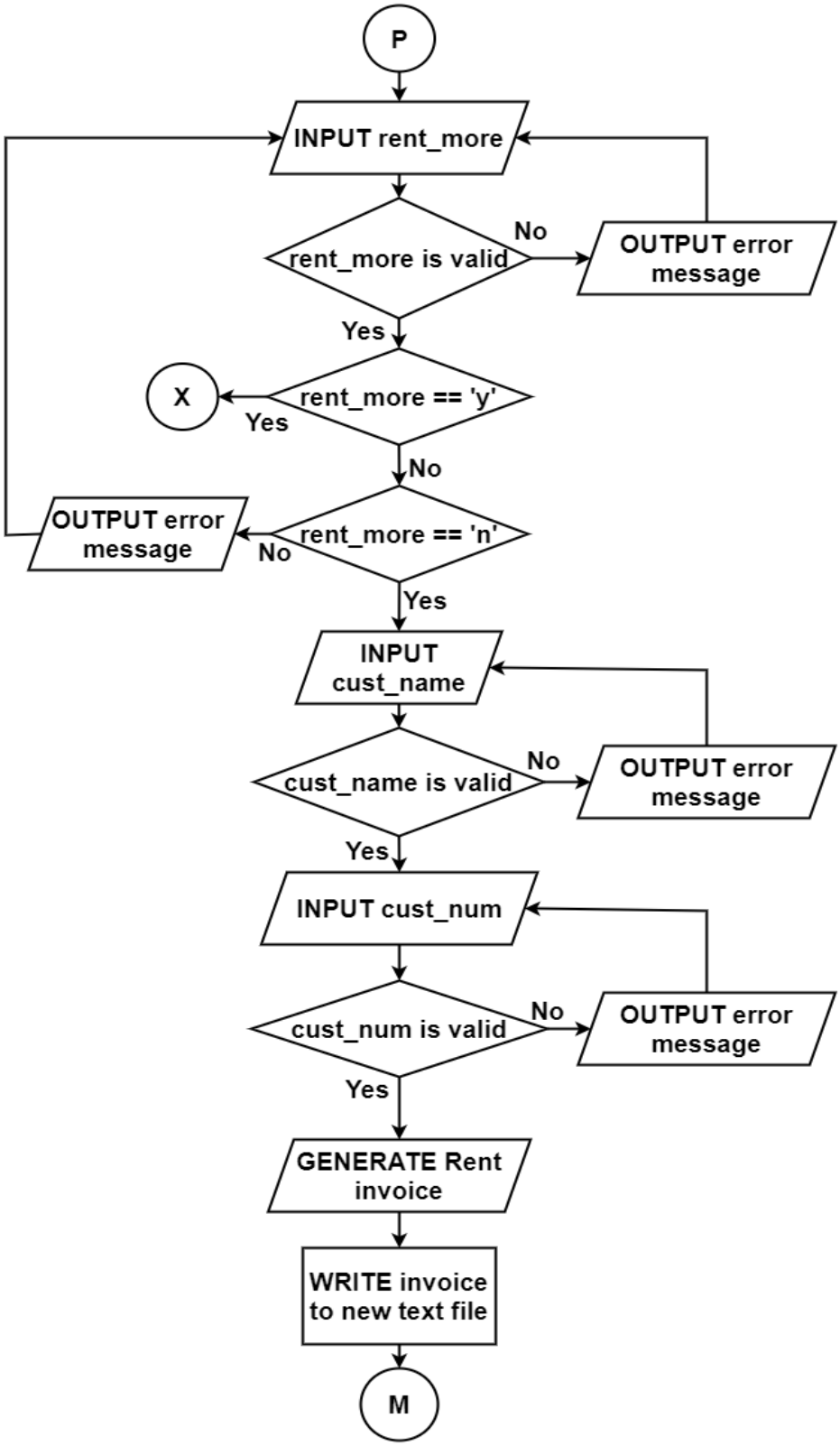


Figure 3: Flowchart of Rent process (part 2)

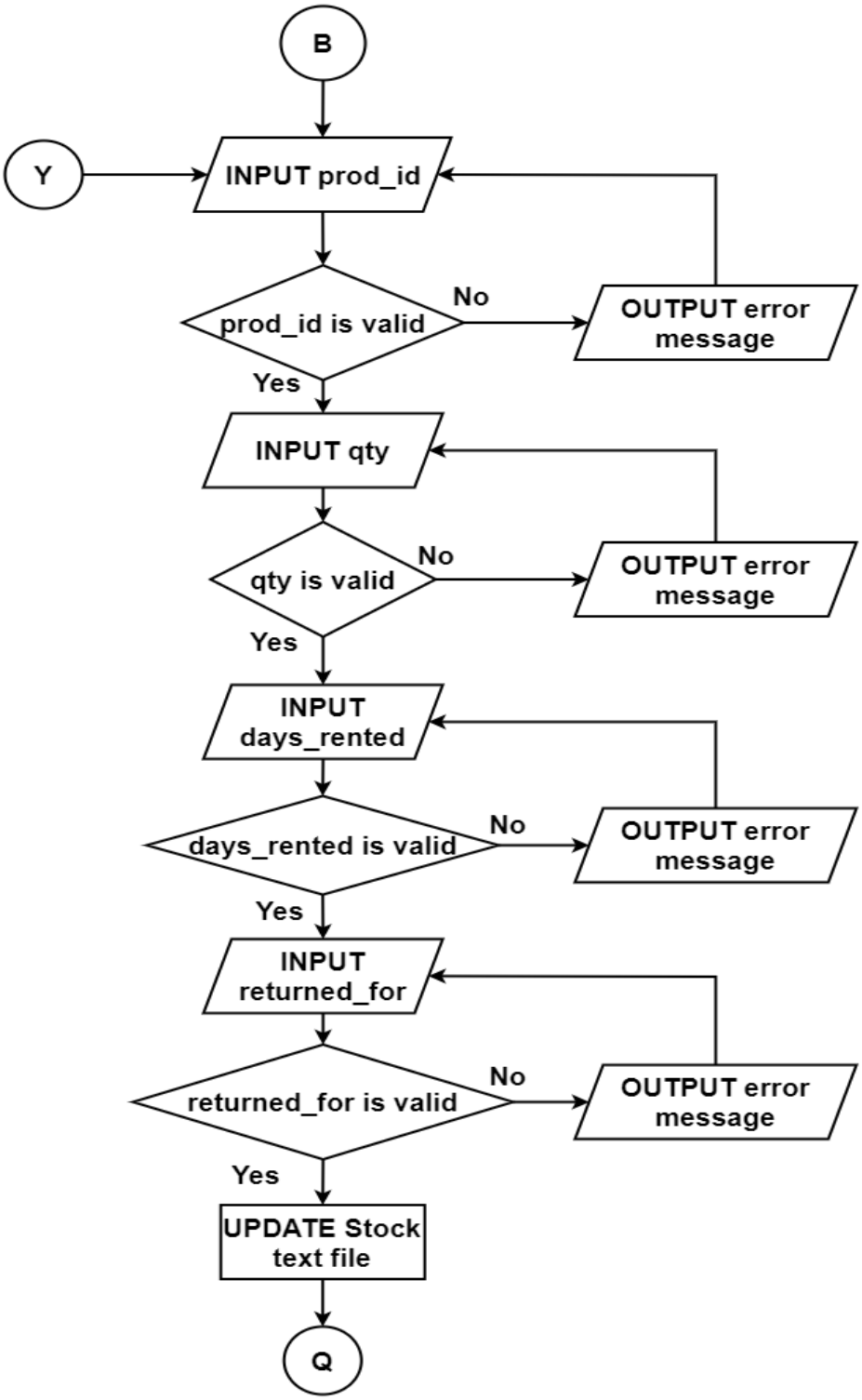


Figure 4: Flowchart of Return process (part 1)

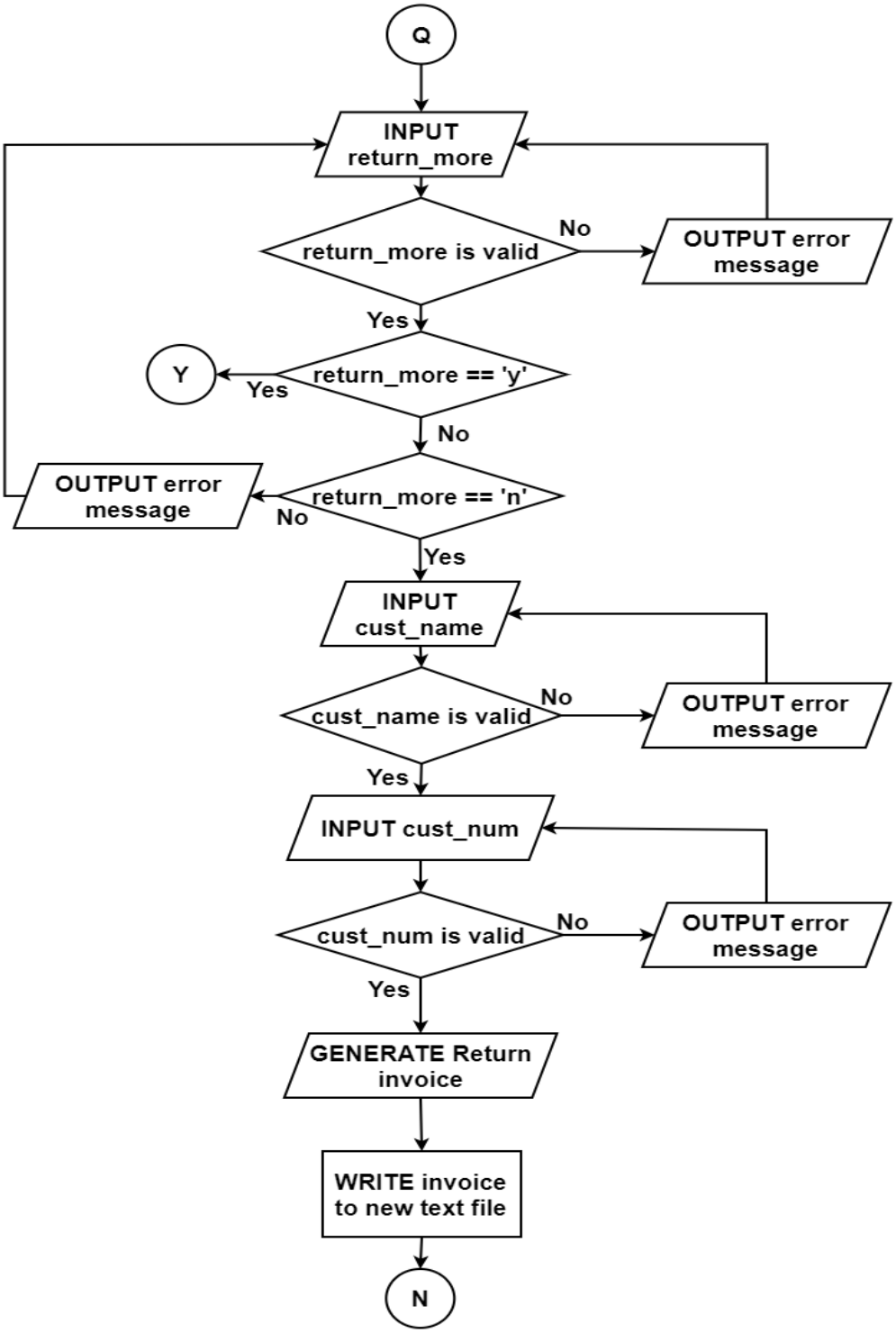


Figure 5: Flowchart of Return process (part 2)

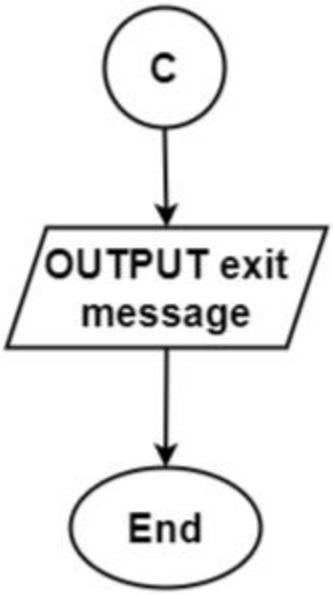


Figure 6: Flowchart of Exit process

2.3. Pseudocode

Pseudocodes are informal, programming language independent ways to represent the lines of code in a program. It explains each line of code in a concise manner without relying on any programming language, so that programmers who are familiar with any language can follow and understand the program.

The pseudocodes for this program are given below for each module:

2.3.1. Main module

```
Import Shop class from operations module
Import Read class from read module
Create class Main
    Call function Shop.show_info()
    Call function Shop.show_welcome_message()
    prod_dict = Call function Read.file_to_dict()
    exit_program = False
    While exit_program == False:
        Call function Shop.show_options()
        option = Call function Shop.valid_option(prod_dict)
        If option == 1 then
            Call function Shop.rent_operations(prod_dict)
        Elif option == 2 then
            Call function Shop.return_operations(prod_dict)
        Else
```

```
                exit_program = True
            End if
        End while
        Call function Shop.show_exit_message()
End class Main
```

2.3.2. Operations module

Import Write class from write module

Create class Shop

```
    Define function rent_operations(prod_dict)
        rent_list = []
        continue_renting = True
        While continue_renting == True:
            Call function Shop.display_stock(prod_dict)

            prod_id = Call function Rent.valid_prod_id(prod_dict)
            qty = Call function Rent.valid_prod_qty(prod_dict, prod_id)
            rented_for = Call function Rent.valid_rented_for()

            rent_prod = Call function Rent(prod_dict, prod_id, qty, rented_for)
            Call function Product.update_stock(rent_prod,prod_dict)
            rent_list.append(rent_prod)

            prod_not_available = Call function
            Shop.are_no_prods_available(prod_dict)
            If prod_not_available == True then
                continue_renting = False
```

```
            Else
                rent_choice = Call function Rent.get_rent_choice()
                If rent_choice == True then
                    continue_renting = True
                Else
                    continue_renting = False
                Endif
            End if
        End while
        Call function Bill.print_invoice_message()
        cust_name = Call function Shop.get_cust_name()
        cust_num = Call function Shop.get_cust_num()

        rent_bill = Call function Rent.create_bill(rent_list,cust_name,cust_num)
        Output rent_bill
        Call function Write.rent_bill_to_file(rent_bill)
    End function rent_operations

    Define function return_operations(prod_dict)
        return_list = []
        continue_returning = True
        While continue_returning == True:
            Call function Shop.display_stock(prod_dict)
```

```
        prod_id  = Call function Return.valid_prod_id(prod_dict)
        qty = Call function Return.valid_prod_qty()
        days_rented = Call function Return.valid_days_rented()
        returned_after = Call function Return.valid_returned_after()

        return_prod = Call function Return(prod_dict, prod_id, qty,
        days_rented, returned_after)
        Call function Product.update_stock(return_prod, prod_dict)
        return_list.append(return_prod)

        return_choice = Call function Return.get_return_choice()
        If return_choice == True then
            continue_returning = True
        Else
            continue_returning = False
        Endif
    End while
    Call function Bill.print_invoice_message()
    cust_name = Call function Shop.get_cust_name()
    cust_num = Call function Shop.get_cust_num()
```

```
        return_bill = Call function
        Return.create_bill(return_list,cust_name,cust_num)

        Output return_bill

        Call function Write.return_bill_to_file(return_bill)

End function return_operations


Define function show_info()

        shop_name = "Shirshak's Rent-an-Equipment Shop"

        shop_address = "Address: Maligaun 05, Kathmandu"

        shop_number = "Phone: 9544443333"

        shop_email = "Email: shirshak_rentalshop@gmail.com"


        shop_info = []

        shop_info.append(shop_name)

        shop_info.append(shop_address)

        shop_info.append(shop_number)

        shop_info.append(shop_email)


        num_of_equals_sign = 129

        equals_sign = Call function Misc.print_equals_sign(num_of_equals_sign)


        centering_num = 129
```

```
        Output equals_sign
        For each_info in shop_info:
            Output each_info.center(centering_num)
        End for
        Output equals_sign
End function show_info


Define function show_welcome_message()
        welcome_msg = “Welcome to Shirshak’s Rent-an-Equipment Shop! Please
        see the instructions below to carry out your desired tasks!”


        num_of_dashes = 122
        dashes = Call function Misc.print_dashes(num_of_dashes)
        Output dashes
        Output welcome_msg
        Output dashes + "\n"
End function show_welcome_message


Define function show_options()
        opt1 = “[1] Press 1 to rent an equipment.”
        opt2 = “[2] Press 2 to return an equipment.”
        opt3 = “[3] Press 3 to exit the program.\n”
```

```
        option_list = []
        option_list.append(opt1)
        option_list.append(opt2)
        option_list.append(opt3)

        For option in option_list:
            Output option
        End for
End function show_options

Define function display_stock(prod_dict)
        stock_list = [[item for item in each] for each in prod_dict.values()]
        header_row = ["Product","Brand","Price","Stock"]
        stock_list.insert(0,header_row)
        aligned_stock_list = Call function Misc.align_2D_list(stock_list)

        num_of_dashes = Call function
        Misc.get_num_of_dashes(aligned_stock_list)
        dashes = Call function Misc.print_dashes(num_of_dashes)

        Output dashes
        Output “ID\t” + "".join(aligned_stock_list[0])
        Output dashes
```

```
        p_id = 1
        For i = 1; i < len(aligned_stock_list); i++:
            Output str(p_id) + "\t" + "".join(aligned_stock_list[i])
            p_id += 1
        Output dashes
        End for
End function display_stock


Define function show_exit_message()
        Output "\nThank you for using our application."
End function show_exit_message


Define function valid_option(prod_dict)
        While True:
            Try:
                Input option
                If option == 1 then
                    no_prods_available = Call function
                    Shop.are_no_prods_available(prod_dict)
                    If no_prods_available == True then
                        Output "Sorry, no equipment is available for
                        rent right now!\n"
```

```
                    Else
                        Return option
                    End if
                Elif option == 2 or option == 3 then
                    Return option
                Else
                    Output "Invalid option; please input either 1, 2 or 3\n"
                End if
            Except:
                Output "Invalid input; please enter the option in numbers,
                either 1, 2 or 3\n"
            End try
        End while
    End function valid_option


    Define function are_no_prods_available(prod_dict)
        all_stock_sum = 0


        For each in prod_dict.values():
            all_stock_sum += int(each[-1])
        End for


        If all_stock_sum == 0 then
```

```
            Return True
        Else
            Return False
        End if
    End function are_no_prods_available


    Define function get_cust_name()
        While True:
            Input cust_name
            If len(cust_name) == 0 then
                Output "Invalid input; please enter the customer name\n"
            Else
                Return cust_name
            End if
        End while
    End function get_cust_name


    Define function get_cust_num()
        While True:
            Input cust_num
            If len(cust_num) == 0 then
                Output "Invalid input; please enter the customer’s phone
                number\n"
```

```
            Else
                Return cust_num
            End if
        End while
    End function get_cust_num


    Define function get_details_for_bill()
        num_of_dashes = 122
        dashes = Call function Misc.print_dashes(num_of_dashes)


        centering_num = 129
        details_for_bill = "\n" + dashes + "\n"
        details_for_bill += "Shirshak's Rent-an-Equipment
        Shop".center(centering_num) +    "\n"
        details_for_bill += "Address: Maligaun 05,
        Kathmandu".center(centering_num) + "\n"
        details_for_bill += "Phone: 9544443333".center(centering_num) + "\n"
        details_for_bill += "Email:
        shirshak_rentalshop@gmail.com".center(centering_num) + "\n"
        details_for_bill += dashes + "\n"


        Return details_for_bill
    End function get_details_for_bill
```

```
End class Shop


Create class Product
    Define function __init__ (self, prod_dict, prod_id, qty)
        self.ID = prod_id
        self.name = prod_dict[prod_id][0]
        self.brand = prod_dict[prod_id][1]
        self.price = prod_dict[prod_id][2]
        self.stock = prod_dict[prod_id][3]
        self.qty = qty
    End function __init__


    Define function get_grand_total(prod_list)
        grand_total = 0
        For each in prod_list:
            grand_total += each.total_price
        End for
        Return grand_total
    End function get_grand_total


    Define function update_stock(prod, prod_dict)
        current_stock = int(prod.stock)
        qty = prod.qty
```

```
            updated_stock = 0

            If isinstance(prod,Rent) then
                updated_stock = current_stock - qty
            Else
                updated_stock = current_stock + qty
            End if
            prod_dict[prod.ID][-1] = str(updated_stock)
            Call function Write.dict_to_file(prod_dict)
        End function update_stock
End class Product

Create class Rent(Product)
        Define function __init__(self, prod_dict, prod_id, qty, rented_for)
            Call function super().__init__(prod_dict, prod_id, qty)

            self.rented_for = rented_for

            price = prod_dict[prod_id][2]
            price_factor = Call function Rent.get_price_factor(rented_for)
            self.total_price = Call function Rent.get_total_price(price, price_factor, qty)
        End function __init__
```

```
Define function get_price_factor(rented_for)
    price_factor = 0

    If rented_for % 5 == 0 then
        price_factor = rented_for/5
    Else
        price_factor = (rented_for // 5) + 1
    End if
    Return price_factor
End function get_price_factor

Define function get_total_price(price, price_factor, qty)
    total_price = price_factor * int(price.replace("$","")) * qty
    Return total_price
End function get_total_price

Define function valid_prod_id(prod_dict)
    While True:
        Try:
            Input prod_id
            If 1 <= prod_id <= len(prod_dict) then
                If int(prod_dict[prod_id][-1]) == 0 then
```

```
                    Output "Sorry, this product is not available right now!\n"
                Else
                    Return prod_id
                End if
            Else
                Output "Invalid product ID; please enter a number from 1 to 5\n"
            End if
        Except:
            Output "Invalid input; please enter the product ID in numbers\n"
        End try
    End while
End function valid_prod_id


Define function valid_prod_qty(prod_dict, prod_id)
    While True:
        Try:
            Input qty
            If 0 < qty <= int(prod_dict[prod_id][-1]) then
                Return qty
            Else
```

```
                    Output "Invalid input; please enter a number within 1
                    and the available stock\n"
                End if
            Except:
                Output "Invalid input; please enter the quantity in numbers\n"
            End try
        End while
    End function valid_prod_qty


    Define function valid_rented_for()
        While True:
            Try:
                Input rented_for
                If rented_for > 0 then
                    Return rented_for
                Else
                    Output "Invalid input; please enter a positive value for
                    the number of days\n"
                End if
            Except:
                Output "Invalid input; please enter the number of days in
                numbers\n"
            End try
```

```
        End while
    End function valid_rented_for


    Define function get_rent_choice()
        While True:
            Input rent_more


            If rent_more == “y” then
                Return True
            Elif rent_more == “n” then
                Return False
            Else
                Output "Invalid answer; please enter either 'y' or 'n'\n"
            End if
        End while
    End function get_rent_choice


    Define function create_bill(rent_list, cust_name, cust_num)
        bill_text = Call function Shop.get_details_for_bill()
        bill_text += "\nInvoice ID: " + Call function Rent.get_bill_id() + “\n”
        bill_text += Call function Bill.create_bill(rent_list, cust_name, cust_num) +
        “\n”
        Return bill_text
```

End function create_bill

Define function get_bill_id()

bill_id = “rent” + **Call function** Write.get_unique_id()

Return bill_id

End function get_bill_id

End class Rent

Create class Return(Product)

Define function __init__(self, prod_dict, prod_id, qty, days_rented, returned_after)

Call function super().__init__(prod_dict, prod_id, qty)

self.days_rented = days_rented

self.returned_after = returned_after

price = prod_dict[prod_id][2]

price_factor_for_days_rented = **Call function** Rent.get_price_factor(days_rented)

base_price = **Call function** Rent.get_total_price(price,price_factor_for_days_rented,qty)

```
        self.fine = Call function Return.get_fine(price, days_rented,
        returned_after, qty)

        self.total_price = base_price + self.fine

End function __init__


Define function get_fine(price, days_rented, returned_after, qty)

        price_factor_for_days_rented = Call function
        Rent.get_price_factor(days_rented)

        price_factor_for_returned_after = Call function
        Rent.get_price_factor(returned_after)


        fine = 0

        If price_factor_for_days_rented >= price_factor_for_returned_after then

                fine = 0

        Else

                fine_days = returned_after - (price_factor_for_days_rented * 5)

                fine = (int(price.replace("$",""))/5) * fine_days * qty

        End if

        Return round(fine,2)

End function get_fine


Define function valid_prod_id(prod_dict)

        While True:
```

```
            Try:
                Input prod_id
                If 1 <= prod_id <= len(prod_dict) then
                    Return prod_id
                Else
                    Output "Invalid product ID; please enter a number from
                    1 to 5\n"
                End if
            Except:
                Output "Invalid input; please enter the product ID in
                numbers\n"
            End try
        End while
    End function valid_prod_id


    Define function valid_prod_qty()
        While True:
            Try:
                Input qty
                If qty > 0 then
                    Return qty
                Else
```

```
                    Output "Invalid input; please enter a number greater
                    than 0\n"
            End if
        Except:
            Output "Invalid input; please enter the quantity in numbers\n"
        End try
    End while
End function valid_prod_qty


Define function valid_days_rented()
    While True:
        Try:
            Input days_rented
            If days_rented > 0 then
                Return days_rented
            Else
                Output "Invalid input; please enter a number greater
                than 0\n"
            End if
        Except:
            Output "Invalid input; please enter the number of days in
            numbers\n"
        End try
```

```
        End while
    End function valid_days_rented


    Define function valid_returned_after()
        While True:
            Try:
                Input returned_after
                If returned_after > 0 then
                    Return returned_after
                Else
                    Output "Invalid input; please enter a non-negative
                    value\n"
                End if
            Except:
                Output "Invalid input; please enter the number of days in
                numbers\n"
            End try
        End while
    End function valid_returned_after


    Define function get_return_choice()
        While True:
            Input return_more
```

```
            If return_more == “y” then
                Return True
            Elif return_more == “n” then
                Return False
            Else
                Output "Invalid answer; please enter either 'y' or 'n'\n"
            End if
        End while
    End function get_return_choice

    Define function create_bill(return_list, cust_name, cust_num)
        bill_text = Call function Shop.get_details_for_bill()
        bill_text += "\nInvoice ID: " + Call function Return.get_bill_id() + “\n”
        bill_text += Call function Bill.create_bill(return_list, cust_name, cust_num)
        + “\n”
        Return bill_text
    End function create_bill

    Define function get_bill_id()
        bill_id = “return” + Call function Write.get_unique_id()
        Return bill_id
    End function get_bill_id
End class Return
```

```
Create class Bill
    Define function get_aligned_bill_list(prod_list)
        bill_list = []

        For each in prod_list:
            If isinstance(each,Rent) then
                bill_list.append([
                    str(each.ID),
                    each.name,
                    each.brand,
                    each.price,
                    str(each.qty),
                    str(each.rented_for),
                    "$" + str(each.total_price)
                ])
            Else
                bill_list.append([
                    str(each.ID),
                    each.name,
                    each.brand,
                    each.price,
                    str(each.qty),
```

```
                    str(each.days_rented),
                    str(each.returned_after),
                    "$" + str(each.fine),
                    "$" + str(each.total_price)
                ])
            End if
        End for

        header_row = []
        If isinstance(prod_list[0],Rent) then
            header_row = Call function Bill.get_rent_bill_header_row()
        Else
            header_row = Call function Bill.get_return_bill_header_row()
        End if

        bill_list.insert(0,header_row)
        aligned_bill_list = Call function Misc.align_2D_list(bill_list)

        Return aligned_bill_list
    End function get_aligned_bill_list

    Define function get_bill_prod_details(aligned_bill_list)
        bill_prod_details = “”
```

```
        sn = 1

        For i = 1; i < len(aligned_bill_list); i++:
            bill_prod_details += str(sn) + "\t" + "".join(aligned_bill_list[i]) + "\n"
            sn += 1
        End for

        Return bill_prod_details
    End function get_bill_prod_details

    Define function get_bill_cust_details(cust_name,cust_num)
        bill_cust_details = "\nCustomer name: " + cust_name
        bill_cust_details += "\nCustomer phone number: " + cust_num + "\n"
        Return bill_cust_details
    End function get_bill_cust_details

    Define function get_bill_grand_total(grand_total)
        bill_grand_total = "Grand total price: $" + str(round(grand_total,2)) + "\n"
        Return bill_grand_total
    End function get_bill_grand_total

    Define function bill_header_row(aligned_bill_list)
        bill_header = "SN\t" + "".join(aligned_bill_list[0]) + "\n"
```

```
        Return bill_header
End function bill_header_row


Define function get_rent_bill_header_row()
        header_row = ["ID","Product","Brand","Price","Qty","Rented For","Total"]
        Return header_row
End function get_rent_bill_header_row


Define function get_return_bill_header_row()
        header_row = ["ID","Product","Brand","Price","Qty","Rented For",
        "Returned After","Fine","Total"]
        Return header_row
End function get_return_bill_header_row


Define function print_invoice_message()
        Output "\nFor generating invoice: "
End function print_invoice_message


Define function create_bill(prod_list, cust_name, cust_num)
        grand_total = Call function Product.get_grand_total(prod_list)


        aligned_bill_list = Call function Bill.get_aligned_bill_list(prod_list)
```

```
        num_of_dashes = Call function
        Misc.get_num_of_dashes(aligned_bill_list)

        bill_text = Call function Write.get_current_date()
        bill_text += Call function Write.get_current_time()

        bill_text += Call function Misc.print_dashes(num_of_dashes) + "\n"
        bill_text += Call function Bill.bill_header_row(aligned_bill_list)
        bill_text += Call function Misc.print_dashes(num_of_dashes) + "\n"

        bill_text += Call function Bill.get_bill_prod_details(aligned_bill_list)
        bill_text += Call function Bill.get_bill_cust_details(cust_name,cust_num)

        bill_text += Call function Misc.print_dashes(num_of_dashes) + "\n"
        bill_text += Call function Bill.get_bill_grand_total(grand_total)
        bill_text += Call function Misc.print_dashes(num_of_dashes) + "\n"

        Return bill_text
    End function create_bill
End class Bill

Create class Misc
    Define function align_2D_list(list_2D)
```

```
        space_l = [max(len(each[i]) for each in list_2D) for i in range(len(list_2D[0]))]

        padding = 3

        For each in list_2D:
            For i = 0; i < len(list_2D[0]), i++:
                each[i] = each[i].ljust(space_l[i] + padding)
            End for
        End for
        Return list_2D
    End function align_2D_list

    Define function get_num_of_dashes(list_2D)
        len_of_each_row = []

        For i = 0; i < len(list_2D); i++:
            len_of_each_row.append(sum(len(item) for item in list_2D[i]))
        End for

        num_of_dashes = max(len_of_each_row)

        Return num_of_dashes
    End function get_num_of_dashes
```

```
    Define function print_dashes(num_of_dashes)
        surplus = 7
        dashes = "-" * (num_of_dashes + surplus)
        Return dashes
    End function print_dashes

    Define function print_equals_sign(num_of_sign)
        surplus = 0
        equals = "=" * (num_of_sign + surplus)
        Return equals
    End function print_equals_sign
End Misc
```

2.3.3. Read module

```
Create class Read

    Define function file_to_dict()

        stock_file = open("stockfile.txt", "r")

        data = stock_file.read()

        data = data.split("\n")


        prod_dict = {}

        c = 1


        For i = 0; i < len(data), i++:

            prod_dict[c] = data[i].split(",")

            c += 1

        End for


        stock_file.close()

        Return prod_dict

    End function file_to_dict

End class Read
```

2.3.4. Write module

Import datetime library

Create class Write

```
    Define function dict_to_file(prod_dict)
        updated_info = ""

        For value in prod_dict.values():
            updated_info += ",".join(value) + "\n"
        End for

        updated_info = updated_info.rstrip("\n")

        stock_file = open("stockfile.txt","w")
        stock_file.write(updated_info)
        stock_file.close()
    End function dict_to_file

    Define function rent_bill_to_file(bill_text)
        file_name = Call function Write.get_rent_invoice_file_name()

        new_rent_file = open(file_name,"w")
```

```
        new_rent_file.write(bill_text)
        new_rent_file.close()
End function rent_bill_to_file

Define function return_bill_to_file(bill_text)
        file_name = Call function Write.get_return_invoice_file_name()

        new_rent_file = open(file_name,"w")
        new_rent_file.write(bill_text)
        new_rent_file.close()
End function return_bill_to_file

Define function get_unique_id()
        year = str(datetime.datetime.now().year)
        month = str(datetime.datetime.now().month)
        day = str(datetime.datetime.now().day)

        hour = str(datetime.datetime.now().hour)
        minute = str(datetime.datetime.now().minute)
        second = str(datetime.datetime.now().second)

        unique_id = year + month + day + "_" + hour + minute + second
```

```
        Return unique_id
End function get_unique_id


Define function get_rent_invoice_file_name()
        file_id = Call function Write.get_unique_id()
        unique_file_name = "Rent Invoices/rent" + file_id + ".txt"
        Return unique_file_name
End function get_rent_invoice_file_name


Define function get_return_invoice_file_name()
        file_id = Call function Write.get_unique_id()
        unique_file_name = "Return Invoices/return" + file_id + ".txt"
        Return unique_file_name
End function get_return_invoice_file_name


Define function get_current_date()
        current_date = "Date: " + str(datetime.datetime.now().date()) + "\t"
        Return current_date
End function get_current_date


Define function get_current_time()
        current_time = "Time: " +
        str(datetime.datetime.now().time().replace(microsecond=0)) + "\n"
```

```
            Return current_time

        End function get_current_time

End class Write
```

2.4. Data Structures

Data structures are the different types of variables that are designed for storing different types of data. Data structures in Python are broadly divided into two categories: **Primitive** and **Collection** data structures.

Primitive data structures are designed for holding simple values, such as those listed below:

- Integers,
- Floating point numbers,
- Boolean values,
- Strings, etc.

Collection data structures, on the other hand, are designed for holding a collection of primitive data structures, other complex data structures, or a mixture of both as well. Examples of collection data structures include:

- Lists
- Tuples
- Sets
- Dictionaries

Python also allows user-defined data structures using classes and objects.

2.4.1. Data structures used in this project

The data structures used in this project are mainly integers, floating numbers, strings, Booleans, lists and dictionaries. Evidences of using such data structures along with the reasoning is illustrated below:

- **Integers and Floating point numbers** were mainly used to store whole number and decimal point values (respectively) to be used in arithmetic calculations, such as updating the stock, calculating the total prices, calculating the price per day of an item, etc.

Use of integer:

```
@staticmethod
def update_stock(prod, prod_dict):
    '''args: rented/returned product, product dictionary
    function: updates stock of rented/returned product in the product dictionary, and writes the dictionary to stock file
    returns: None'''

    current_stock = int(prod.stock)  # setting current stock of product
    qty_rented = prod.qty  # setting product qty
    updated_stock = 0  # setting updated product stock to be modified below

    # calculating updated product stock
    if isinstance(prod,Rent):  # if product is rented
        updated_stock = current_stock - qty_rented  # subtracting rented qty from current stock
    else:  # if product is returned
        updated_stock = current_stock + qty_rented  # adding returned qty to current stock

    prod_dict[prod.ID][-1] = str(updated_stock)  # updating stock value in product dictionary
    Write.dict_to_file(prod_dict)  # writing updated stock to stock file
```

Figure 7: An int data structure used to store the value of updated stock

Use of floating point number:

```
@staticmethod
def get_fine(price, days_rented, returned_after, qty):
    '''args: price, days rented, days returned after
    function: calculates the fine
    returns: the fine'''

    # getting price factor for days rented
    price_factor_for_days_rented = Rent.get_price_factor(days_rented)
    # getting price factor for days returned after
    price_factor_for_returned_after = Rent.get_price_factor(returned_after)

    '''this if statement checks if price factor for days rented and days returned after is same
    if they are same, it sets the fine to 0
    else it sets the fine as the product of the extra days and price for 5 days'''
    fine = 0  # setting a variable to store the fine
    if price_factor_for_days_rented >= price_factor_for_returned_after:  # if price factor is same
        fine = 0  # setting fine to 0
    else:  # if price factor is different
        fine_days = returned_after - (price_factor_for_days_rented * 5)  # calculating num of days to be fined for
        fine = (int(price.replace("$",""))/5) * fine_days * qty  # calculating fine
```

Figure 8: Floating point data structure used to store the fine applied in returning process

- **String** data structures were mainly used to store the messages that are displayed at various points of operation of the code.

```python
@staticmethod
def show_welcome_message():
    '''args: None
    function: prints a welcome message
    returns: None'''

    # setting a welcome message
    welcome_msg = "Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!"

    num_of_dashes = 122  # setting num of dashes to be printed
    dashes = Misc.print_dashes(num_of_dashes)  # setting dashes

    print(dashes)  # printing dashes
    print(welcome_msg)  # printing welcome message
    print(dashes + "\n")  # printing dashes
```

Figure 9: A string data structure used to store the welcome message of the program

- **Lists** were mainly used to store the products currently being rented or returned.

operations.py - C:\Users\Shirshak\OneDrive - London Metropolitan University\Islington Non-Lecture material\FOC\Python Coursework Main Folder\Python Coursework IDLE\Python_coursework-C

File Edit Format Run Options Window Help

```python
class Shop():  # creating a class Shop

    @staticmethod
    def rent_operations(prod_dict):
        '''args: product dictionary
        function: carries out renting operations
        returns: None'''

        rent_list = []  # creating a list to store products currently being rented

        # this while loop takes user input, carries out renting operations,
        # and repeatedly asks user if they want to rent more, until they enter 'n'
        continue_renting = True  # setting a flag
        while continue_renting == True:
            Shop.display_stock(prod_dict)  # displaying the info from the stock file

            prod_id = Rent.valid_prod_id(prod_dict)  # validating product ID input from user
            qty = Rent.valid_prod_qty(prod_dict, prod_id)  # validating product qty input from user
            rented_for = Rent.valid_rented_for()  # validating num of days rented for input from user

            rent_prod = Rent(prod_dict, prod_id, qty, rented_for)  # creating Rent object
            Product.update_stock(rent_prod,prod_dict)  # updating stock of the object
            rent_list.append(rent_prod)  # adding the product to the rent product list
```

Figure 10: A list used to store currently rented items in renting process

```
@staticmethod
def return_operations(prod_dict):
    '''args: product dictionary
    function: carries out returning operations
    returns: None'''

    return_list = []  # creating a list to store products currently being returned

    # this while loop takes user input, carries out returning operations,
    # and repeatedly asks user if they want to returning more, until they enter 'n'
    continue_returning = True  # setting a flag
    while continue_returning == True:
        Shop.display_stock(prod_dict)  # displaying the info from the stock file

        prod_id = Return.valid_prod_id()  # validating product ID input from user
        qty = Return.valid_prod_qty()  # validating product qty input from user
        returned_after = Return.valid_returned_after()  # validating num of days returned after input from user

        return_prod = Return(prod_dict, prod_id, qty, returned_after)  # creating product object
        Product.update_stock(return_prod, prod_dict)  # # updating stock of the product object
        return_list.append(return_prod)  # adding the product to the return product list

        return_choice = Return.get_return_choice()  # validating input from user if they want to return more
        if return_choice == True:  # if user wants to rent more
            continue_returning = True  # setting flag to true, continuing this while loop
        else:  # if user does not want to rent more
            continue_returning = False  # setting flag to false, breaking this while loop
```

Figure 11: A list used to store currently returned items in returning process

- A **dictionary** was used to store the information in the stock file and provide custom indices (i.e., keys) to the items in it for convenience of accessing them at other parts of the code.

```python
class Read():  # creating a class Read

    @staticmethod
    def file_to_dict():
        '''args: None
        function: reads the stock file and returns the info in a dictionary
        returns: a dictionary with the info in the stock file'''

        stock_file = open("stockfile.txt", "r")  # opening the stock file
        data = stock_file.read()  # reading the stock file
        data = data.split("\n")  # splitting the data per each new line

        prod_dict = {}  # creating a dictionary to store the data
        c = 1  # setting the counter for the dictionary keys

        # this loop adds the list of details of each product to the dictionary
        for i in range(len(data)):
            prod_dict[c] = data[i].split(",")
            c += 1  # increasing the counter by 1 each time

        stock_file.close()  # closing the stock file

        return prod_dict  # returning prod_dict
```

Figure 12: A dictionary used to store the product details after reading from stock text file

- **Boolean** data structures were used mainly as flags for while loops in the code.

```
class Main():  # creating a class Main

    Shop.show_info()  # displaying shop details
    Shop.show_welcome_message()  # displaying welcome message

    prod_dict = Read.file_to_dict()  # storing product info from stock file in a

    # this loop is the main loop of the whole program,
    # and is exited only when option 3 is selected
    exit_program = False  # setting a flag
    while exit_program == False:
        Shop.show_options()  # displaying the rent, return and exit options
        option = Shop.valid_option(prod_dict)  # asking input for the options an

        if option == 1:  # if user wants to rent a product
            Shop.rent_operations(prod_dict)  # carrying out the rent operations

        elif option == 2:  # if user wants to return a prouct
            Shop.return_operations(prod_dict)  # carrying out the return operati

        else:  # if user wants to exit the program
            exit_program = True  # setting flag to false, breaking the main loop

    Shop.show_exit_message()  # printing exit message after user exits the progr
```

Figure 13: A boolean data structure used as a flag for the while loop in the main module

3. Program

The program starts by displaying the information related to the shop, like the name, contact number, address, etc. It then displays a message welcoming the user to the program interface. It then reads the stock text file and stores the information in a dictionary, then displays the 3 options a user can choose from, i.e., rent, return, or exit.

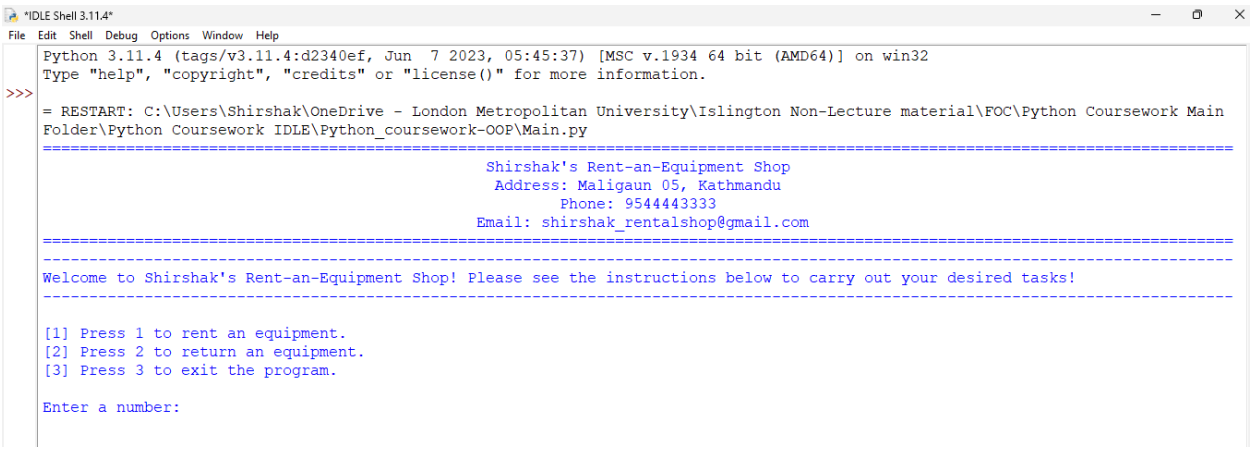


Figure 14: The shop details, welcome message, and the 3 options shown

After it asks the user to enter an option and validates it, the processes that follow are described below:

3.1. Rent

If the user chooses to rent a product, the program first asks the user for entering the product ID, product quantity, and the number of days the product is being rented for. It does so by validating each input and not progressing to the next until ensuring that the previous input is valid. After that, the program updates the stock of the product, utilizing the product ID as the key to the dictionary and subtracting the quantity of the product from the current stock.

The program then asks the user if they want to rent more products. If the user enters ‘y’, the program restarts the process by asking the product ID. This loop continues to run until the user enters ‘n’, or the user rents all the products to the point where none are left.

After exiting the loop, it asks the user to enter the customer’s name and phone number, validates each input, and prints out an invoice while at the same time writing the invoice to a new text file.

After this process ends, the program restarts by displaying the 3 options to the user.

```
Enter a number: 1
--------------------------------------------------------------------------
ID      Product                           Brand              Price   Stock
--------------------------------------------------------------------------
1       Velvet Table Cloth                Saathi             $8      100
2       Microphone Set                    Audio Technica     $189    91
3       Disco Light Set                   Sonoff             $322    100
4       7.1 Surround Sound Speaker Set    Dolby              $489    100
5       Dinner Table 8x5                  Panda Furnitures   $344    100
--------------------------------------------------------------------------
Enter product ID (1 to 5): 1
Enter how many items to rent: 3
How many days is this product being rented for?: 4
Do you want to rent more products? (y/n): y
--------------------------------------------------------------------------
ID      Product                           Brand              Price   Stock
--------------------------------------------------------------------------
1       Velvet Table Cloth                Saathi             $8      97
2       Microphone Set                    Audio Technica     $189    91
3       Disco Light Set                   Sonoff             $322    100
4       7.1 Surround Sound Speaker Set    Dolby              $489    100
5       Dinner Table 8x5                  Panda Furnitures   $344    100
--------------------------------------------------------------------------
Enter product ID (1 to 5): 2
Enter how many items to rent: 1
How many days is this product being rented for?: 5
Do you want to rent more products? (y/n): n

For generating invoice:
```

Figure 15: Asking inputs for product ID, quantity, and number of days rented in renting process

```
For generating invoice:
Enter customer name: Shirshak
Enter customer phone number: 9388477263

--------------------------------------------------------------------------------------------
                                        Shirshak's Rent-an-Equipment Shop
                                         Address: Maligaun 05, Kathmandu
                                                Phone: 9544443333
                                       Email: shirshak_rentalshop@gmail.com
--------------------------------------------------------------------------------------------

Invoice ID: rent2023823_184916
Date: 2023-08-23        Time: 18:49:16
-------------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Total
-------------------------------------------------------------------------------------------
1       1    Velvet Table Cloth   Saathi           $8      3     4            $24
2       2    Microphone Set       Audio Technica   $189    1     5            $189.0

Customer name: Shirshak
Customer phone number: 9388477263
-------------------------------------------------------------------------------------------
Grand total price: $213.0
-------------------------------------------------------------------------------------------
```

Figure 16: Asking input for customer details and generating the unique invoice in shell window

Name	Status	Date modified	Type	Size
rent2023823_184917		8/23/2023 6:49 PM	Text Document	2 KB

Figure 17: Text file with the above invoice

```
--------------------------------------------------------------------------------------------------------------------------------
                                                Shirshak's Rent-an-Equipment Shop
                                                 Address: Maligaun 05, Kathmandu
                                                         Phone: 9544443333
                                               Email: shirshak_rentalshop@gmail.com
--------------------------------------------------------------------------------------------------------------------------------

Invoice ID: rent2023823_184916
Date: 2023-08-23        Time: 18:49:16
-------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Total
-------------------------------------------------------------------------------------
1       1    Velvet Table Cloth   Saathi           $8      3     4            $24
2       2    Microphone Set       Audio Technica   $189    1     5            $189.0

Customer name: Shirshak
Customer phone number: 9388477263
-------------------------------------------------------------------------------------
Grand total price: $213.0
-------------------------------------------------------------------------------------
```

Figure 18: The above invoice inside the text file

3.2. Return

If the user chooses to return a product, the program first asks the user for entering the product ID, product quantity, number of days rented, and the number of days the product is being returned after. It does so by validating each input and not progressing to the next until ensuring that the previous input is valid. After that, the program updates the stock of the product, utilizing the product ID as the key to the dictionary and adding the quantity of the product from the current stock.

The program then asks the user if they want to return more products. If the user enters ‘y’, the program restarts the process by asking the product ID. This loop continues to run until the user enters ‘n’.

After exiting the loop, it asks the user to enter the customer’s name and phone number, validates each input, and prints out an invoice while at the same time writing the invoice to a new text file.

After this process ends, the program restarts by displaying the 3 options to the user.

```
Enter a number: 2
-------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      103
2       Microphone Set                   Audio Technica     $189    92
3       Disco Light Set                  Sonoff             $322    99
4       7.1 Surround Sound Speaker Set   Dolby              $489    99
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-------------------------------------------------------------------------
Enter product ID (1 to 5): 1
Enter how many items to return: 3
How many days was this product rented for?: 3
How many days is this product being returned after?: 6
Do you want to return more products? (y/n): y
-------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      106
2       Microphone Set                   Audio Technica     $189    92
3       Disco Light Set                  Sonoff             $322    99
4       7.1 Surround Sound Speaker Set   Dolby              $489    99
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-------------------------------------------------------------------------
Enter product ID (1 to 5): 2
Enter how many items to return: 2
How many days was this product rented for?: 3
How many days is this product being returned after?: 4
Do you want to return more products? (y/n): n

For generating invoice:
```

Figure 19: Asking inputs for product Id, quantity, number of days rented for, and number of days returned after in renting process

```
For generating invoice:
Enter customer name: Shirshak
Enter customer phone number: 9837744555
-------------------------------------------------------------------------------------------------------------------
                                    Shirshak's Rent-an-Equipment Shop
                                     Address: Maligaun 05, Kathmandu
                                            Phone: 9544443333
                                   Email: shirshak_rentalshop@gmail.com
-------------------------------------------------------------------------------------------------------------------

Invoice ID: return2023824_105219
Date: 2023-08-24        Time: 10:52:19
-------------------------------------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Returned After   Fine   Total
-------------------------------------------------------------------------------------------------------------------
1       1    Velvet Table Cloth   Saathi           $8      3     3            6                $4.8   $28.8
2       2    Microphone Set       Audio Technica   $189    2     3            4                $0     $378

Customer name: Shirshak
Customer phone number: 9837744555
-------------------------------------------------------------------------------------------------------------------
Grand total price: $406.8
-------------------------------------------------------------------------------------------------------------------


[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.
```

Figure 20: Asking input for customer details and generating the unique invoice in shell window

Name	Status	Date modified	Type	Size
return2023824_105219		8/24/2023 10:52 AM	Text Document	2 KB

Figure 21: Text file with the above invoice

```
---------------------------------------------------------------------------------------------------------------------
                                                Shirshak's Rent-an-Equipment Shop
                                                 Address: Maligaun 05, Kathmandu
                                                        Phone: 9544443333
                                               Email: shirshak_rentalshop@gmail.com
---------------------------------------------------------------------------------------------------------------------

Invoice ID: return2023824_105219
Date: 2023-08-24        Time: 10:52:19
-------------------------------------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Returned After   Fine   Total
-------------------------------------------------------------------------------------------------------------------
1       1    Velvet Table Cloth   Saathi           $8      3     3            6                $4.8   $28.8
2       2    Microphone Set       Audio Technica   $189    2     3            4                $0     $378

Customer name: Shirshak
Customer phone number: 9837744555
-------------------------------------------------------------------------------------------------------------------
Grand total price: $406.8
-------------------------------------------------------------------------------------------------------------------
```

Figure 22: The above invoice in the text file

3.3. Exit

If the user chooses to exit the program, an exit message is shown to the user.

```
-------------------------------------------------------


[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 3

Thank you for using our application.
```

Figure 23: Exit message shown after user exits the program

4. Testing

4.1. Test 1: To show the implementation of exception handling

Objective	To show the implementation of exception handling
Action	1. String values were entered for the product ID, quantity, and number of days in renting process.
Expected Result	Error messages should have been shown for each of the steps above.
Actual Result	Error messages were shown for each of the steps above.
Conclusion	The test was successful.

Table 1: Test 1: To show the implementation of exception handling

```
==========================================================================================
                    Shirshak's Rent-an-Equipment Shop
                     Address: Maligaun 05, Kathmandu
                            Phone: 9544443333
                   Email: shirshak_rentalshop@gmail.com
==========================================================================================
------------------------------------------------------------------------------------------
Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!
------------------------------------------------------------------------------------------

[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): a
Invalid input; please enter the product ID in numbers

Enter product ID (1 to 5):
```

Figure 24: Error message shown after entering string value for product ID in renting process

```
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!
--------------------------------------------------------------------------------------------------------------------------------

[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): a
Invalid input; please enter the product ID in numbers

Enter product ID (1 to 5): 1
Enter how many items to rent: a
Invalid input; please enter the quantity in numbers

Enter how many items to rent: |
```

Figure 25: Error message shown after entering string value for product quantity in renting process

```
[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): a
Invalid input; please enter the product ID in numbers

Enter product ID (1 to 5): 1
Enter how many items to rent: a
Invalid input; please enter the quantity in numbers

Enter how many items to rent: 1
How many days is this product being rented for?: a
Invalid input; please enter the number of days in numbers

How many days is this product being rented for?: |
```

Figure 26: Error message shown after entering string value for number of days in renting process

4.2. Test 2: To test negative and non-existent values for the product ID in renting and returning processes

Objective	To test negative and non-existent values for the product ID in renting and returning processes
Action	1. The options for renting and returning products were entered (separately). 2. Negative values for the product ID were entered in both cases. 3. Non-existent values for the product ID were entered in both cases.
Expected Result	Error messages should have been shown foe each of the steps above.
Actual Result	Error messages were shown for each of the steps above.
Conclusion	The test was successful.

Table 2: Test 2: To test negative and non-existent values for the product ID in renting and returning processes

Entering negative value for product ID in renting:

```
==========================================================================================
                    Shirshak's Rent-an-Equipment Shop
                     Address: Maligaun 05, Kathmandu
                            Phone: 9544443333
                   Email: shirshak_rentalshop@gmail.com
==========================================================================================
------------------------------------------------------------------------------------------
Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!
------------------------------------------------------------------------------------------

[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): -1
Invalid product ID; please enter a number from 1 to 5

Enter product ID (1 to 5):
```

Figure 27: Error message shown after entering negative value for product ID in renting process

Entering non-existent value for product ID in renting:

```
------------------------------------------------------------------------------------------
Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!
------------------------------------------------------------------------------------------

[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): -1
Invalid product ID; please enter a number from 1 to 5

Enter product ID (1 to 5): 6
Invalid product ID; please enter a number from 1 to 5

Enter product ID (1 to 5):
```

Figure 28: Error message shown after entering non-existent value for product ID in renting process

Entering negative value for product ID in returning process:

```
==========================================================================================
                    Shirshak's Rent-an-Equipment Shop
                     Address: Maligaun 05, Kathmandu
                            Phone: 9544443333
                   Email: shirshak_rentalshop@gmail.com
==========================================================================================
------------------------------------------------------------------------------------------
Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!
------------------------------------------------------------------------------------------

[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 2
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): -11
Invalid product ID; please enter a number from 1 to 5

Enter product ID (1 to 5): |
```

Figure 29: Error message shown after entering negative value for product ID in returning process

Entering non-existent value for product ID in returning process:

```
------------------------------------------------------------------------------------------
Welcome to Shirshak's Rent-an-Equipment Shop! Please see the instructions below to carry out your desired tasks!
------------------------------------------------------------------------------------------

[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 2
-----------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-----------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-----------------------------------------------------------------------------
Enter product ID (1 to 5): -11
Invalid product ID; please enter a number from 1 to 5

Enter product ID (1 to 5): 6
Invalid product ID; please enter a number from 1 to 5

Enter product ID (1 to 5): |
```

Figure 30: Error message shown after entering non-existent value for product ID in returning process

4.3. Test 3: To check the generation of a text file containing the invoice created for renting multiple items

Objective	To check the generation of a text file containing the invoice created for renting multiple items
Action	1. The renting process was carried out.
Expected Result	A text file containing the invoice of the rented items should have been generated.
Actual Result	A text file containing the invoice of the rented items was generated.
Conclusion	The test was successful.

Table 3: Test 3: To check the generation of a text file containing the invoice created for renting multiple items

```
Enter a number: 1
--------------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
--------------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
--------------------------------------------------------------------------------
Enter product ID (1 to 5): 1
Enter how many items to rent: 2
How many days is this product being rented for?: 4
Do you want to rent more products? (y/n): y
--------------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
--------------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      86
2       Microphone Set                   Audio Technica     $189    96
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
--------------------------------------------------------------------------------
Enter product ID (1 to 5): 2
Enter how many items to rent: 3
How many days is this product being rented for?: 7
Do you want to rent more products? (y/n): n

For generating invoice:
```

Figure 31: Renting multiple items

```
For generating invoice:
Enter customer name: Shirshak
Enter customer phone number: 9834827736

--------------------------------------------------------------------------------------------------------------------------------
                                                Shirshak's Rent-an-Equipment Shop
                                                 Address: Maligaun 05, Kathmandu
                                                        Phone: 9544443333
                                               Email: shirshak_rentalshop@gmail.com
--------------------------------------------------------------------------------------------------------------------------------

Invoice ID: rent2023822_1813
Date: 2023-08-22        Time: 18:01:03
-------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Total
-------------------------------------------------------------------------------------
1       1    Velvet Table Cloth   Saathi           $8      2     4            $16
2       2    Microphone Set       Audio Technica   $189    3     7            $1134

Customer name: Shirshak
Customer phone number: 9834827736
-------------------------------------------------------------------------------------
Grand total price: $1150
-------------------------------------------------------------------------------------
```

Figure 32: Rent invoice shown in shell window

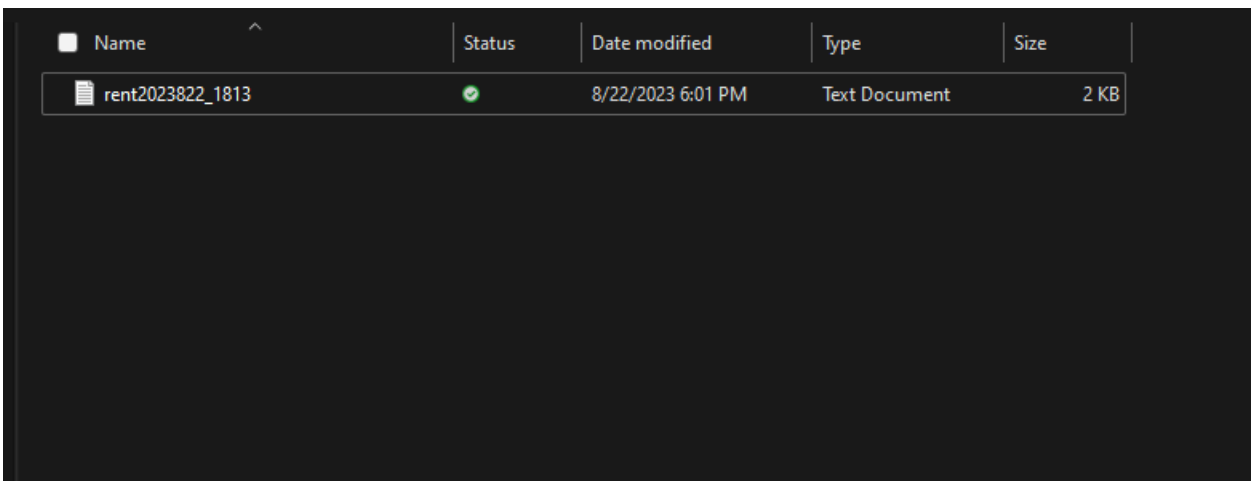
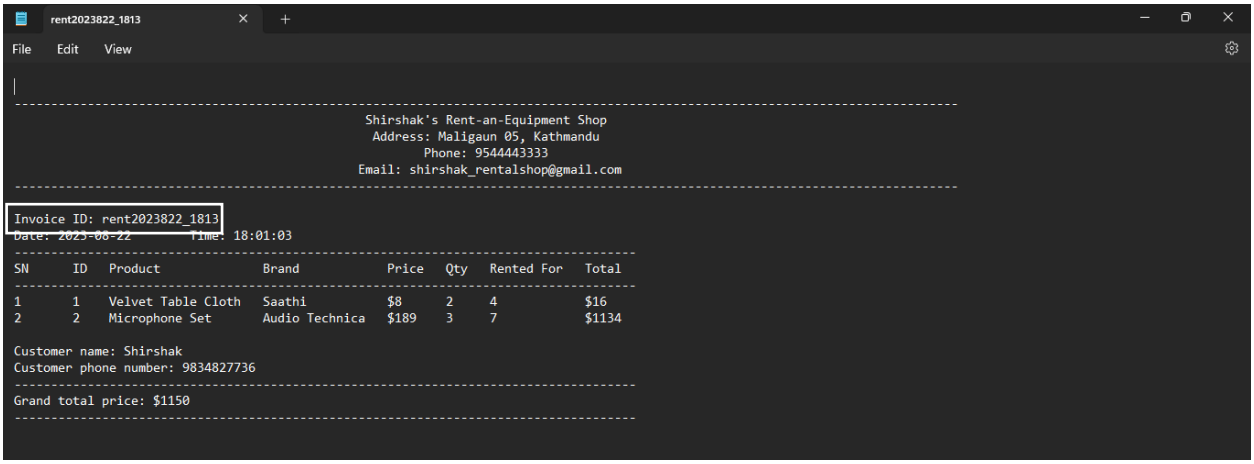


Figure 33: Text file containing the above invoice



```
rent2023822_1813
File   Edit   View

--------------------------------------------------------------------------------------------------------------------------
                                        Shirshak's Rent-an-Equipment Shop
                                         Address: Maligaun 05, Kathmandu
                                                Phone: 9544443333
                                       Email: shirshak_rentalshop@gmail.com
--------------------------------------------------------------------------------------------------------------------------

Invoice ID: rent2023822_1813
Date: 2023-08-22          Time: 18:01:03
--------------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Total
--------------------------------------------------------------------------------------------
1       1    Velvet Table Cloth   Saathi           $8      2     4            $16
2       2    Microphone Set       Audio Technica   $189    3     7            $1134

Customer name: Shirshak
Customer phone number: 9834827736
--------------------------------------------------------------------------------------------
Grand total price: $1150
--------------------------------------------------------------------------------------------
```

Figure 34: The above invoice in the text file

4.4. Test 4: To check the generation of a text file containing the invoice created for returning multiple items

Objective	To check the generation of a text file containing the invoice created for returning multiple items
Action	1. The main module was run. 2. The returning process was carried out.
Expected Result	A text file containing the invoice of the returned items should have been generated.
Actual Result	A text file containing the invoice of the returned items was generated.
Conclusion	The test was successful.

Table 4: Test 4: To check the generation of a text file containing the invoice created for returning multiple items

```
Enter a number: 2
--------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
--------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      100
2       Microphone Set                   Audio Technica     $189    92
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
--------------------------------------------------------------------------
Enter product ID (1 to 5): 2
Enter how many items to return: 3
How many days was this product rented for?: 4
How many days is this product being returned after?: 5
Do you want to return more products? (y/n): y
--------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
--------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      100
2       Microphone Set                   Audio Technica     $189    95
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
--------------------------------------------------------------------------
Enter product ID (1 to 5): 1
Enter how many items to return: 1
How many days was this product rented for?: 3
How many days is this product being returned after?: 6
Do you want to return more products? (y/n): n
```

Figure 35: Returning multiple items

```
--------------------------------------------------------------------------------------------------------------------------
                                                Shirshak's Rent-an-Equipment Shop
                                                Address: Maligaun 05, Kathmandu
                                                        Phone: 9544443333
                                               Email: shirshak_rentalshop@gmail.com
--------------------------------------------------------------------------------------------------------------------------

Invoice ID: return2023823_19838
Date: 2023-08-23        Time: 19:08:38
--------------------------------------------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Returned After   Fine   Total
--------------------------------------------------------------------------------------------------------------------------
1       2    Microphone Set       Audio Technica   $189    3     4            5                $0     $567
2       1    Velvet Table Cloth   Saathi           $8      1     3            6                $1.6   $9.6

Customer name: Shirshak
Customer phone number: 9287366455
--------------------------------------------------------------------------------------------------------------------------
Grand total price: $576.6
--------------------------------------------------------------------------------------------------------------------------
```

Figure 36: Return invoice shown in shell window

Name	Status	Date modified	Type	Size
return2023823_19838		8/23/2023 7:08 PM	Text Document	2 KB

Figure 37: Text file containing the above invoice

```
-----------------------------------------------------------------------------------------------------------------------------
                                                Shirshak's Rent-an-Equipment Shop
                                                 Address: Maligaun 05, Kathmandu
                                                        Phone: 9544443333
                                               Email: shirshak_rentalshop@gmail.com
-----------------------------------------------------------------------------------------------------------------------------

Invoice ID: return2023823_19838
Date: 2023-08-23        Time: 19:08:38
-----------------------------------------------------------------------------------------------------------------------------
SN      ID   Product              Brand            Price   Qty   Rented For   Returned After   Fine   Total
-----------------------------------------------------------------------------------------------------------------------------
1       2    Microphone Set       Audio Technica   $189    3     4            5                $0     $567
2       1    Velvet Table Cloth   Saathi           $8      1     3            6                $1.6   $9.6

Customer name: Shirshak
Customer phone number: 9287366455
-----------------------------------------------------------------------------------------------------------------------------
Grand total price: $576.6
-----------------------------------------------------------------------------------------------------------------------------
```

Figure 38: The above invoice in the text file

4.5. Test 5: To check the updated stock of items after renting and returning

Objective	To check the updated stock of items after renting and returning
Action	1. An item was rented. 2. An item was returned.
Expected Result	The updated stock of the rented/returned items should have been reflected in the stock text file.
Actual Result	The updated stock of the rented/returned items was reflected in the stock text file.
Conclusion	The test was successful.

Table 5: Test 5: To check the updated stock of items after renting and returning

Renting an item:

```
[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
--------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
--------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      90
2       Microphone Set                   Audio Technica     $189    95
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
--------------------------------------------------------------------------
Enter product ID (1 to 5): 1
Enter how many items to rent: 2
How many days is this product being rented for?: 5
Do you want to rent more products? (y/n): n

For generating invoice:
Enter customer name: Shirshak
Enter customer phone number: 9837746655
```

Figure 39: Renting 2 items of product ID 1

Updated stock of the rented item in shell window:

```
[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 2
--------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
--------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    95
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
--------------------------------------------------------------------------
Enter product ID (1 to 5):
```

Figure 40: The updated stock of the rented item reflected in the shell window

Updated stock of the rented item in stock text file

```
stockfile
File   Edit   View

Velvet Table Cloth,Saathi,$8,88
Microphone Set,Audio Technica,$189,95
Disco Light Set,Sonoff,$322,100
7.1 Surround Sound Speaker Set,Dolby,$489,100
Dinner Table 8x5,Panda Furnitures,$344,100
```

Figure 41: The updated stock of the rented item reflected in the stock text file

Returning an item

```
[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 2
-------------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-------------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    95
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-------------------------------------------------------------------------------
Enter product ID (1 to 5): 2
Enter how many items to return: 5
How many days was this product rented for?: 5
How many days is this product being returned after?: 7
Do you want to return more products? (y/n): n

For generating invoice:
Enter customer name: Shirshak
Enter customer phone number: 93847665833
```

Figure 42: Returning 5 items of product ID 2

```
[1] Press 1 to rent an equipment.
[2] Press 2 to return an equipment.
[3] Press 3 to exit the program.

Enter a number: 1
-------------------------------------------------------------------------------
ID      Product                          Brand              Price   Stock
-------------------------------------------------------------------------------
1       Velvet Table Cloth               Saathi             $8      88
2       Microphone Set                   Audio Technica     $189    100
3       Disco Light Set                  Sonoff             $322    100
4       7.1 Surround Sound Speaker Set   Dolby              $489    100
5       Dinner Table 8x5                 Panda Furnitures   $344    100
-------------------------------------------------------------------------------
Enter product ID (1 to 5):
```

Figure 43: Updated stock of the returned item reflected in the shell window

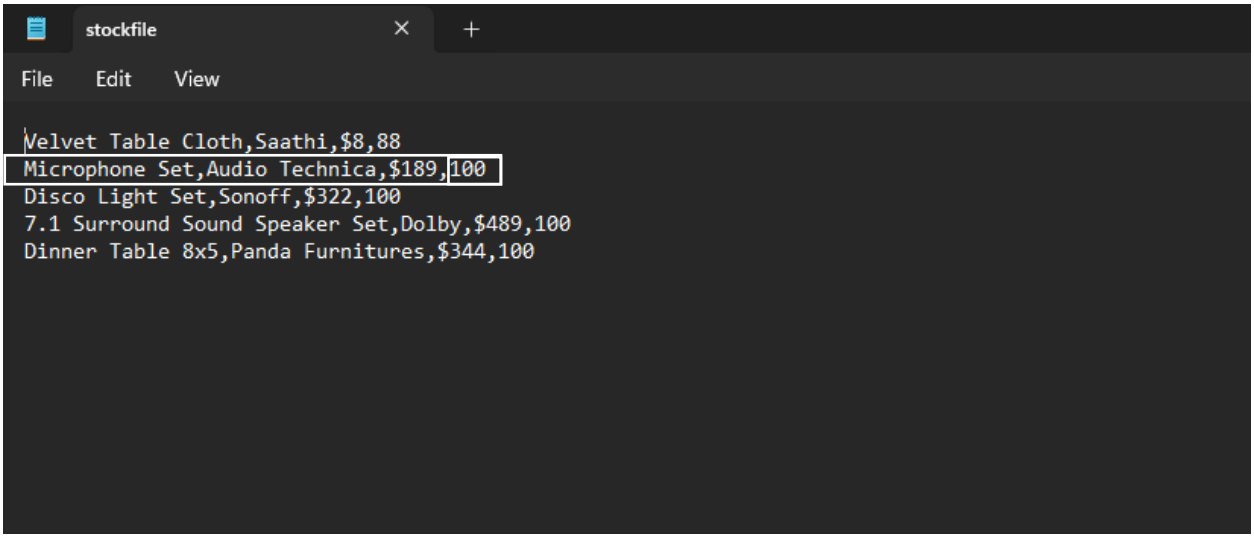


Figure 44: Updated stock of the returned item reflected in the stock text file

5. Conclusion

5.1. Things learnt

The development of this project has, in a broad sense, led to a much higher level of understanding of Python and object-oriented programming itself as well. The ideas of encapsulation and abstraction have proven to be remarkably useful in avoiding unnecessary code duplication and have simplified the maintenance of the code. This has also been replicated in the use of object-oriented programming over procedural programming, specifically in the use of inheritance and polymorphism.

This project, along with a few others, has made me understand the importance of planning before coding, and the consequences that follow in not doing so. Planning the overview of the code greatly helps in simplifying the development and to understand your own code, debug it and maintain it over time.

This project has also introduced me to the concept of file handling, something which I had never heard about before. It has helped me understand it quite a bit and also helped me realize where and why it could be important.

5.2. Limitations of this project and their solutions

5.2.1. Limitations

This project has a few limitations that makes it inconvenient to be used in practical situations. Some examples of such limitations are listed below:

- It is not possible to know when a customer is lying about when they are returning a product.
- It is also difficult to keep track of customers who have rented products before and have not returned them yet.
- Another limitation of this project is the inconvenience of working in a terminal window instead of a GUI.

5.2.2. Solutions

Instead of depending solely on programming, this project can overcome this limitation by implementing these solutions:

- instead implementing a database system. This way, it becomes much easier to keep track of products and customers both and reduces or even removes the need for code entirely.
- A GUI greatly simplifies and enhances user interaction with the program and serves as a means of abstraction.

In a nutshell, this project can be developed further into actual user applications with the right use of technologies and ideas as mentioned above.

5.3. Research and Findings

During the development of this research, I found a few functions and concepts unknown to me beforehand that helped in simplifying some aspects of the code. Some examples are given below:

- The ljust() function justifies the given text according to the number passed as its argument.
- List comprehension is also one of such concepts that was new for me and has reduced potentially several lines of code into a single line.
- The strip() function has also proven to be convenient, helping to remove any leading or trailing whitespaces that could potentially cause some errors if not removed.

In summary, developing this project has led to the understanding of several concepts and their applications, and has also led to the findings of new, convenient functions and ideas. This project would not have been possible at all if not for the lecturers and tutors of our college who are always eager to help students in their work, so I would like to express my gratitude towards them. The internet has also proven once again to be

an incredibly helpful tool for obtaining help in virtually anything, hence I am grateful for that as well.

6. References

6.1. Bibliography

Anon., 2023. *draw.io.* [Online] Available at: https://www.drawio.com/about [Accessed 18 August 2023].

Anon., 2023. *IDLE.* [Online] Available at: https://docs.python.org/3/library/idle.html [Accessed 18 August 2023].

Anon., 2023. *Microsoft.* [Online] Available at: https://www.microsoft.com [Accessed 18 August 2023].

Anon., 2023. *Python.* [Online] Available at: https://www.python.org [Accessed 18 August 2023].

7. Appendix

The codes of each module in this project are shown below:

7.1. Main module

```
from operations import Shop  # importing Shop class for renting and returning functions,
and displaying shop information,
                    # welcome message, stock information and the 3 options
from read import Read  # importing Read class for reading stock file to convert the info
into a dictionary


class Main():  # creating a class Main


    Shop.show_info()  # displaying shop details
    Shop.show_welcome_message()  # displaying welcome message


    prod_dict = Read.file_to_dict()  # storing product info from stock file in a dictionary


    '''this loop is the main loop of the whole program,
    and is exited only when option 3 is selected'''
    exit_program = False  # setting a flag
    while exit_program == False:
        Shop.show_options()  # displaying the rent, return and exit options
        option = Shop.valid_option(prod_dict)  # asking input for the options and validating
it
```

```
    '''this if statement checks input value of option
    if input is 1, it carries out renting operations
    elif input is 2, it carries out returning operations
    else it breaks the while loop and exits the program'''
    if option == 1:  # if user wants to rent a product
        Shop.rent_operations(prod_dict)  # carrying out the rent operations

    elif option == 2:  # if user wants to return a prouct
        Shop.return_operations(prod_dict)  # carrying out the return operations

    else:  # if user wants to exit the program
        exit_program = True  # setting flag to false, breaking the main loop

Shop.show_exit_message()  # printing exit message after user exits the program
```

7.2. Operations module

```
from write import Write  # importing Write class for updating stock file and writing invoices to text files


class Shop():  # creating a class Shop


    @staticmethod
    def rent_operations(prod_dict):
        '''args: product dictionary
        function: carries out renting operations
        returns: None'''

        rent_list = []  # creating a list to store products currently being rented

        '''this while loop carries out renting operations.
        it repeatedly asks user if they want to rent more, until they enter 'n'
        then it asks input for customer details,
        generates invoice,
        and writes the invoice to a new text file'''
        continue_renting = True  # setting a flag
        while continue_renting == True:
            Shop.display_stock(prod_dict)  # displaying the info from the stock file
```

```
        prod_id = Rent.valid_prod_id(prod_dict)  # validating product ID input from user
        qty = Rent.valid_prod_qty(prod_dict, prod_id)  # validating product qty input from user
        rented_for = Rent.valid_rented_for()  # validating num of days rented for input from user

        rent_prod = Rent(prod_dict, prod_id, qty, rented_for)  # creating Rent object
        Product.update_stock(rent_prod,prod_dict)  # updating stock of the object
        rent_list.append(rent_prod)  # adding the product to the rent product list


        '''this if statement checks if no products are available
        if so, it directly breaks this while loop
        else it asks user if they want to rent more
        if yes, it restarts by again asking for product ID
        if not, it breaks this loop'''
        # getting rent choice from user
        prod_not_available = Shop.are_no_prods_available(prod_dict)  # setting a flag to check if products are available for rent
        if prod_not_available == True:  # if no products are available for renting
            continue_renting = False  # setting flag to false, breaking this while loop
        else:  # if products are available for renting
            rent_choice = Rent.get_rent_choice()  # validating input from user if they want to rent more
```

```
            if rent_choice == True:  # if user wants to rent more
                continue_renting = True  # setting flag to true, continuing this while loop
            else:  # if user does not want to rent more
                continue_renting = False  # setting flag to false, breaking this while loop

        # this section is related to invoice generation only
        Bill.print_invoice_message()  # printing message to inform user to enter customer details
        cust_name = Shop.get_cust_name()  # taking customer name input from user
        cust_num = Shop.get_cust_num()  # taking customer phone number input from user

        rent_bill = Rent.create_bill(rent_list,cust_name,cust_num)  # creating an invoice for rented products
        print(rent_bill)  # printing the invoice
        Write.rent_bill_to_file(rent_bill)  # writing the invoice to a new text file

    @staticmethod
    def return_operations(prod_dict):
        '''args: product dictionary
        function: carries out returning operations
        returns: None'''

        return_list = []  # creating a list to store products currently being returned
```

```
'''this while loop carries out returning operations.
it repeatedly asks user if they want to return more, until they enter 'n'
then it asks input for customer details,
generates invoice,
and writes the invoice to a new text file'''
continue_returning = True  # setting a flag
while continue_returning == True:
    Shop.display_stock(prod_dict)  # displaying the info from the stock file


    prod_id = Return.valid_prod_id(prod_dict)  # validating product ID input from user
    qty = Return.valid_prod_qty()  # validating product qty input from user
    days_rented = Return.valid_days_rented()  # validating num of days rented for input from user
    returned_after = Return.valid_returned_after()  # validating num of days returned after input from user



    return_prod = Return(prod_dict, prod_id, qty, days_rented, returned_after)  # creating product object
    Product.update_stock(return_prod, prod_dict)  # # updating stock of the product object
    return_list.append(return_prod)  # adding the product to the return product list
```

```
        '''this if statement asks user if they want to rent more
        if yes, it restarts by again asking for product ID
        if not, it breaks this loop'''
        return_choice = Return.get_return_choice()  # validating input from user if they want to return more
        if return_choice == True:  # if user wants to rent more
            continue_returning = True  # setting flag to true, continuing this while loop
        else:  # if user does not want to rent more
            continue_returning = False  # setting flag to false, breaking this while loop


    # this section is related to invoice generation only
    Bill.print_invoice_message()  # printing message to inform user to enter customer details
    cust_name = Shop.get_cust_name()  # taking customer name input from user
    cust_num = Shop.get_cust_num()  # taking customer phone number input from user


    return_bill = Return.create_bill(return_list,cust_name,cust_num)  # creating an invoice for returned products
    print(return_bill)  # printing the invoice
    Write.return_bill_to_file(return_bill)  # writing the invoice to a new text file
```

```
    @staticmethod
    def show_info():
        '''args: None
        function: prints the shop details
        returns: None'''

        shop_name = "Shirshak's Rent-an-Equipment Shop"  # setting shop name
        shop_address = "Address: Maligaun 05, Kathmandu"  # setting shop address
        shop_number = "Phone: 9544443333"  # setting shop contact num
        shop_email = "Email: shirshak_rentalshop@gmail.com"  # setting shop email address

        shop_info = []  # creating a list to store shop details
        shop_info.append(shop_name)  # adding shop name to the list
        shop_info.append(shop_address)  # adding shop addrss to the list
        shop_info.append(shop_number)  # adding shop contact number to the list
        shop_info.append(shop_email)  # adding shop email address to the list

        num_of_equals_sign = 129  # setting num of equals sign to be printed
        equals_sign = Misc.print_equals_sign(num_of_equals_sign)  # setting equals sign
```

```
        centering_num = 129  # setting the num for centering the text


        # this for loop prints the shop information at the center of the screen
        print(equals_sign)  # printing equals sign
        for each_info in shop_info:
            print(each_info.center(centering_num))  # printing shop details
        print(equals_sign)  # printing equals sign


    @staticmethod
    def show_welcome_message():
        '''args: None
        function: prints a welcome message
        returns: None'''


        # setting a welcome message
        welcome_msg = "Welcome to Shirshak's Rent-an-Equipment Shop! Please see the
instructions below to carry out your desired tasks!"


        num_of_dashes = 122  # setting num of dashes to be printed
        dashes = Misc.print_dashes(num_of_dashes)  # setting dashes


        print(dashes)  # printing dashes
        print(welcome_msg)  # printing welcome message
```

```
        print(dashes + "\n")  # printing dashes


    @staticmethod
    def show_options():
        '''args: None
        function: displays options to rent, return or exit
        returns: None'''

        opt1 = "[1] Press 1 to rent an equipment."  # setting message of rent option
        opt2 = "[2] Press 2 to return an equipment."  # setting message of return option
        opt3 = "[3] Press 3 to exit the program.\n"  # setting message of exit option

        option_list = []  # creating a list to store the 3 options
        option_list.append(opt1)  # adding option 1 to the list
        option_list.append(opt2)  # adding option 2 to the list
        option_list.append(opt3)  # adding option 3 to the list

        # displaying each option in the list
        for option in option_list:
            print(option)

    @staticmethod
    def display_stock(prod_dict):
```

```
    '''args: product dictionary
    functions: displays the info in the dictionary
    returns: None'''

    # creating a list that contains the list info of each product
    # to align its columns
    stock_list = [[item for item in each] for each in prod_dict.values()]
    header_row = ["Product","Brand","Price","Stock"]  # creating a header row
    stock_list.insert(0,header_row)  # adding header row to the list
    aligned_stock_list = Misc.align_2D_list(stock_list)  # aligning the list

    num_of_dashes = Misc.get_num_of_dashes(aligned_stock_list)  # setting num of dashes to be printed
    dashes = Misc.print_dashes(num_of_dashes)  # setting dashes

    #this section is for printing the header row
    print(dashes)  # printing dashes
    print("ID\t" + "".join(aligned_stock_list[0]))  # printing the header row
    print(dashes)  # printing dashes

    # this for loop prints the product details from the list
    p_id = 1  # setting product ID
    for i in range(1,len(aligned_stock_list)):
```

```
            print(str(p_id) + "\t" + "".join(aligned_stock_list[i]))  # printing the product details
            p_id += 1  # increasing product ID by 1 each time

        print(dashes)  # printing dashes

    @staticmethod
    def show_exit_message():
        '''args: None
        function: prints an exit message after user exits main loop
        returns: None'''

        # printing the exit message
        print("\nThank you for using our application.")

    @staticmethod
    def valid_option(prod_dict):
        '''args: product dictionary
        function: validates user input for rent, return or exit
        returns: valid option number'''

        '''this while loop asks user input for rent, return or exit options
        and repeatedly asks for valid input unless user enters valid input
        here, invalid input being anything other than 1, 2 or 3
```

```
        it also displays an error message if no product is available for rent at all'''
        while True:
            try:
                option = int(input("Enter a number: ").replace(" ","")) # asking user to input a number
                if option == 1: # if user enters 1
                    no_prods_available = Shop.are_no_prods_available(prod_dict)
                    if no_prods_available == True: # if no product is available for rent
                        print("Sorry, no equipment is available for rent right now!\n") # printing error message
                    else: # if products are available for rent
                        return option # returning option

                elif option == 2 or option == 3: # if user enters 2 or 3
                    return option # returning option

                else: # if user enters anything other than 1, 2 or 3
                    print("Invalid option; please input either 1, 2 or 3\n") # printing an error message
            except ValueError:
                print("Invalid input; please enter the option in numbers, either 1, 2 or 3\n")

    @staticmethod
```

```
    def are_no_prods_available(prod_dict):
        '''args: product dictionary
        function: checks if no products are available for renting
        returns: True if no products are available, else False'''

        all_stock_sum = 0  # setting thesum to 0 to be added to

        # calculating the sum of stock nums of each product from the dictionary
        for each in prod_dict.values():
            all_stock_sum += int(each[-1])

        if all_stock_sum == 0:  # if no products are available
            return True  # returning True
        else:  # if products are available
            return False  # returning false

    @staticmethod
    def get_cust_name():
        '''args: None
        function: asks input for customer name
        returns: customer name'''

        '''this while loop asks input for customer name
```

```
        and repeatedly asks for valid input until user enters valid input
        here, invalid input means empty text (whitespaces do not count)'''
        while True:
            cust_name = input("Enter customer name: ").strip()  # asking user for customer name
            if len(cust_name) == 0:  # if user inputs empty string
                print("Invalid input; please enter the customer name\n")  # printing error message
            else:  # if user input is valid
                return cust_name  # returning customer name

    @staticmethod
    def get_cust_num():
        '''args: None
        function: asks input for customer phone number
        returns: customer phone number'''

        '''this while loop asks input for customer phone number
        and repeatedly asks for valid input until user enters valid input
        here, invalid input means empty text (whitespaces do not count)'''

        while True:
```

```
        cust_num = input("Enter customer phone number: ").strip()  # asking user for customer phone number
        if len(cust_num) == 0:  # if user inputs empty string
            print("Invalid input; please enter the customer's phone number\n")  # printing error message
        else:  # if user input is valid
            return cust_num  # returning customer phone number


    @staticmethod
    def get_details_for_bill():
        '''args: None
        function: sets the text containing shop details, to be printed in invoices
        returns: shop details'''


        num_of_dashes = 122  # setting num of dashes
        dashes = Misc.print_dashes(num_of_dashes)  # getting dashes


        centering_num = 129
        # setting shop details to be printed in invoices
        details_for_bill = "\n" + dashes + "\n"  # dashes
        details_for_bill += "Shirshak's Rent-an-Equipment Shop".center(centering_num) + "\n"  # shop name
        details_for_bill += "Address: Maligaun 05, Kathmandu".center(centering_num) + "\n"  # shop address
```

```
        details_for_bill  +=  "Phone:  9544443333".center(centering_num)  +  "\n"    #  shop contact num

        details_for_bill += "Email: shirshak_rentalshop@gmail.com".center(centering_num) + "\n"  # shop email address

        details_for_bill += dashes + "\n"  # dashes


        return details_for_bill  # returning details for bill


class Product():  # creating a class Product


    def __init__(self, prod_dict, prod_id, qty):

        '''args: product dictionary, product ID, product qty

        function: creates a Product object

        returns: None'''


        # using values from product dictionary

        self.ID = prod_id  # setting product ID
```

```
        self.name = prod_dict[prod_id][0]  # setting product name
        self.brand = prod_dict[prod_id][1]  # setting brand name
        self.price = prod_dict[prod_id][2] # setting product price
        self.stock = prod_dict[prod_id][3]  # setting product stock
        self.qty = qty  # setting product qty

    @staticmethod
    def get_grand_total(prod_list):
        '''args: list of products currently being rented/returned
        function: calculates grand total price
        returns: grand total price'''

        grand_total = 0  # setting a variable to store grand total price
        for each in prod_list:
            grand_total += each.total_price  # calculating grand total price
        return grand_total  # returning grand total price

    @staticmethod
    def update_stock(prod, prod_dict):
        '''args: rented/returned product, product dictionary
        function: updates stock of rented/returned product in the product dictionary, and writes the dictionary to stock file
        returns: None'''
```

```
        current_stock = int(prod.stock)  # setting current stock of product
        qty = prod.qty  # setting product qty
        updated_stock = 0  # setting updated product stock to be modified below

        '''this if statement calculates the updated stock
        if the product is an object of Rent class, it deducts the rented quantity from current stock
        else it adds the returned quantity to current stock'''
        if isinstance(prod,Rent):  # if product is rented
            updated_stock = current_stock - qty  # subtracting rented qty from current stock
        else:  # if product is returned
            updated_stock = current_stock + qty  # adding returned qty to current stock

        prod_dict[prod.ID][-1] = str(updated_stock)  # updating stock value in product dictionary
        Write.dict_to_file(prod_dict)  # writing updated stock to stock file


class Rent(Product):  # creating a class Rent that inherits Product class

    def __init__(self, prod_dict, prod_id, qty, rented_for):
        '''args: product dictionary, product ID, product qty, num of days rented for
```

```
        function: creates a Rent object by calling superclass init method
        returns: None'''

        # calling superclass init method
        super().__init__(prod_dict, prod_id, qty)

        self.rented_for = rented_for  # setting num of days product is being rented for

        price = prod_dict[prod_id][2]  # getting price from product dictiionary
        price_factor = Rent.get_price_factor(rented_for)  # setting price factor
        self.total_price = Rent.get_total_price(price, price_factor, qty)  # setting total price

    @staticmethod
    def get_price_factor(rented_for):
        '''args: num of days rented for
        function: calculates the price factor of the product
        returns: price factor of the product'''

        '''this if-else statement sets price factor of product
        if rented_for is divisible by 5, it sets price factor as rented_for / 5
        else it sets price factor as (rented_for // 5) + 1'''

        price_factor = 0 # setting a variable to store price factor
```

```
        if rented_for % 5 == 0:  # if rented_for is divisible by 5
            price_factor = rented_for/5
        else:  # if rented_for is not divisible by 5
            price_factor = (rented_for // 5) + 1

        return price_factor  # returning price factor

    @staticmethod
    def get_total_price(price, price_factor, qty):
        '''args: price, price factor
        function: calculates the total price the product
        returns: total price of the product'''

        # calculating total price
        total_price = price_factor * int(price.replace("$","")) * qty
        return total_price  # returning total price

    @staticmethod
    def valid_prod_id(prod_dict):
        '''args: product dictionary
        function: validates input for product ID
        returns: valid product ID'''
```

```
    '''this loop asks input for product ID
    and displays error messages if input is invalid
    here, invalid input being non-integer values and values not within 1 to 5
    it also displays error message if the product with input ID is unavailable'''
    while True:
        try:
            prod_id = int(input("Enter product ID (1 to 5): ").replace(" ","")) # asking input for product ID

            if 1 <= prod_id <= len(prod_dict): # checking if input is between 1 to the length of product dictionary

                if int(prod_dict[prod_id][-1]) == 0: # checking if the product with that ID is not available
                    print("Sorry, this product is not available right now!\n") # printing error message
                else: # if the product with that ID is available
                    return prod_id # returning product ID

            else: # if input is not between 1 to 5
                print("Invalid product ID; please enter a number from 1 to 5\n") # printing error message
```

```
        except ValueError:  # if input is a non-integer value
            print("Invalid input; please enter the product ID in numbers\n")  # printing error message


    @staticmethod
    def valid_prod_qty(prod_dict, prod_id):
        '''args: product dictionary, product ID
        function: validates input for product qty
        returns: valid product qty'''


        '''this loop asks input for product qty
        and displays error messages if input is invalid
        here, invalid input being non-integer values and values not within 1 and the available stock'''
        while True:
            try:
                qty = int(input("Enter how many items to rent: ").replace(" ","")) # asking input for product qty
                if 0 < qty <= int(prod_dict[prod_id][-1]):  # checking if input qty is available
                    return qty  # returning product qty
                else:  # if input qty is not available
                    print("Invalid input; please enter a number within 1 and the available stock\n") # printing error message
            except ValueError:  # if input is a non-integer value
```

```
            print("Invalid input; please enter the quantity in numbers\n")  # printing error message


    @staticmethod
    def valid_rented_for():
        '''args: None
        function: validates input for num of days rented for
        returns: valid num of days rented for'''

        '''this loop asks input for num of days rented for
        if the input is more than 0, it returns that value
        else it prints error message
        if input is not an integer, it prints error message'''
        while True:
            try:
                # asking input for num of days rented for
                rented_for = int(input("How many days is this product being rented for?: ").replace(" ",""))

                if rented_for > 0:  # if input is a positive number
                    return rented_for  # returning num of days rented for
                else:  # if input is 0 or a negative value
```

```
                print("Invalid input; please enter a positive value for the number of days\n") # printing error message


        except:  # if input is a non-integer value
            print("Invalid input; please enter the number of days in numbers\n")  # printing error message


    @staticmethod
    def get_rent_choice():
        '''args: None
        function: asks input for if user wants to rent more products
        returns: rent choice'''

        '''this loop asks input for if user wants to rent more products
        if input is 'y', it returns True
        elif input is 'n', it returns False
        else it prints error message'''
        while True:
            # asking if user wants to rent more products
            rent_more = input("Do you want to rent more products? (y/n): ").lower().replace(" ","")


            if rent_more == "y":  # if user wants to rent more products
```

```
            return True  # returning True
        elif rent_more == "n":  # if user does not want to rent more products
            return False  # returning False
        else:  # if user enters anything but "y" or "n"
            print("Invalid answer; please enter either 'y' or 'n'\n")  # printing error message

    @staticmethod
    def create_bill(rent_list, cust_name, cust_num):
        '''args: list of currently rented products, customer name, customer phone number
        function: creates a rent invoice
        returns: rent invoice'''

        bill_text = Shop.get_details_for_bill()  # getting shop details to be printed in invoice
        bill_text += "\nInvoice ID: " + Rent.get_bill_id() + "\n"  # setting invoice ID
        bill_text += Bill.create_bill(rent_list, cust_name, cust_num) + "\n"  # creating rent invoice
        return bill_text  # returning rent invoice

    @staticmethod
    def get_bill_id():
        '''args: None
        function: sets unique invoice ID
        returns: unique invoice ID'''
```

```
        bill_id = "rent" + Write.get_unique_id()  # setting ID for rent invoice
        return bill_id  # returning the invoice ID


class Return(Product):  # creating a class Return that inherits Product class

    def __init__(self, prod_dict, prod_id, qty, days_rented, returned_after):
        '''args: product dictionary, product ID, product qty, num of days returned after
        function: creates a Return object by calling superclass init method
        returns: None'''

        # calling superclass init method
        super().__init__(prod_dict, prod_id, qty)

        self.days_rented = days_rented  # setting num of days rented for
        self.returned_after = returned_after  # setting num of days returned after

        price = prod_dict[prod_id][2]  # getting price from product dictionary

        price_factor_for_days_rented = Rent.get_price_factor(days_rented)  # getting price
factor for days_rented
```

```
        base_price = Rent.get_total_price(price,price_factor_for_days_rented,qty)  # getting base price of the product


        self.fine = Return.get_fine(price, days_rented, returned_after, qty)  # setting fine

        self.total_price = base_price + self.fine  # setting total price


    @staticmethod

    def get_fine(price, days_rented, returned_after, qty):

        '''args: price, days rented, days returned after

        function: calculates the fine

        returns: the fine'''


        # getting price factor for days rented

        price_factor_for_days_rented = Rent.get_price_factor(days_rented)

        # getting price factor for days returned after

        price_factor_for_returned_after = Rent.get_price_factor(returned_after)


        '''this if statement checks if price factor for days rented and days returned after is same

        if they are same, it sets the fine to 0

        else it sets the fine as the product of the extra days and price for 5 days'''

        fine = 0  # setting a variable to store the fine
```

```
        if price_factor_for_days_rented >= price_factor_for_returned_after:  # if price factor is same
            fine = 0  # setting fine to 0
        else:  # if price factor is different
            fine_days = returned_after - (price_factor_for_days_rented * 5)  # calculating num of days to be fined for
            fine = (int(price.replace("$",""))/5) * fine_days * qty  # calculating fine

        return round(fine,2)  # returning fine

    @staticmethod
    def valid_prod_id(prod_dict):
        '''args: None
        function: validates input for product ID
        returns: valid product ID'''

        '''this loop asks input for product ID
        and displays error messages if input is invalid
        here, invalid input being non-integer values and values not within 1 to 5'''
        while True:
            try:
                prod_id = int(input("Enter product ID (1 to 5): ").replace(" ","")) # asking input for product ID
```

```
            if 1 <= prod_id <= len(prod_dict):  # checking if input is between 1 to 5
                return prod_id  # returning product ID
            else:  # if input is not between 1 to 5
                print("Invalid product ID; please enter a number from 1 to 5\n")  # printing error message
        except ValueError:  # if input is a non-integer value
            print("Invalid input; please enter the product ID in numbers\n")  # printing error message


    @staticmethod
    def valid_prod_qty():
        '''args: None
        function: validates input for product qty
        returns: valid product qty'''

        '''this loop asks input for product qty
        and displays error messages if input is invalid
        here, invalid input being 0, and negative and non-integer values'''
        while True:
            try:
                qty = int(input("Enter how many items to return: ").replace(" ","'"))  # asking input for product qty
                if qty > 0:  # checking if input is a positive value
```

```
                return qty  # returning product qty
            else:  # if input is 0 or negative
                print("Invalid input; please enter a number greater than 0\n")  # printing error message
        except ValueError:  # if input is a non-integer value
            print("Invalid input; please enter the quantity in numbers\n")  # printing error message


    @staticmethod
    def valid_days_rented():
        '''args: None
        function: validates the num of days rented
        returns: valid num of days rented'''


        '''this while loop asks input for num of days rented
        if input is more than 0, it returns that value
        else it prints error message
        if input is not an integer, it prints error message'''
        while True:
            try:
                # asking input for num of days rented
                days_rented = int(input("How many days was this product rented for?: ").replace(" ",""))
```

```
            if days_rented > 0:  # if input is more than 0
                return days_rented  # returning the value
            else:  # if input is 0 or less
                print("Invalid input; please enter a number greater than 0\n")  # printing error message
        except:  # if input is not an integer
            print("Invalid input; please enter the number of days in numbers\n")  # printing error message

    @staticmethod
    def valid_returned_after():
        '''args: None
        function: validates input for num of days returned after
        returns: valid num of days returned after'''

        '''this loop asks input for num of days returned after
        and displays error messages if input is invalid
        here, invalid input being negative and non-integer values'''
        while True:
            try:
                # asking input for num of days returned after
```

```
            returned_after = int(input("How many days is this product being returned after?: ").replace(" ",""))


            if returned_after > 0:  # if input is a positive value
                return returned_after  # returning num of days returned after
            else:  # if input is 0 or a negative value
                print("Invalid input; please enter a non-negative value\n")  # printing error message


        except:  # if input is a non-integer value
            print("Invalid input; please enter the number of days in numbers\n")  # printing error message


    @staticmethod
    def get_return_choice():
        '''args: None
        function: asks input for if user wants to return more products
        returns: return choice'''


        '''this while loop asks input for if user wants to return more producs
        and repeatedly asks for valid input until user enters valid input
        here, invalid input being any text other than 'y' or 'n' '''
        while True:
```

```
        # asking if user wants to return more products
        return_more = input("Do you want to return more products? (y/n): ").lower().replace(" ","")
        if return_more == "y":  # if user wants to return more products
            return True  # returning True
        elif return_more == "n":  # if user does not want to return more products
            return False  # returning False
        else:  # if the user enters anything but "y" or "n"
            print("Invalid answer; please enter either 'y' or 'n'\n")  # printing error message

    @staticmethod
    def create_bill(return_list, cust_name, cust_num):
        '''args: list of currently returned products, customer name, customer phone number
        function: creates a return invoice
        returns: return invoice'''

        bill_text = Shop.get_details_for_bill()  # getting shop details to be printed in invoice
        bill_text += "\nInvoice ID: " + Return.get_bill_id() + "\n"  # setting invoice ID
        bill_text += Bill.create_bill(return_list, cust_name, cust_num) + "\n"  # creating return invoice
        return bill_text  # returning return invoice

    @staticmethod
```

```
    def get_bill_id():
        '''args: None
        function: sets unique invoice ID
        returns: unique invoice ID'''

        bill_id = "return" + Write.get_unique_id()  # setting ID for return invoice
        return bill_id  # returning the invoice ID



class Bill():  # creating a class Bill

    @staticmethod
    def get_aligned_bill_list(prod_list):
        '''args: list of currently rented/returned products
        function: creates a list of aligned items to be printed in the invoice
        returns: the list of aligned items'''

        bill_list = []  # creating a list to store the product details from the arg list

        #this loop adds a list of product details of each product in the arg list
        for each in prod_list:
            if isinstance(each,Rent):  # if the list is of rented products
                bill_list.append([
```

```
            str(each.ID),  # product ID
            each.name,  # product name
            each.brand,  # brand name
            each.price,  # product price
            str(each.qty),  # product qty
            str(each.rented_for),  # num of days product is being rented for
            "$" + str(each.total_price)  # total price
        ])
    else:  # if the list is of returned products
        bill_list.append([
            str(each.ID),  # product ID
            each.name,  # product name
            each.brand,  # brand name
            each.price,  # product price
            str(each.qty),  # product qty
            str(each.days_rented),  # num of days rented for
            str(each.returned_after),  # num of days returned after
            "$" + str(each.fine),  # fine
            "$" + str(each.total_price)  # total price
        ])

header_row = []  # setting a variable to store header row
if isinstance(prod_list[0],Rent):  # if the list is of rented products
```

```
        header_row = Bill.get_rent_bill_header_row()  # setting the rent invoice header row
    else:  # if the list is of returned products
        header_row = Bill.get_return_bill_header_row()  # setting the return invoice header row

    bill_list.insert(0,header_row)  # inserting the header row of the invoice to the bill list
    aligned_bill_list = Misc.align_2D_list(bill_list)  # aligning the bill list

    return aligned_bill_list  # returning the aligned bill list

@staticmethod
def get_bill_prod_details(aligned_bill_list):
    '''args: list of invoice items
    function: sets each row of product details with a serial number
    returns: each row of product details with a serial number'''

    bill_prod_details = ""  # setting the variable as an empty string to add to
    sn = 1  # initializing sn as the serial number

    # this loop sets each row of product details with a serial number
    for i in range(1,len(aligned_bill_list)):
        bill_prod_details += str(sn) + "\t" + "".join(aligned_bill_list[i]) + "\n"
```

```
            sn += 1  # increasing sn by 1 each time


        return bill_prod_details  # returning the modified rows of product details


    @staticmethod
    def get_bill_cust_details(cust_name,cust_num):
        '''args: customer name, customer phone number
        function: prints the customer name and phone number
        returns: the customer name and phone number'''


        bill_cust_details = "\nCustomer name: " + cust_name # printing customer name
        bill_cust_details += "\nCustomer phone number: " + cust_num + "\n"  # printing customer phone number
        return bill_cust_details  # returning the customer details


    @staticmethod
    def get_bill_grand_total(grand_total):
        '''args: grand total price
        function: sets the grand total price as a string with the '$' sign
        returns: the grand total price as a string with the '$' sign'''


        # setting the grand total price as a string with the '$' sign
        bill_grand_total = "Grand total price: $" + str(round(grand_total,2)) + "\n"
```

```
        return bill_grand_total  # returning the grand total price as a string with the '$' sign


    @staticmethod
    def bill_header_row(aligned_bill_list):
        '''args: list of invoice items
        function: sets the header row of the invoice with a serial number column header
        returns: the header row of the invoice with a serial number column header'''

        # setting the header row of the invoice with a serial number column header
        bill_header = "SN\t" + "".join(aligned_bill_list[0]) + "\n"
        return bill_header  # returning the header row

    @staticmethod
    def get_rent_bill_header_row():
        '''args: None
        function: sets the header row for rent invoice
        returns: the header row for rent invoice'''

        # setting header row for rent invoice
        header_row = ["ID","Product","Brand","Price","Qty","Rented For","Total"]
        return header_row  # returning header row

    @staticmethod
```

```
    def get_return_bill_header_row():
        '''args: None
        function: sets the header row for return invoice
        returns: the header row for return invoice'''

        # setting header row for return invoice
        header_row = ["ID","Product","Brand","Price","Qty","Rented For", "Returned After","Fine","Total"]
        return header_row  # returning header row

    @staticmethod
    def print_invoice_message():
        '''args: None
        function: prints a message to inform user to enter customer details
        returns: None'''

        print("\nFor generating invoice: ")  # printing the invoice message

    @staticmethod
    def create_bill(prod_list, cust_name, cust_num):
        '''args: list of currently rented/returned products, customer name, customer phone number
        function: creates an invoice
```

```
    returns: the invoice'''

    grand_total = Product.get_grand_total(prod_list)  # getting grand total price in string form

    aligned_bill_list = Bill.get_aligned_bill_list(prod_list)  # aligning the items to be printed in the invoice

    num_of_dashes = Misc.get_num_of_dashes(aligned_bill_list)  # setting num of dashes to be printed

    bill_text = Write.get_current_date()  # getting current date
    bill_text += Write.get_current_time()  # getting current time

    bill_text += Misc.print_dashes(num_of_dashes) + "\n"  # printing dashes
    bill_text += Bill.bill_header_row(aligned_bill_list)  # printing header row of invoice
    bill_text += Misc.print_dashes(num_of_dashes) + "\n"  # printing dashes

    bill_text += Bill.get_bill_prod_details(aligned_bill_list)  # printing product details in the invoice
    bill_text += Bill.get_bill_cust_details(cust_name,cust_num)  # printing customer name and phone number

    bill_text += Misc.print_dashes(num_of_dashes) + "\n"  # printing dashes
```

```
        bill_text += Bill.get_bill_grand_total(grand_total)  # printing grand total price
        bill_text += Misc.print_dashes(num_of_dashes) + "\n"  # printing dashes

        return bill_text  # returning the invoice


class Misc():  # creating a class Misc

    @staticmethod
    def align_2D_list(list_2D):
        '''args: a 2D list
        function: aligns the items inside every list in the 2D list
        returns: the aligned form of the arg 2D list'''

        #calculating the number of spaces to left-justify each column with in the 2D list
        space_l = [max(len(each[i]) for each in list_2D) for i in range(len(list_2D[0]))]
```

```
        padding = 3  # setting the padding between each column

        # aligning the items inside each list in the 2D list
        for each in list_2D:
            for i in range(len(list_2D[0])):
                each[i] = each[i].ljust(space_l[i] + padding)

        return list_2D  # returning the aligned 2D list

    @staticmethod
    def get_num_of_dashes(list_2D):
        '''args: a 2D list
        function: calculates the num of dashes to print for decorating a 2D list
        returns: the num of dashes'''

        # creating a list to store the total lengths of items for each list in the 2D list
        len_of_each_row = []

        # adding those lengths to the list above
        for i in range(len(list_2D)):
            len_of_each_row.append(sum(len(item) for item in list_2D[i]))

        # setting the maximum length from the list above as the num of dashes
```

```
        num_of_dashes = max(len_of_each_row)

        return num_of_dashes  # returning the number of dashes

    @staticmethod
    def print_dashes(num_of_dashes):
        '''args: num of dashes
        function: prints that many dashes
        returns: that many dashes'''

        surplus = 7  # setting the surplus number of dashes when necessary
        dashes = "-" * (num_of_dashes + surplus)  # printing the required number of dashes
        return dashes  # returning the required amount of dashes

    @staticmethod
    def print_equals_sign(num_of_sign):
        '''args: num of equals signs
        function: prints that many equals signs above and below the shop details
        returns: that many equals signs'''

        surplus = 0  # setting the surplus number of equals signs when necessary
        equals = "=" * (num_of_sign + surplus)  # printing the required number of equals signs
```

```
    return equals  # returning the required amount of equals signs
```

7.3. Read module

```
class Read():  # creating a class Read


    @staticmethod
    def file_to_dict():
        '''args: None
        function: reads the stock file and returns the info in a dictionary
        returns: a dictionary with the info in the stock file'''

        stock_file = open("stockfile.txt", "r")  # opening the stock file
        data = stock_file.read()  # reading the stock file
        data = data.split("\n")  # splitting the data per each new line

        prod_dict = {}  # creating a dictionary to store the data
        c = 1  # setting the counter for the dictionary keys

        # this loop adds the list of details of each product to the dictionary
        for i in range(len(data)):
            prod_dict[c] = data[i].split(",")
            c += 1  # increasing the counter by 1 each time

        stock_file.close()  # closing the stock file
```

```
    return prod_dict  # returning prod_dict
```

7.4. Write module

```
import datetime  # importing datetime library for generating unique file names for invoices


class Write():  # creating a class Write


    @staticmethod
    def dict_to_file(prod_dict):
        '''args: product dictionary
        function: updates the stock file with the info in the dictionary
        returns: None'''

        updated_info = ""  # setting an empty string to store the updated product info

        # this loop adds each list of the updated product details to updated_info
        for value in prod_dict.values():
            updated_info += ",".join(value) + "\n"

        updated_info = updated_info.rstrip("\n")  # removing the trailing newline from the last line

        stock_file = open("stockfile.txt","w")  # opening the stock file in write mode
        stock_file.write(updated_info)  # writing the updated product information to the
        stock_file.close()  # closing the stock file
```

```
    @staticmethod
    def rent_bill_to_file(bill_text):
        '''args: rent invoice
        function: writes the rent invoice to a new text file
        returns: None'''

        file_name = Write.get_rent_invoice_file_name()  # setting the file name

        new_rent_file = open(file_name,"w")  # creating a new file with the unique name
        new_rent_file.write(bill_text)  # writing the bill to the file
        new_rent_file.close()  # closing the file

    @staticmethod
    def return_bill_to_file(bill_text):
        '''args: return invoice
        function: writes the return invoice to a new text file
        returns: None'''

        file_name = Write.get_return_invoice_file_name()  # setting the file name

        new_rent_file = open(file_name,"w")  # creating a new file with the unique name
        new_rent_file.write(bill_text)  # writing the bill to the file
```

```
        new_rent_file.close()  # closing the file


    @staticmethod
    def get_unique_id():
        '''args: None
        function: creates a unique ID using the current date and time
        returns: the unique ID'''


        year = str(datetime.datetime.now().year)  # storing the current year in a variable
        month = str(datetime.datetime.now().month)  # storing the current month in a variable
        day = str(datetime.datetime.now().day)  # storing the current day in a variable


        hour = str(datetime.datetime.now().hour)  # storing the current hour in a variable
        minute = str(datetime.datetime.now().minute)  # storing the current minute in a variable
        second = str(datetime.datetime.now().second)  # storing the current second in a variable


        unique_id = year + month + day + "_" + hour + minute + second  # setting the unique ID


        return unique_id  # returning the unique ID
```

```
    @staticmethod
    def get_rent_invoice_file_name():
        '''args: None
        function: creates a unique file name for the text file for rent invoices
        returns: the unique file name'''

        file_id = Write.get_unique_id()  # setting a unique ID for the file name
        unique_file_name = "Rent Invoices/rent" + file_id + ".txt"  # setting the file name
        return unique_file_name  # returning the unique file name

    @staticmethod
    def get_return_invoice_file_name():
        '''args: None
        function: creates a unique file name for the text file for return invoices
        returns: the unique file name'''

        file_id = Write.get_unique_id()  # setting a unique ID for the file name
        unique_file_name = "Return Invoices/return" + file_id + ".txt"  # setting the file name
        return unique_file_name  # returning the unique file name

    @staticmethod
    def get_current_date():
        '''args: None
```

```
        function: gets the current date
        returns: the current date'''

        # getting the current date
        current_date = "Date: " + str(datetime.datetime.now().date()) + "\t"
        return current_date  # returning the current date

    @staticmethod
    def get_current_time():
        '''args: None
        function: gets the current time
        returns: the current time'''

        # getting the current time
        current_time = "Time: " + str(datetime.datetime.now().time().replace(microsecond=0)) + "\n"
        return current_time  # returning the current time
```