C++ Final Project Documentation

Ramen Shop Summary App

Eric Liu

345822795

ICS4U1 - 03

Dr. Noukhovitch

April 4, 2021

INTRODUCTION

Design Overview:

Ramen Shop Summary App is a convenient calculation/short database program designed for a small ramen shop to calculate its earnings from a set of orders. Users including the cashier and manager can input data for up to 100 orders made during the day; they are also allowed to delete any orders that were entered mistakenly or incorrectly. The user also has the option to display all orders, and can also use the list to check which order they wish to delete.

The base class Ramen contains the templates(member variables) that can be used by its subclasses. It also includes all the functions that would be used by the subclasses and int main() for calculations or manipulations. The subclasses Tonkotsu and Shoyu, derived from the Ramen base class, depicts two different types of ramen sold by the shop, each containing its unique attributes and cost.

Each order is added in the perspective of an object pointer within the object pointers array of type Ramen. According to the type of ramen sold to the customer, the attributes of the ramen and cost will be set up when the object pointer directs the user to the matching constructor in one of the subclasses. Each of the constructor in the subclass will initially allow the user to input customers' general information by directing them to a member function of the base class.

The user can enter an integer to select an option from the options menu. The input will be validated by a function in the base class, which throws an exception if the input is non-numeric, and will display an error message. In addition, when the ramen attributes setup requires character type input, the invalid message will also be displayed when the user inputs an integer instead or inputs an invalid character option.

When the user chooses to quit the program, the entire object pointers array will have its memory deallocated. The total earning made from the orders and a thank you message will be displayed.

Requirement Traceability Matrix:

	Main	Base Class - Ramen	Subclass - Tonkotsu	Subclass - Shoyu	Constr uctor	Add Customer Bill	Remove Customer Bill	Display Customer Bill	New Keyword	Delete Keyword	Quit
Set-up	x										
Create object pointers array	x										
Initialize object pointer address	x								x		
Set attributes for objects created on the heap			x	x	x						
constructor			x	x							

Input validation	x	x				x	x	x			
Error Output (exceptions handling)	x	x				x	x	x			
Remove Allocated Memory	x									x	
Reset object pointer to nullptr							x				
End of program											x

SYSTEM ARCHITECTURAL DESIGN

Chose Program Architecture: Layered Pattern

Main program:

- Contains the object pointers array that has its object pointers initialized to nullptr
- Uses a while loop that would keep iterating until the user chooses to quit the program through the list of commands. The sentinel, running, is initialized to true and will be reassigned to false when the user chooses to quit
- User will select an option from the options menu
    - Input validation and exception handling are performed when the user inputs an option
- If statements are used within the while loop to guide the user to the correct code that will perform the chosen function/option
    - Input 1: “Add Customer Bill” - allows the user to initialize the order

        - If the “Add Customer Bill” option is selected, the user will be asked to select the type of Ramen sold. Input validation and exception handling will also be performed
            - This will reassign the address of object pointer of type Ramen dynamically to the matching subclass address using the new keyword
            - As the object pointer is reassigned on the heap, the matching subclass constructor is called to initialize the order’s attributes and costs, also adds the current bill to the total earning

    - Input 2: “Remove Customer Bill” - allows the user to remove an order inputted incorrectly or accidentally
        - If this option is selected, the user will be asked to select an order number to delete. Input validation and exceptions handling will be performed promptly
            - The program will reassign the matching object pointer to nullptr based on the user input
            - If the inputted order number is out of range(not between 1 and 100), a message will be displayed to ask for re-input

    - Input 3: “Display Customer Bill” - displays all orders available
        - Outputs all non-null object pointer data
        - If all object pointers are nullptr, then there are no available orders to display
    - Input 4: “Quit” - quits the program and displays the total earning
        - Outputs the total earning from all available bills
        - Displays a goodbye message

Base Class Ramen

- Contains all functions that will be used by the subclass functions and constructors and the main program.
    - inputInfo(): enables the user to input basic information of the customer
    - charValidity(): checks the character input validity
    - intValidity(): checks the integer input validity
    - displayOrder(): the virtual function that directs the object pointer to the matching overridden function
    - additionalToppings(): displays the toppings added
    - isNumeric(): exception handling for non-numeric inputs
- Contains the private member variables
    - i.e. name and contactNum
- Contains public member variables that can be accessed by main and subclass functions for manipulations and outputs

Subclasses

- The subclasses are derived from the base class Ramen
- Each class has its own constructor that initializes the order's attributes and costs.
- There are two subclasses:
    - Tonkotsu
    - Shoyu
        - The two subclasses depicts the two famous types of ramen
- Each of the subclasses has its own overridden function that enables the object pointer to use the matching overridden displayOrder function and display the corresponding attributes and costs of the order. The object pointer will be directed to the correct overridden function based on its address that was assigned dynamically.

Layered Pattern

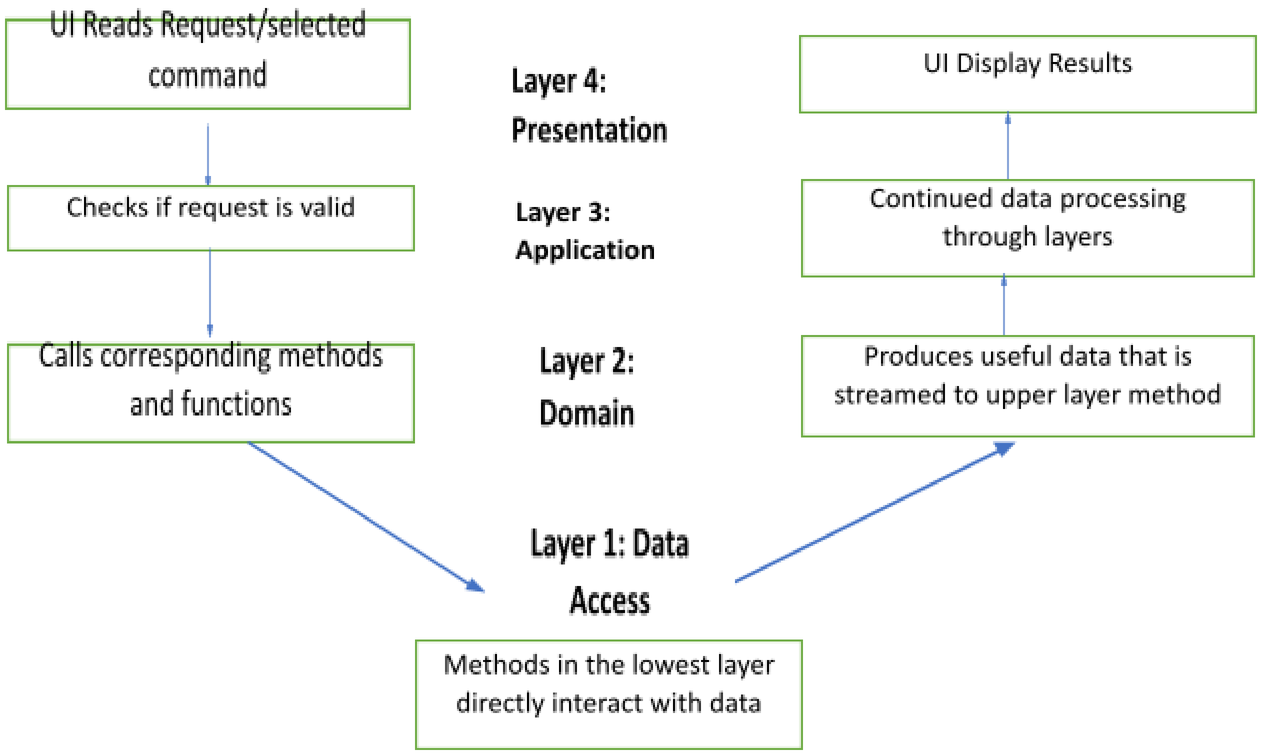
- The layered pattern is a specific program architecture in which the program is able to be divided into subtasks
    - Layer 4: The presentation layer is represented by the options list, cout statements, and particular instructions that enables the user to correctly select the prefered commands through the user interface. This layer also displays the final results after program execution.

    - Layer 3: The application layer is built on the basis of the previous layer. It ensures the valid inputs are made by the user so that the next layer can be triggered. It displays helpful messages through input validations and exceptions handling to help the user make valid inputs.

    - Layer 2: The business logic layer directs the user to the correct option based on user input, which appends the user to the matching function and constructor to perform the desired operation. This layer will only occur upon the occurrence of layer 3 and 4.

    - Layer 1: The data access layer provides detailed instructions on operations that will access and manipulate the database, such as storing names and information, deleting the orders, and setting attributes of orders. Successful modification during the data access layer provides the desired output to the screen. This layer must only occur when all layers beforehand are handled correctly.

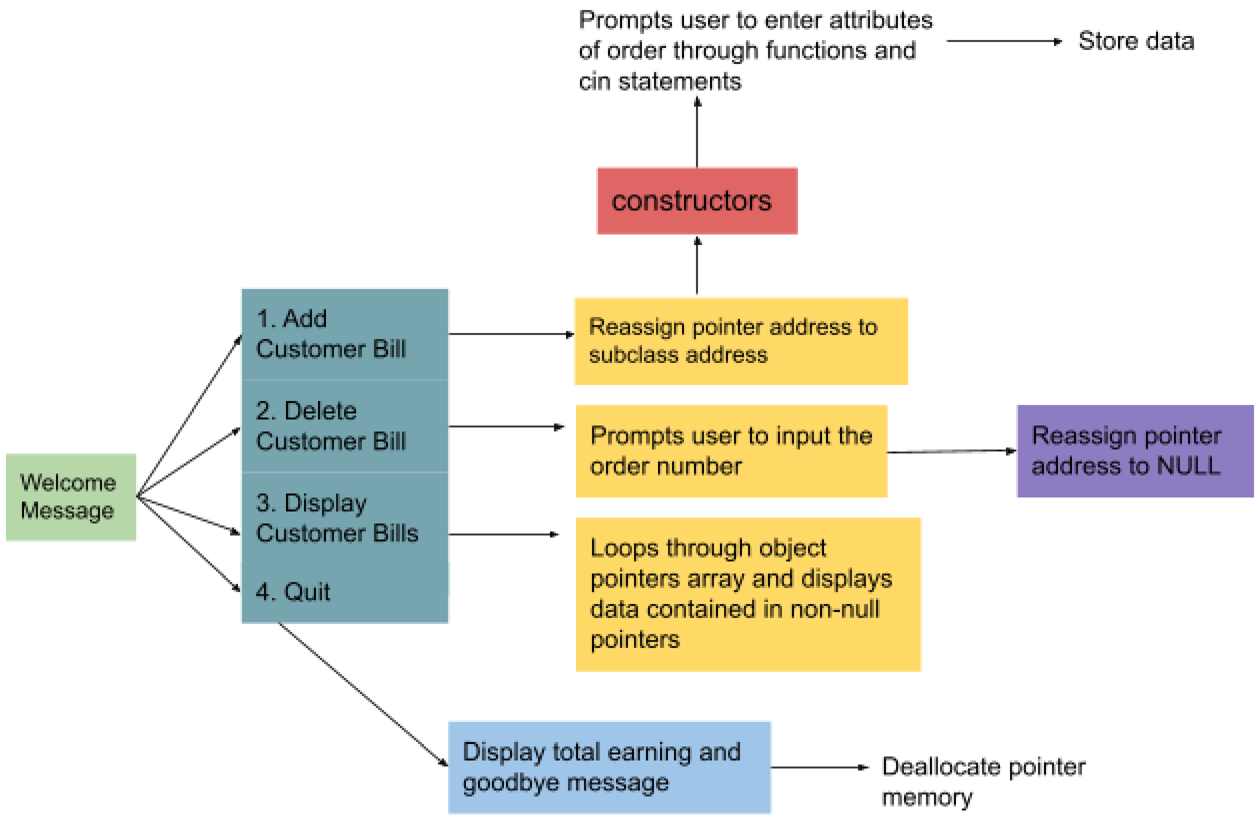
    After all execution and process are completed, the data is carried back to the uppermost layer and is presented to the user.


UI Reads Request/selected command
Layer 4: Presentation
UI Display Results
Checks if request is valid
Layer 3: Application
Continued data processing through layers
Calls corresponding methods and functions
Layer 2: Domain
Produces useful data that is streamed to upper layer method
Layer 1: Data Access
Methods in the lowest layer directly interact with data

Alternative Designs

There are two alternative designs in which I pondered initially. My initial conception was to use a vector array that stores object pointers as elements. Using vectors could be more efficient than my code right now, because it avoids the need to initialize the object pointers array to nullptr in which I did in my current code. With the use of vectors, the vector array can be expanded by one object pointer each time the user wishes to add a customer bill to the list, and can shrink when the user desires to delete; there is no set size for the array. With my code however, I set the maximum number of orders to 100, limiting the number of inputs the user can make (although the program is designed to summarize a small number of orders) — it is not as flexible as if I used the vectors array. Now to the cons of vectors array, it would inevitably cost me more lines of code as I tried to keep mine as short as possible. I would have to introduce the #include <vectors> header and use many additional functions that can potentially take up more lines. When I considered the delete function, I perceived that it was easier if I set my object pointers to nullptr; when I display all available orders, I can easily skip over those that have their address pointing to NULL to correctly output the available orders. My other idea was to read from a text file. This, however, is a time consuming process as the user also needs to type information into several text files. The program will then perform its operation by reading through the files, writing over the files, saving them, and output from the files — this process is not very efficient. Similar goal can be achieved instead by simply allowing the user to input information based on an user interface and directly reference the information from the program itself.

System Interface Design



Prompts user to enter attributes of order through functions and cin statements
Store data
constructors
Welcome Message
1. Add Customer Bill
2. Delete Customer Bill
3. Display Customer Bills
4. Quit
Reassign pointer address to subclass address
Prompts user to input the order number
Reassign pointer address to NULL
Loops through object pointers array and displays data contained in non-null pointers
Display total earning and goodbye message
Deallocate pointer memory

DETAIL DESCRIPTION OF COMPONENTS

Classes

- Base class Ramen contains all templates (member variables) that are used by its subclasses. It also contains all member functions that are used by the subclasses and the main for data processing and exception handling

- Subclass Tonkotsu contains its own class constructor that initializes the attributes of the order when the object pointer is being dynamically assigned to the address of this subclass using the new keyword.
  - The Tonkotsu subclass contains its own overridden function displayOrder() that correctly prints order information for an object pointer pointing to the address of this subclass.
  - Inherits all public member variables and functions from its base class Ramen

- Subclass Shoyu also contains its own class constructor that initializes the attributes of the order when the object pointer is being dynamically assigned to the address of subclass Shoyu using the new keyword.
  - The Shoyu subclass also contains its own overridden function displayOrder() that correctly prints order information for an object pointer pointing to the address of subclass Shoyu.
  - Inherits all public member variables and functions from its base class Ramen

Object Pointers Array

- **orders** (declared as Ramen *orders[100]) of type Ramen is a base class object pointers array that holds 100 object pointers.
  - Each of the object pointer was initially assigned with nullptr so that the pointer addresses will not be assigned with random values

    - The address of each object pointer can be easily reassigned during the program execution and point to its subclasses to perform upcasting
    - Upcasting will be performed in the main using the new keyword along with the subclass name with parenthesis, which invokes the corresponding subclass constructor to begin attribute initialization

Constructors

- Each subclass contains its own constructor
    - Each constructor accesses the base class function for adding orders and input validation checks
    - When the constructors are invoked, they initialize the base class public member variables for each new object pointer assigned dynamically(in other words, initialize the attributes of each order)

Base Class Member Functions

- Input validation functions
    - charValidity - tests validity of char type inputs
    - intValidity - tests integer type inputs, specifically used by the subclass constructor for inputs limited to either 1 or 0

- General information input function:
    - inputInfo - specifically used by the subclass constructors to input basic informations about the order, such as name of the customer and contact number

- Functions that displays data
    - virtual function displayOrder - used by the main to output order information for each object pointer
        - Overridden functions in both subclasses will be invoked according to the origins (address assigned dynamically) of the object pointers

    - displayInfo - included as a part of the displayOrder function, specifically used to output basic information of the customer
    - additionalToppings - specifically used by the displayOrder in each subclass to output if there is an additional topping added to each order and the name of the topping if applicable

- Exception handling function
    - isNumeric - used by both the main and the subclass constructors to check if the input is numeric when it is supposed to be

Subclass Member Functions

- Overridden displayOrder functions

Statements

- if and else statements
    - Used in the main to direct execution to the section of code containing the user selected function
    - Used in input validation loops within the main
    - Used in subclass constructors and overridden functions
    - Acts as the sentinel

Loops

- While loops
    - Main while loop iterates until the user chooses to quit
    - Additional while loops in the main for input validations
    - Used in charValidity which keeps prompting user for input until the correct character type input is made
    - Used in intValidity which keeps prompting user for input until the correct integer type input is made

- For loops
    - Used to loop through all object pointers and output the information contained in the non-null pointers
    - Used for initialization of object pointers to nullptr in main
    - Used for deallocation of the object pointers in main
- Try and catch blocks
    - Used in isNumeric function to catch runtime exceptions caused by non-numeric input from the user
    - Used in the main to catch exceptions caused by inputs that are out of the available range for orders

Variables

Base class

<u>Basic information variables (inputted through the base class function):</u>

- String name: stores the name of the customer of a particular order
- String contactNum: stores the phone number of the customer for a particular order

<u>Template variables (used by all subclasses and values are directly inputted using constructors):</u>

- Double basePrice: each constructor will initialize the this variable with a different value, it is the base cost of the order without toppings
- Double bill: stores the cost of a particular order
- Static double totalEarning: stores the total earning from all available orders, can be directly accessed by main for output

Subclass Tonkotsu

- Char style: the Tonkotsu ramen has two styles — character 'O' represents the original style (pure pork broth) and character 'R' represents Red (spicy)
- Int tamago: the additional topping for this type of ramen(a half boiled egg). If user inputs 1, then extra cost will be added. If not, the bill will remain at base cost.

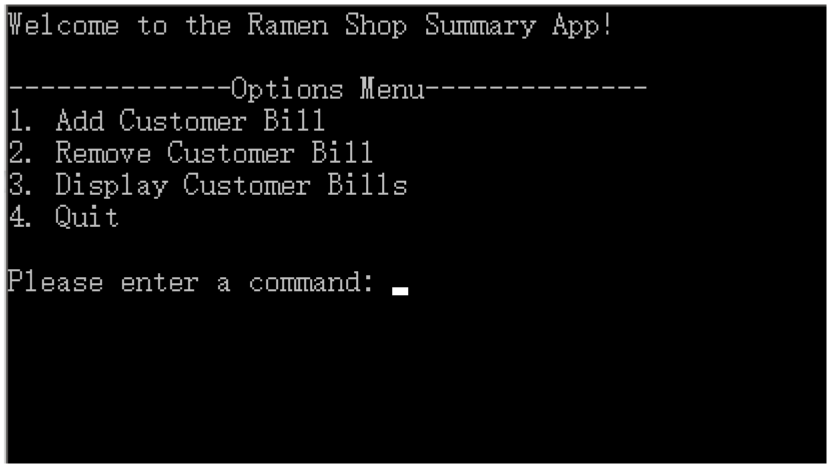
Subclass Shoyu

- Char baseBroth: the Shoyu ramen has two different bases for broth — character ‘C’ represents chicken broth, character ‘P’ represents pork broth.
- Int karaage: the additional topping for this type of ramen(fried chicken). If user inputs 1, then extra cost will be added. If not, the bill will remain at base cost.

Main

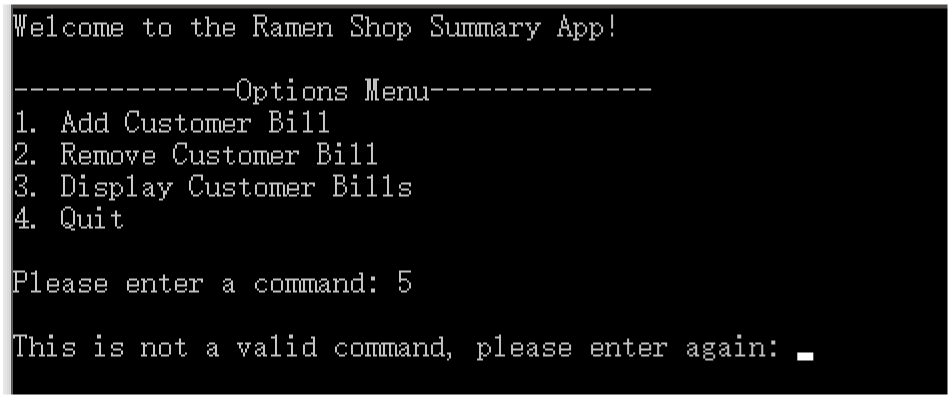
- Int Counter: counts the number of orders created
- Int firstDeletion: when the first order is deleted(index 0 of the pointers array) initially, the index for next order 1’s deletion is shifted 1 to the right
- Char ramenType: this enables the user to input which type of Ramen to begin upcasting
- Bool running: the while loop uses this variable as a sentinel and keeps iterating until running is evaluated to false from the quit option
- Int option: stores the input for command from the user

User Interface Design



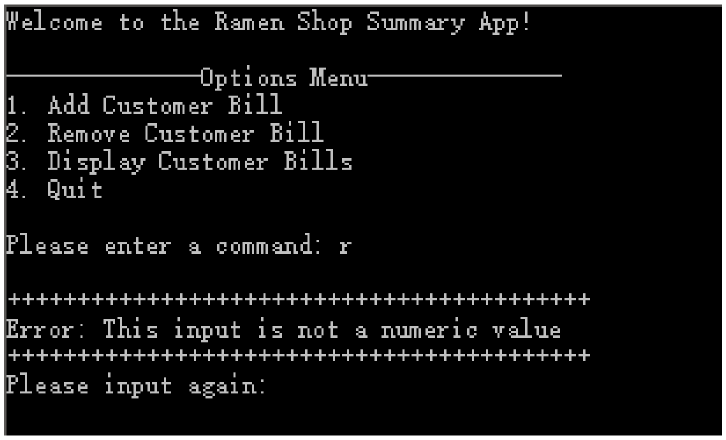
```
Welcome to the Ramen Shop Summary App!

--------------Options Menu--------------
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 
```

Figure 1: Start up displays a welcome message and prompts the user to select a command



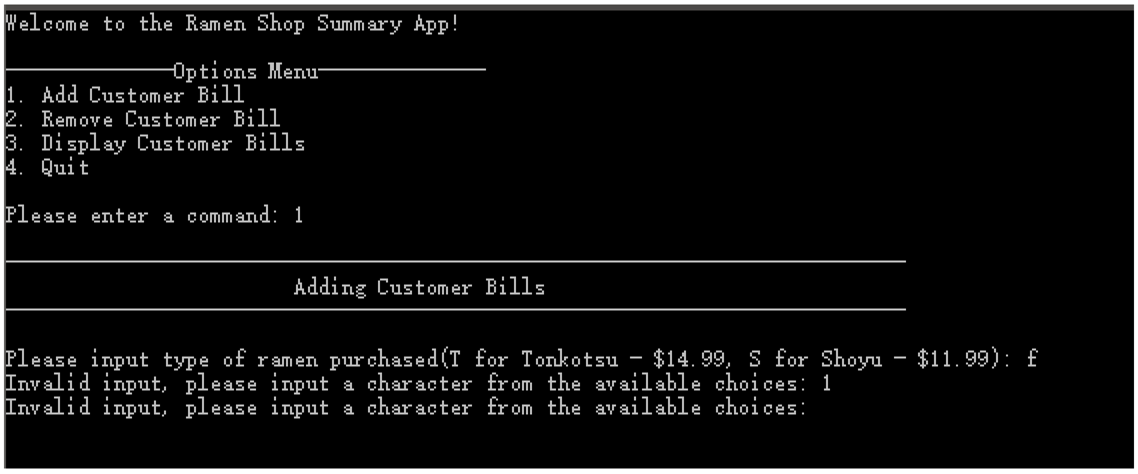
```
Welcome to the Ramen Shop Summary App!

--------------Options Menu--------------
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 5

This is not a valid command, please enter again:
```

Figure 2: command selection is invalid. A validation error message is displayed and asks for the user to input again. The program will keep prompting the user for input until a command in the range of 1 to 4 is entered.



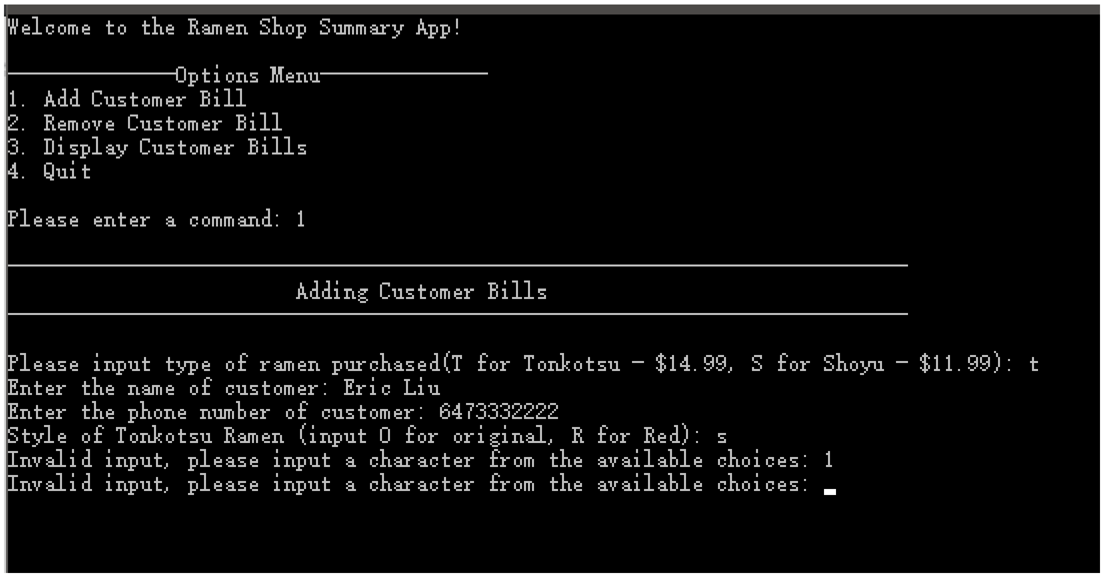
```
Welcome to the Ramen Shop Summary App!

--------------Options Menu--------------
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: r

++++++++++++++++++++++++++++++++++++++++
Error: This input is not a numeric value
++++++++++++++++++++++++++++++++++++++++
Please input again:
```

Figure 3: When the user input for command is a non-numeric value, the exception will be handled and displays the error message. The program will prompt the user to input until the command is numeric and in the correct range.



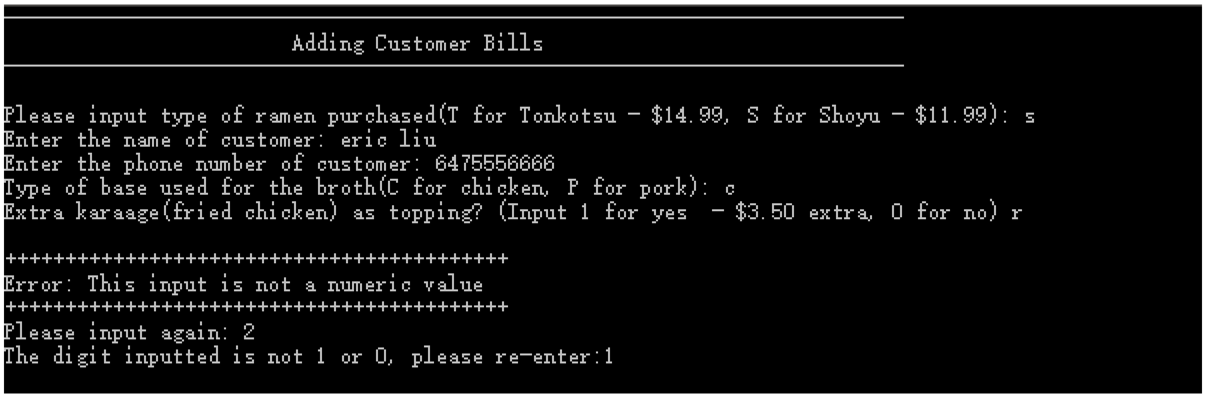
```
Welcome to the Ramen Shop Summary App!

──────────────Options Menu─────────────
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 1

───────────────────────────────────────────────────────────────────────────
                        Adding Customer Bills
───────────────────────────────────────────────────────────────────────────

Please input type of ramen purchased(T for Tonkotsu - $14.99, S for Shoyu - $11.99): f
Invalid input, please input a character from the available choices: 1
Invalid input, please input a character from the available choices:
```

Figure 4: When the user selects the first option, the program will prompt the user to choose the type of ramen sold for a particular order. Input validation will be performed promptly and will keep asking the user to re-enter until the user inputs a choice from the available types (T or S, smaller cased also works)



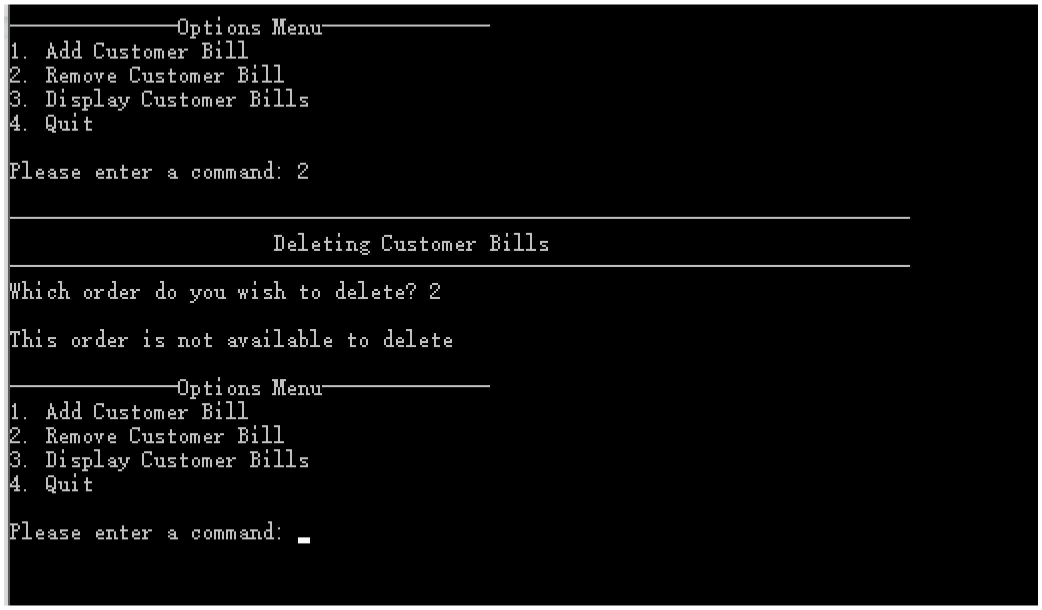
```
Welcome to the Ramen Shop Summary App!

──────────────Options Menu─────────────
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 1

───────────────────────────────────────────────────────────────────────────
                        Adding Customer Bills
───────────────────────────────────────────────────────────────────────────

Please input type of ramen purchased(T for Tonkotsu - $14.99, S for Shoyu - $11.99): t
Enter the name of customer: Eric Liu
Enter the phone number of customer: 6473332222
Style of Tonkotsu Ramen (input O for original, R for Red): s
Invalid input, please input a character from the available choices: 1
Invalid input, please input a character from the available choices:
```

Figure 5: When the user chooses an available choice of Ramen, the program prompts the user to input the customer's general information. For input of particular attributes of each type of ramen, input validation is again performed to prompt the user for re-input until a valid choice is selected.



```
                        Adding Customer Bills

Please input type of ramen purchased(T for Tonkotsu - $14.99, S for Shoyu - $11.99): s
Enter the name of customer: eric liu
Enter the phone number of customer: 6475556666
Type of base used for the broth(C for chicken, P for pork): c
Extra karaage(fried chicken) as topping? (Input 1 for yes  - $3.50 extra, 0 for no) r

++++++++++++++++++++++++++++++++++++++++++
Error: This input is not a numeric value
++++++++++++++++++++++++++++++++++++++++++
Please input again: 2
The digit inputted is not 1 or 0, please re-enter:1
```

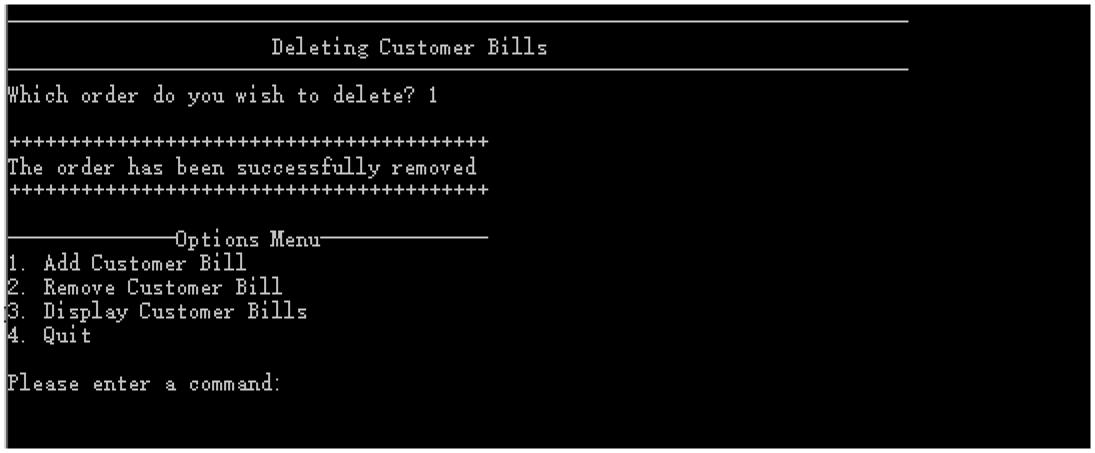
Figure 6: When adding additional toppings, input validation and exception handling are performed and prompts the user to re-enter until a valid input is made.



```
--------------Options Menu--------------
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 2

                      Deleting Customer Bills

Which order do you wish to delete? 2

This order is not available to delete

--------------Options Menu--------------
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: _
```

Figure 7: Note - there is currently one order stored in the program.

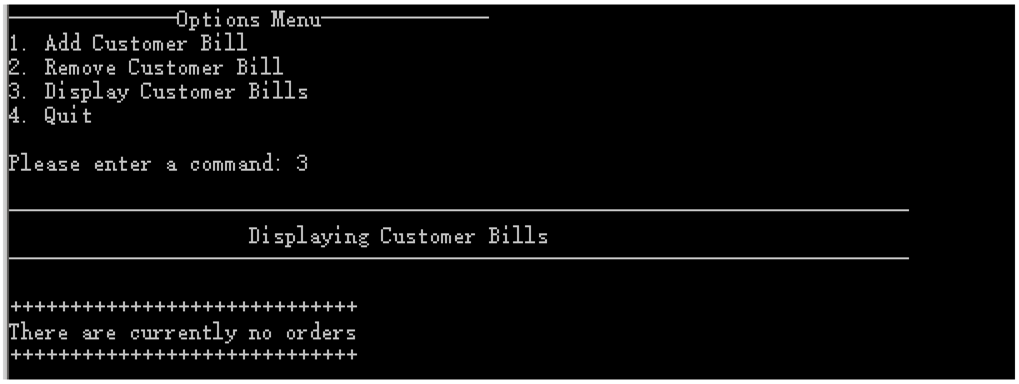
When the user chooses the second option and attempts to delete an unavailable order, the program displays the “This order is not available to delete” message.



```
──────────────────────────────────────────────────────────────
                     Deleting Customer Bills
──────────────────────────────────────────────────────────────
Which order do you wish to delete? 1

+++++++++++++++++++++++++++++++++++++++
The order has been successfully removed
+++++++++++++++++++++++++++++++++++++++

─────────────Options Menu─────────────
1.  Add Customer Bill
2.  Remove Customer Bill
3.  Display Customer Bills
4.  Quit

Please enter a command:
```

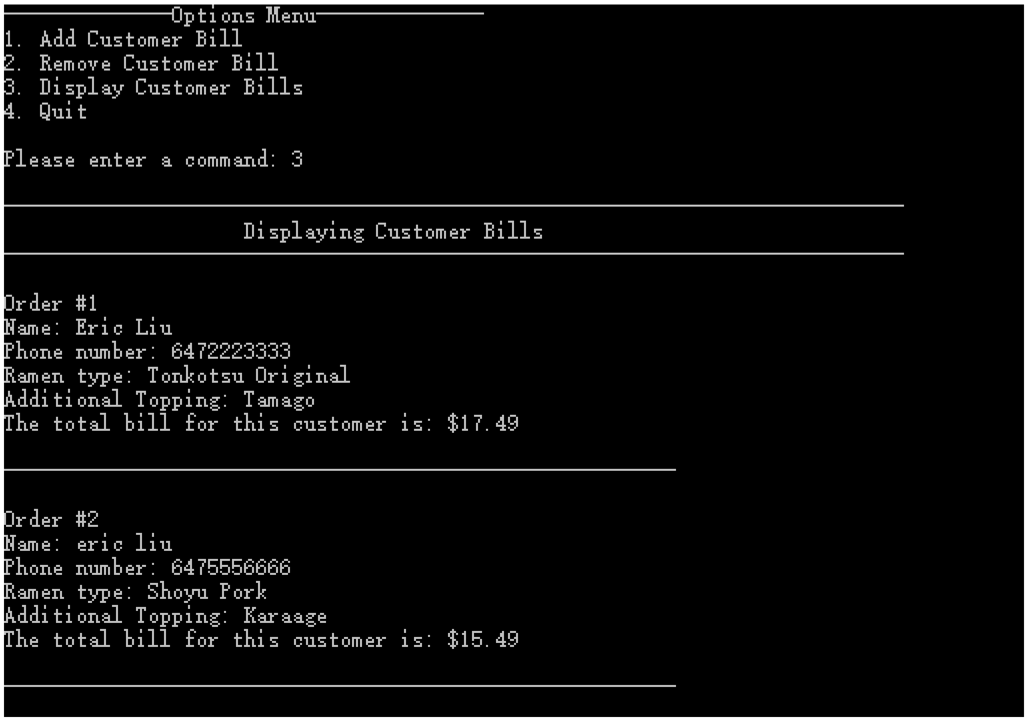
Figure 8: According to the same condition as last figure (currently only order 1 is in the record).

When the order is successfully removed, the successful message is displayed.



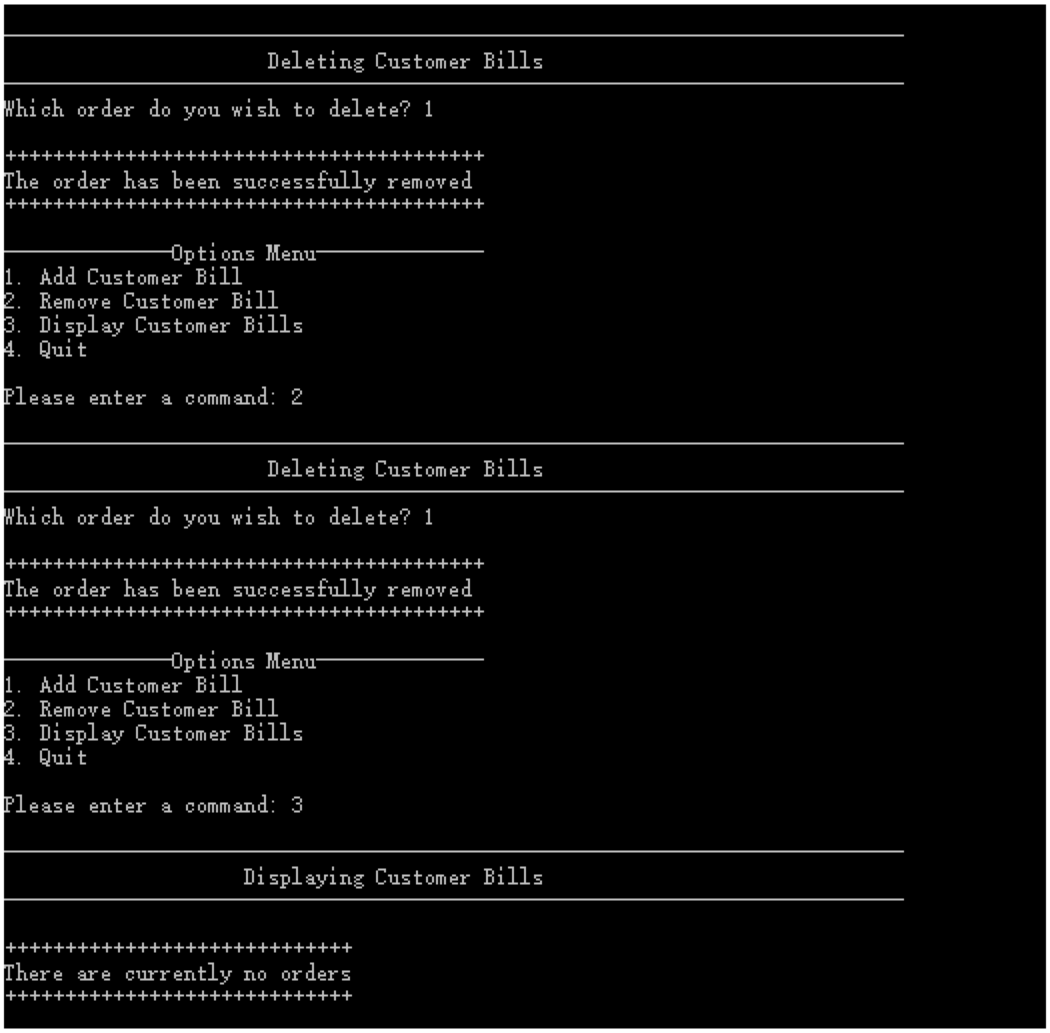
```
─────────────Options Menu─────────────
1.  Add Customer Bill
2.  Remove Customer Bill
3.  Display Customer Bills
4.  Quit

Please enter a command: 3

──────────────────────────────────────────────────────────────
                    Displaying Customer Bills
──────────────────────────────────────────────────────────────

++++++++++++++++++++++++++++
There are currently no orders
++++++++++++++++++++++++++++
```

Figure 9: When the user selects option 3 to display all available orders (currently there is no order stored in the program), the “There are currently no orders” message is displayed.



```
──────────────Options Menu──────────────
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 3

────────────────────────────────────────────────────────────────────────────
                    Displaying Customer Bills
────────────────────────────────────────────────────────────────────────────


Order #1
Name: Eric Liu
Phone number: 6472223333
Ramen type: Tonkotsu Original
Additional Topping: Tamago
The total bill for this customer is: $17.49

─────────────────────────────────────────────────────────


Order #2
Name: eric liu
Phone number: 6475556666
Ramen type: Shoyu Pork
Additional Topping: Karaage
The total bill for this customer is: $15.49

─────────────────────────────────────────────────────────
```

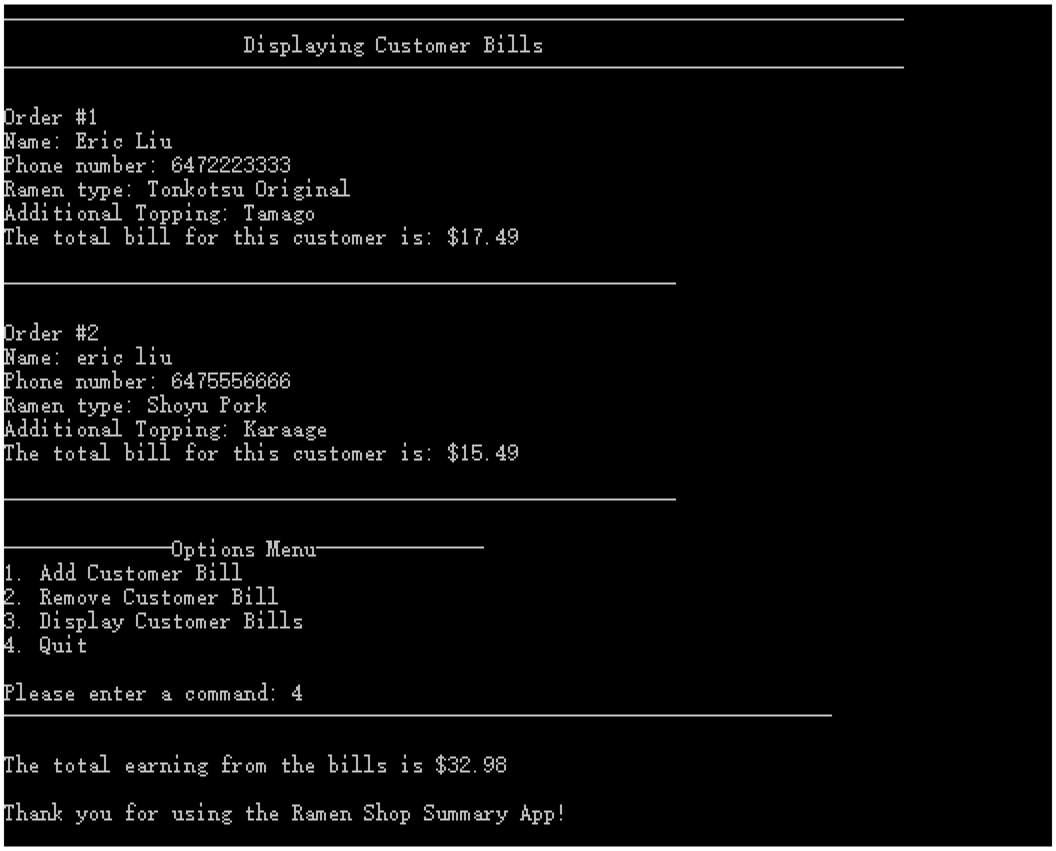
Figure 10: Displays all available orders in the program (two orders were added using option 1)



```
─────────────────────────────────────────────────────────────────────────
                     Deleting Customer Bills
─────────────────────────────────────────────────────────────────────────
Which order do you wish to delete? 1

+++++++++++++++++++++++++++++++++++++++
The order has been successfully removed
+++++++++++++++++++++++++++++++++++++++

─────────────Options Menu─────────────
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 2

─────────────────────────────────────────────────────────────────────────
                     Deleting Customer Bills
─────────────────────────────────────────────────────────────────────────
Which order do you wish to delete? 1

+++++++++++++++++++++++++++++++++++++++
The order has been successfully removed
+++++++++++++++++++++++++++++++++++++++

─────────────Options Menu─────────────
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 3

─────────────────────────────────────────────────────────────────────────
                   Displaying Customer Bills
─────────────────────────────────────────────────────────────────────────

++++++++++++++++++++++++++++
There are currently no orders
++++++++++++++++++++++++++++
```

Figure 11: continuation from the figure 10

Orders are all removed (removing order 1 each time). When all available orders are removed and the user selects option 3, the “There are currently no orders” message is displayed.



```
───────────────────────────────────────────────────────────────────────────
                    Displaying Customer Bills
───────────────────────────────────────────────────────────────────────────

Order #1
Name: Eric Liu
Phone number: 6472223333
Ramen type: Tonkotsu Original
Additional Topping: Tamago
The total bill for this customer is: $17.49

────────────────────────────────────────────────────────

Order #2
Name: eric liu
Phone number: 6475556666
Ramen type: Shoyu Pork
Additional Topping: Karaage
The total bill for this customer is: $15.49

────────────────────────────────────────────────────────

──────────────Options Menu──────────────
1. Add Customer Bill
2. Remove Customer Bill
3. Display Customer Bills
4. Quit

Please enter a command: 4
─────────────────────────────────────────────────────────────────────

The total earning from the bills is $32.98

Thank you for using the Ramen Shop Summary App!
```

Figure 12: continuation to figure 10

When the user chooses option 4 to quit, the program displays total earnings from all the bills and outputs the farewell message.

Testing

The program was tested numerous times for improvements in efficiency, user friendliness, and bugs. Each time I implement bug fixes and improvements in code efficiency, a new version of the cpp file is created. This enabled me to refer back to my code's weaknesses and make adjustments accordingly. In case a modification did not work out, I could easily trace back to the previous version.