Assignment

Personal Data (ID)

Goup Leader:	Tee Le Xuan	**Part 1 - Program Component (60%)**	
Member 1:	Tee Le Xuan 2302521 AM		
Member 2:	Nicoleete Tu Tze Ying 2401431 AM	**Part 2 - Report (40%)**	
Member 3:	Evelyn Chin Shien Lin 2401457 AM		
Member 4:	Hooi Guan Weng 2302821 AM	**Total Mark:**	

Submission Status

No submission	Late submission	No C++ source	No text file	No report	No issue

Part 1 - Program Component (60%)

Components	Remarks	Max. Marks	Mark Scale Allocated	Mark Obtained
Menu and main program		10		
Read from file (building the arrays of structures)		4		
User, System, and File Update: Tracking		8		
User, System, and File Update: Review		8		
Creative Features		10		
Program Style: Indent Style/Blank Line		5		

Program Style: Identifier Names		5		

Presentation, including Q&A		10		

Part 2 - Report and other components (40%)

Components	Remarks	Max. Marks	Mark Scale Allocated	Mark Obtained
Solution Design		10		
Structure Chart		10		
Flowchart/Pseudocode		10		
Sample text file(s)		5		
Screenshots and test cases		5		
Total:		40		

Table of Contents

FUNDAMENTAL OF SOLUTION DESIGN

1.1 Introduction

The core problem faced by UTAR students and staff at the Sungai Long campus is the lack of a centralized platform to find up-to-date information about nearby food stalls, such as their operating hours, types of food offered, and user feedback. This makes it difficult for UTARIANs to make informed decisions when choosing where to eat, especially within the limited time available between classes or during lunch breaks.

To address this issue, we propose developing a Food Stalls Tracking and Review System using C++ programming and text files as the database. This system is specifically designed to:

1. **Track Food Stall Information**:
    - Store and display details such as stall name, location, food type and operating hours
    - Allow users to search and filter stalls based on food type or location
2. **Allow User Reviews and Ratings**:
    - Enable students and staff to submit reviews and rate stalls they have visited on a scale (e.g., 1 to 5 stars).
3. **Store and Manage Data in Text Files**:
    - Stall data and user reviews will be saved in .txt files to ensure the system can run without a complex database.
    - The program will be able to read from and write to these files dynamically as users interact with the system.
4. **User Interaction**:
    - Provide a user-friendly menu interface for:
        - Viewing available stalls
        - Adding reviews
        - Viewing existing reviews
        - Searching/filtering stalls

This system provides a practical, real-time solution to the problem by helping UTARIANs make better food choices conveniently and promoting higher quality

among the food stalls through a public review mechanism.

1.2 Function Overview

Main_menu

void clear_screen()

This function clears the terminal screen before displaying new content. It uses system("cls") for Windows and system("clear") for other operating systems. This helps keep the program interface clean and easy to read.

void instruct()

This function displays the main menu interface to the user. It prints a decorative title and lists the main options: Food Stalls Tracking, Review, and Quit. It makes the program user-friendly and guides the user on what they can do.

bool is_digit(char Opt)

This helper function checks if the user's input is a valid menu option (either '1' or '2'). It returns true if the input is a digit between '1' and '2'; otherwise, it returns false. It is used to validate the input before calling the corresponding function.

void select_option(char &Opt, bool &loop)

This function takes the user's menu selection and performs the corresponding action:

- If the user selects '1', it calls the choose_option() function from the food_track.h module to handle food stall tracking.
- If the user selects '2', it calls the choose_option_r() function to handle the review part.
- If the user enters 'Q' or 'q', it exits the loop and quits the program.
- Any other input will show an error message.

This function also clears the input buffer using cin.ignore() and waits for user confirmation with cin.get() before returning to the menu.

int main()

This is the entry point of the program. It uses a loop to continuously show the main menu until the user decides to quit. It calls the clear_screen(), instruct(), and select_option() functions in each loop iteration to keep the program running interactively.

Food_track

void FOOD_TRACK::clear_screen()

This function clears the terminal screen using the appropriate command for the operating system (cls for Windows, clear for others). It ensures that the menu or display is refreshed cleanly each time it is shown to the user.

void FOOD_TRACK::instruct()

This function displays the sub-menu for selecting the **type of food store**, showing options like:

- <1> Restaurant
- <2> Cafe
- <3> Fast Food Store

  It helps users decide which category of food stalls they want to view or review.

void FOOD_TRACK::choose_option()

This function handles the **Food Stalls Tracking** part:

- It repeatedly displays the food store type menu.
- Based on user input, it calls the corresponding function from the STORE class:
  - restaurant(), cafe(), or fast_food()
- If the user enters 'Q', the loop breaks and returns to the main menu.
- Invalid inputs show an error message and prompt the user again.

void FOOD_TRACK::choose_option_r()

This function handles the **Review** part of the system:

- Like choose_option(), it shows the food type menu.
- Based on user input, it calls the matching function from the REVIEW class:
  - restaurant_r(), cafe_r(), or fast_food_r()
- If the user inputs 'Q', it exits the loop and returns to the main menu.
- Invalid inputs are handled the same way with an error message and retry.

Store

void STORE::clear_screen()

Clears the console screen, adapting to both Windows and Unix-like systems.

void STORE::display_data (string text_file)

Displays food stall data (codes, names, and prices) from a specified text file in a structured format.

bool STORE::check_code(string text_file, string code)

Checks if a specific food code exists in the given text file.

void STORE::add_data(string text_file, string code, string food, double f_price)

Allows the addition of new food items (with code, description, and price) to the text file, ensuring no duplicate codes.

void STORE::modify_data(string text_file, string code, string food, double f_price)

Facilitates the modification of existing food items by code, allowing changes to the code, description, or price.

void STORE::delete_data(string text_file, string code, string food, double f_price)

Deletes a food item by its code from the text file.

void STORE::calculate(string text_file, string code, int quantity, double f_price)

Calculates the total price for a selected food item based on its code and quantity.

void STORE::track(string text_file, int distance, string hrs)

Manages the tracking of food orders and operations in various restaurants or cafes by allowing adding, modifying, deleting, calculating, and more.

void STORE::restaurant()// void STORE::cafe()// void STORE::fast_food()

Display options for different restaurants or cafes and invoke the track() function to manage food-related operations.

Review

void REVIEW::clear_screen()

Clears the terminal screen, adapting to both Windows and Unix-like systems.

void REVIEW::user_info(string &name, string &city, string &state, string &date)

Prompts the user to input personal details like name, city, state, and date, and stores them in respective variables.

void REVIEW::user_review(string name, string city, string state, string date, string &rate, string &main_reason, string &comments)

Allows the user to provide a review, including a rating, reason for the rating, and comments, which are saved temporarily in a file (temp_review.txt).

void REVIEW::display_review(string text_file)

Displays all the reviews stored in a given text file.

void REVIEW::temp_review()

Displays a preview of the current review stored in temp_review.txt.

void REVIEW::modify_review(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments)

Allows the user to modify an existing review by updating personal information and review details, and then confirms the changes.

void REVIEW::delete_review(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments)

Deletes the current review by clearing the contents of temp_review.txt.

void REVIEW::confirmation(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments)

Appends the review from temp_review.txt to the main review file and displays all confirmed reviews.

void REVIEW::give_comment(string text_file)

Prompts the user to enter a review, provides options to confirm, modify, delete, or quit, and processes the user's choice accordingly.

void REVIEW::track(string text_file)

Displays a menu for users to select the type of review they want to give (restaurant, cafe, or fast food) and manages the process based on their choice.

void REVIEW::restaurant_r()

Displays available restaurants, lets the user select one, and then proceeds to the review tracking for that restaurant.

void REVIEW::cafe_r()

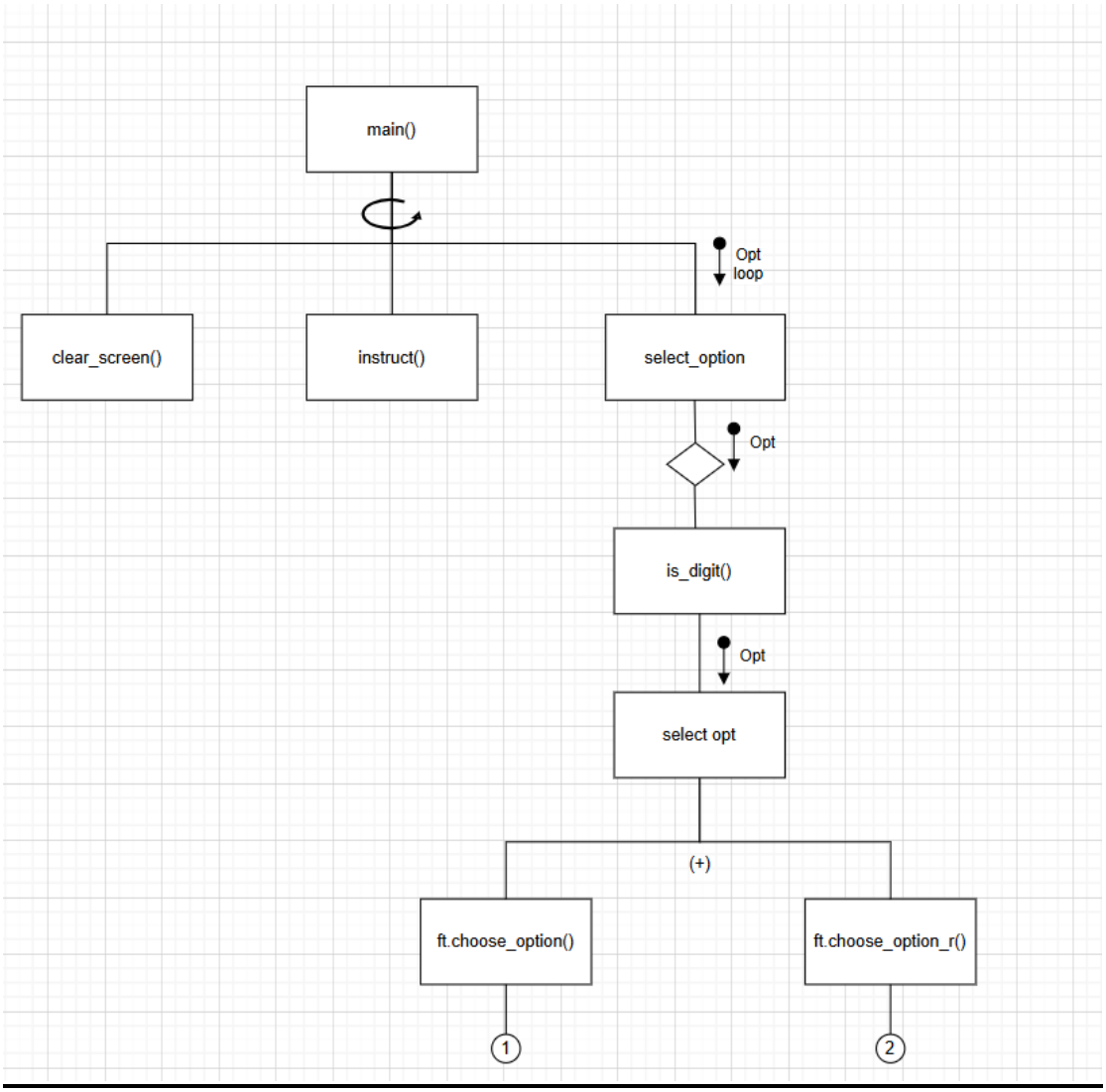
Displays available cafes, lets the user select one, and then proceeds to the review tracking for that cafe.

void REVIEW::fast_food_r()

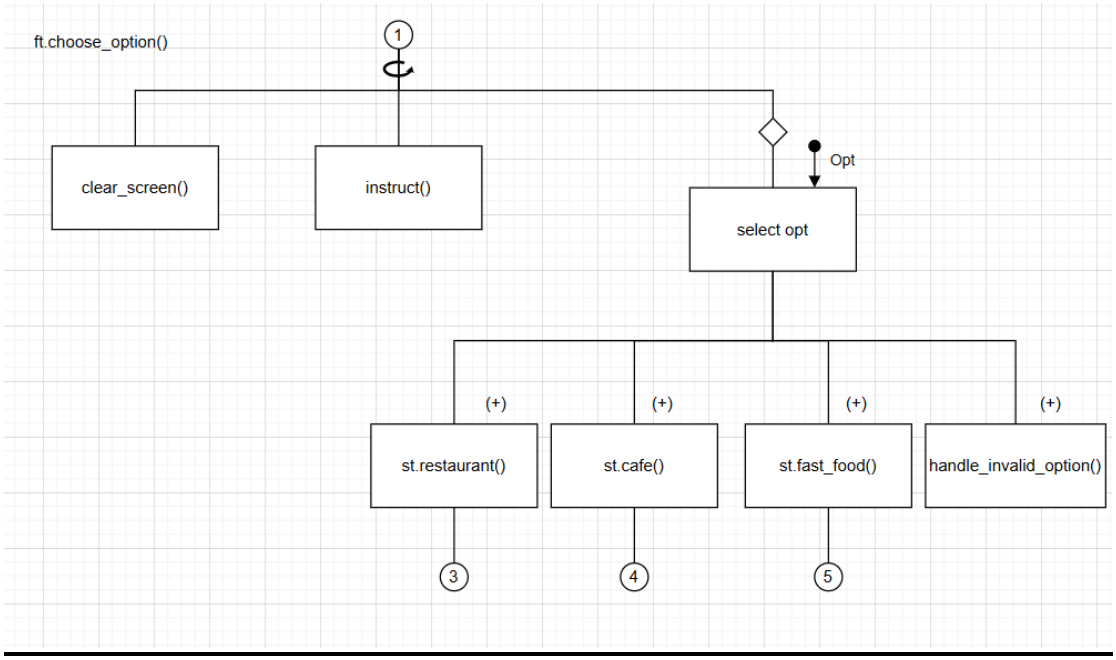
Displays available fast food stores, lets the user select one, and then proceeds to the review tracking for that store.

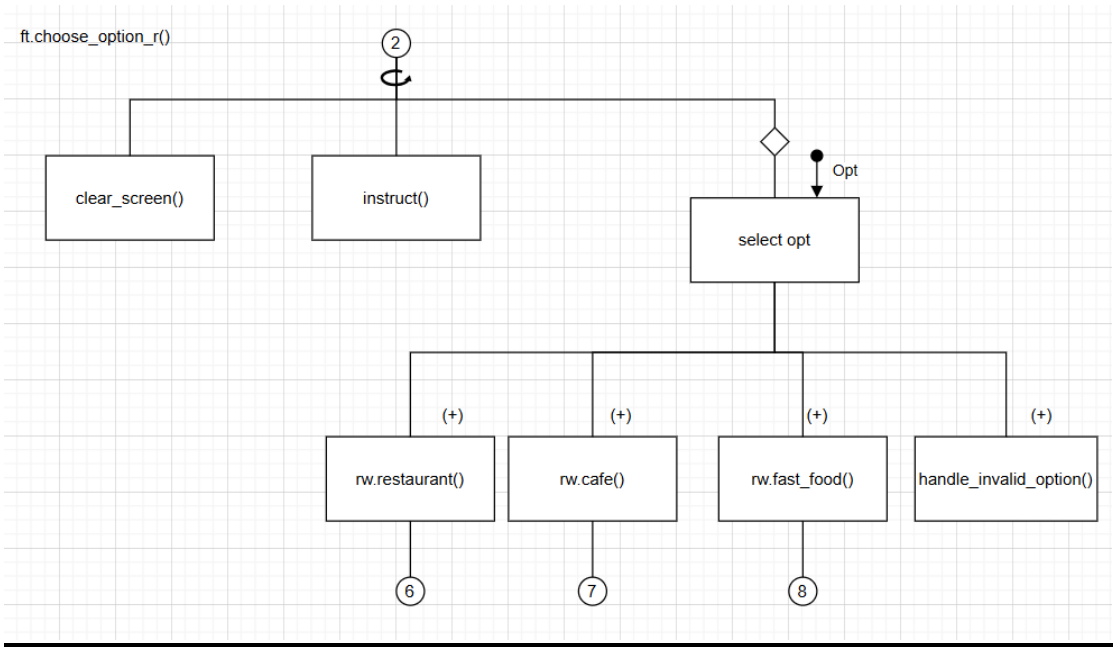
1.3 Structure Chart

Main menu

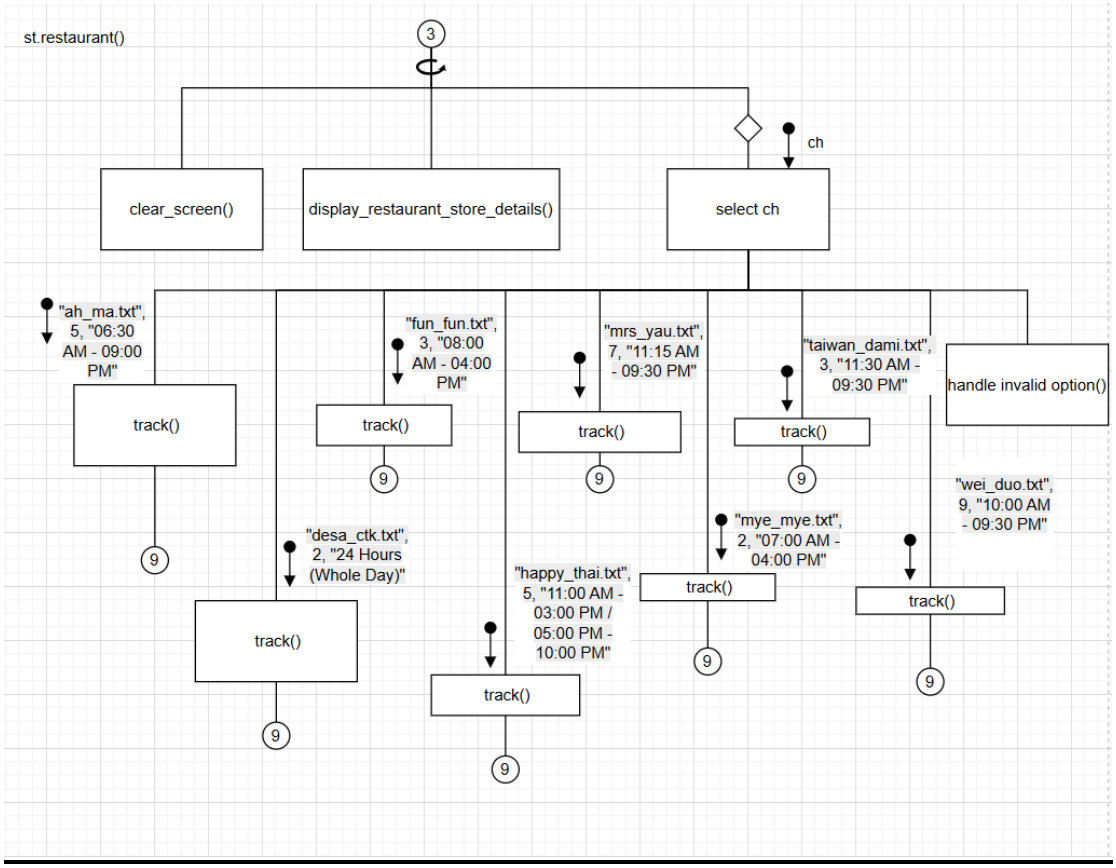


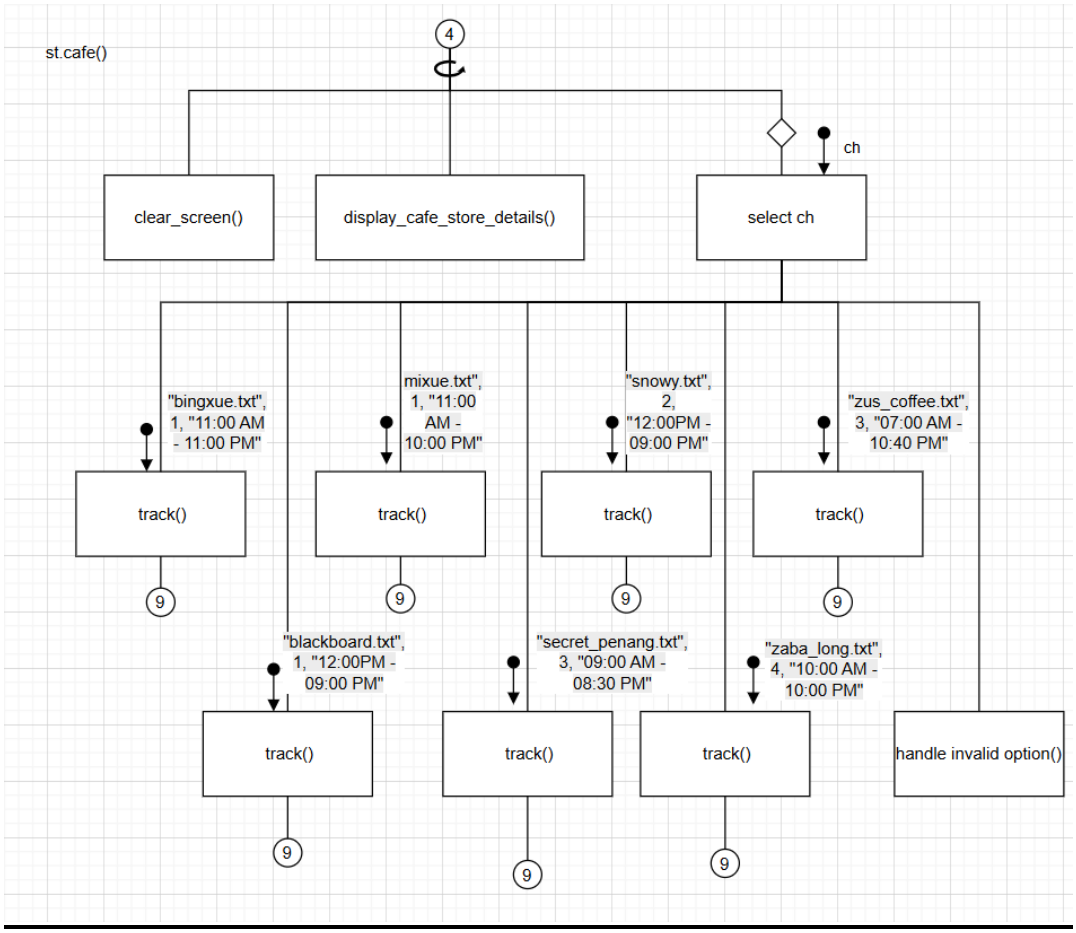
Food track

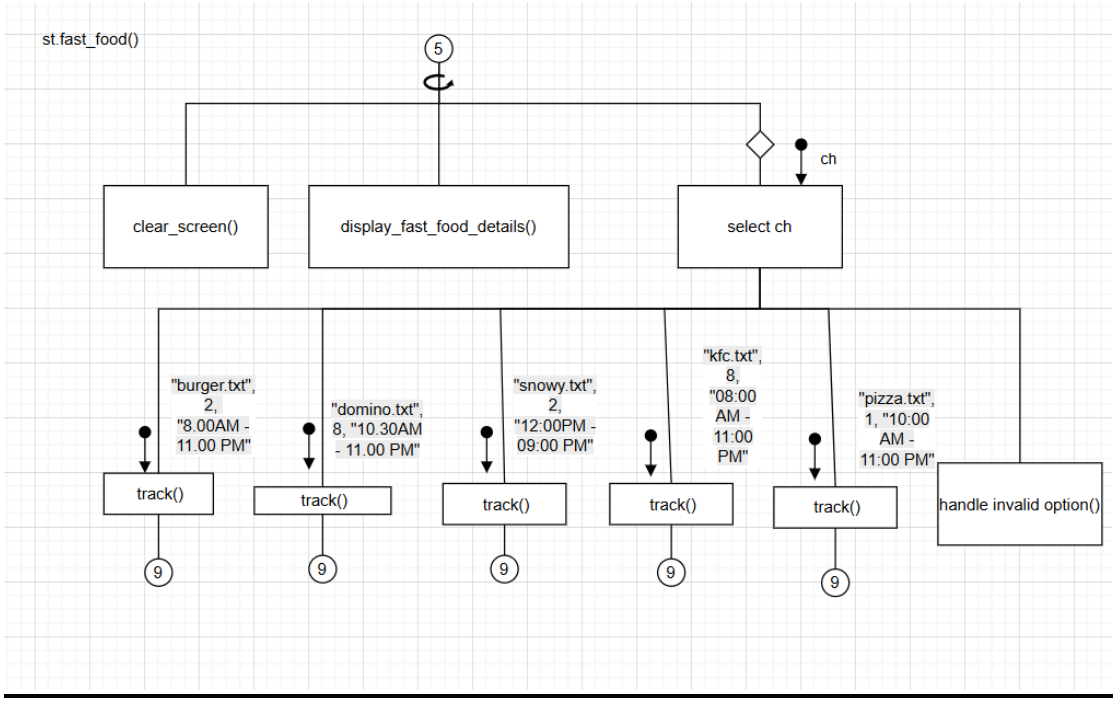

ft.choose_option()
1
clear_screen()
instruct()
Opt
select opt
(+)
(+)
(+)
(+)
st.restaurant()
st.cafe()
st.fast_food()
handle_invalid_option()
3
4
5

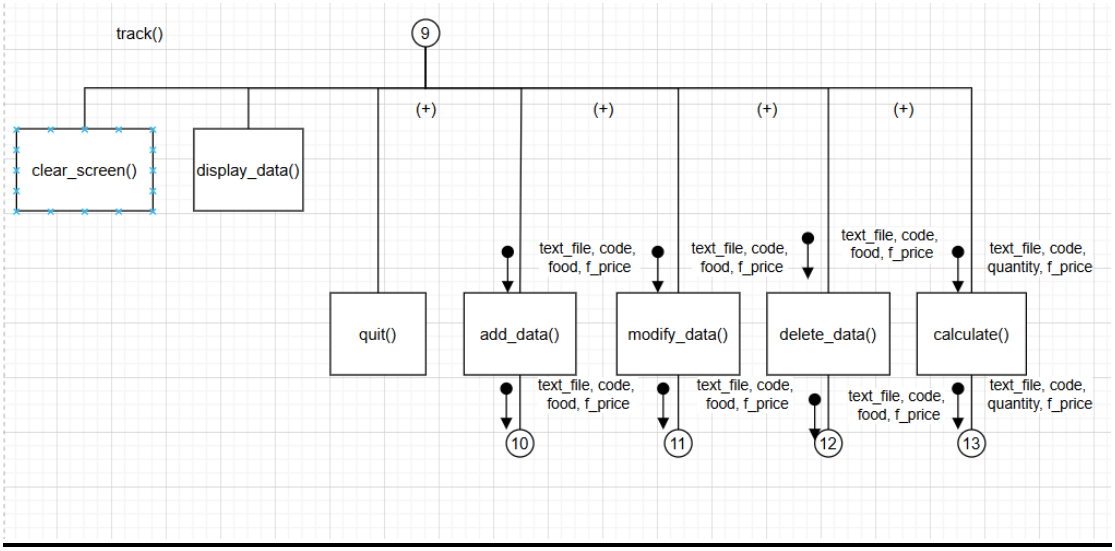

ft.choose_option_r()
2
clear_screen()
instruct()
Opt
select opt
(+)
(+)
(+)
(+)
rw.restaurant()
rw.cafe()
rw.fast_food()
handle_invalid_option()
6
7
8

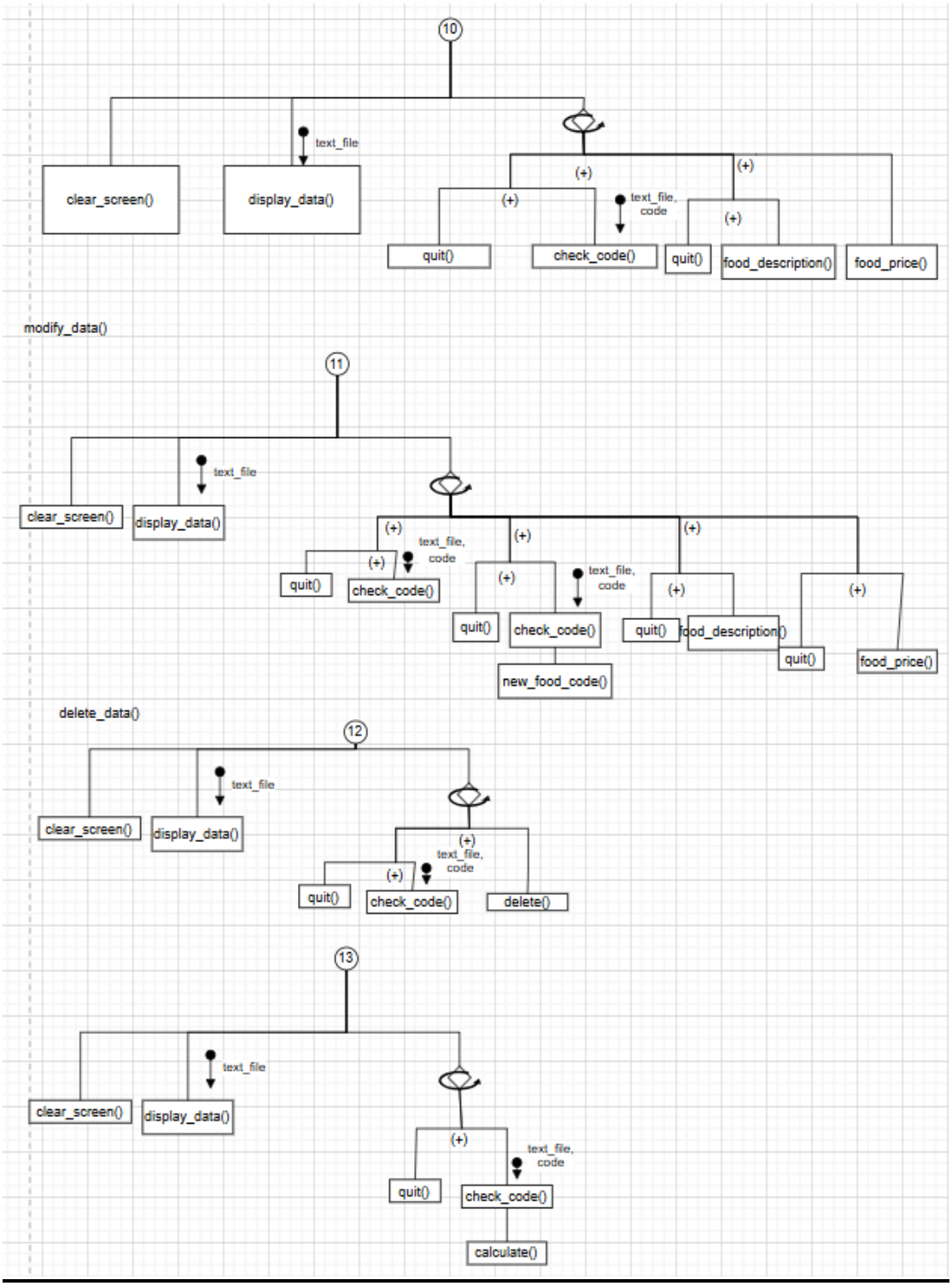
Store


st.restaurant()
3
clear_screen()
display_restaurant_store_details()
ch
select ch
"ah_ma.txt", 5, "06:30 AM - 09:00 PM"
track()
9
"desa_ctk.txt", 2, "24 Hours (Whole Day)"
track()
9
"fun_fun.txt", 3, "08:00 AM - 04:00 PM"
track()
9
"happy_thai.txt", 5, "11:00 AM - 03:00 PM / 05:00 PM - 10:00 PM"
track()
9
"mrs_yau.txt", 7, "11:15 AM - 09:30 PM"
track()
9
"mye_mye.txt", 2, "07:00 AM - 04:00 PM"
track()
9
"taiwan_dami.txt", 3, "11:30 AM - 09:30 PM"
track()
9
handle invalid option()
"wei_duo.txt", 9, "10:00 AM - 09:30 PM"
track()
9

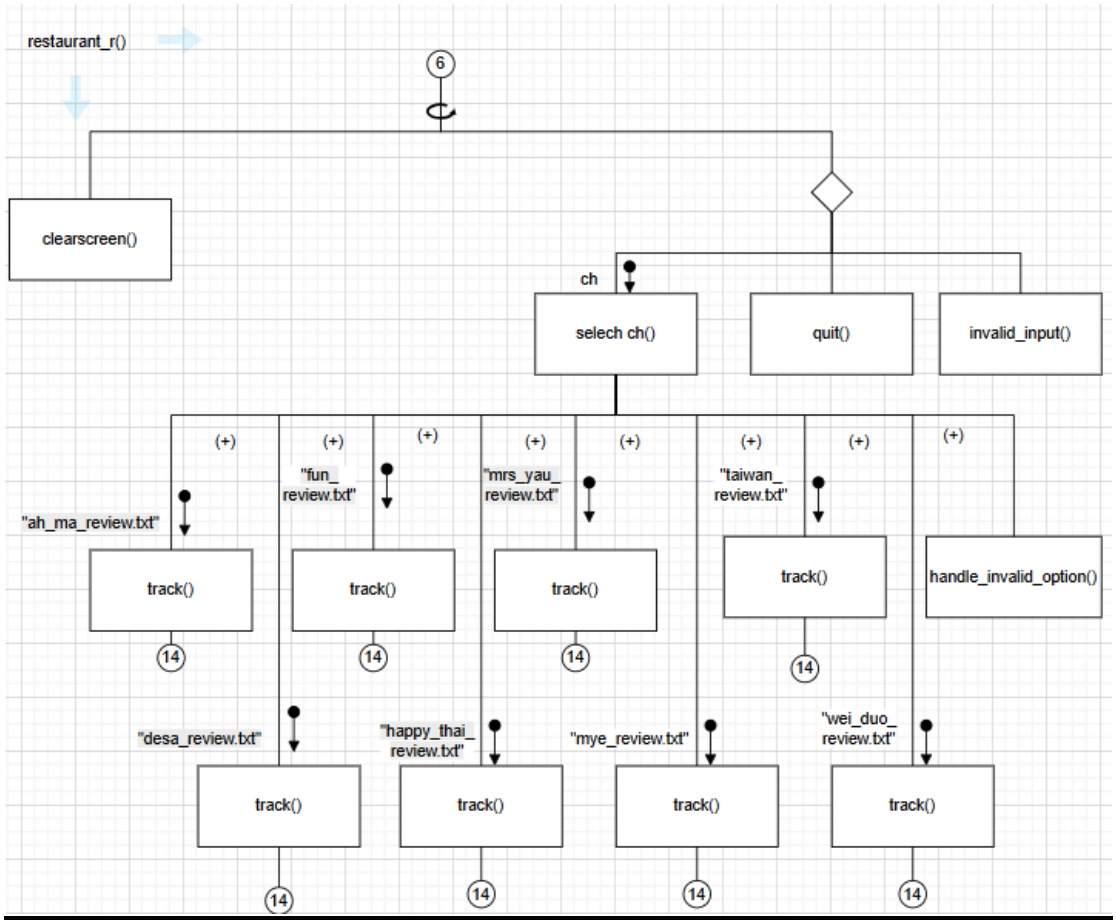

st.cafe()
4
ch
clear_screen()
display_cafe_store_details()
select ch
"bingxue.txt", 1, "11:00 AM - 11:00 PM"
mixue.txt", 1, "11:00 AM - 10:00 PM"
"snowy.txt", 2, "12:00PM - 09:00 PM"
"zus_coffee.txt", 3, "07:00 AM - 10:40 PM"
track()
track()
track()
track()
9
9
9
9
"blackboard.txt", 1, "12:00PM - 09:00 PM"
"secret_penang.txt", 3, "09:00 AM - 08:30 PM"
"zaba_long.txt", 4, "10:00 AM - 10:00 PM"
track()
track()
track()
handle invalid option()
9
9
9

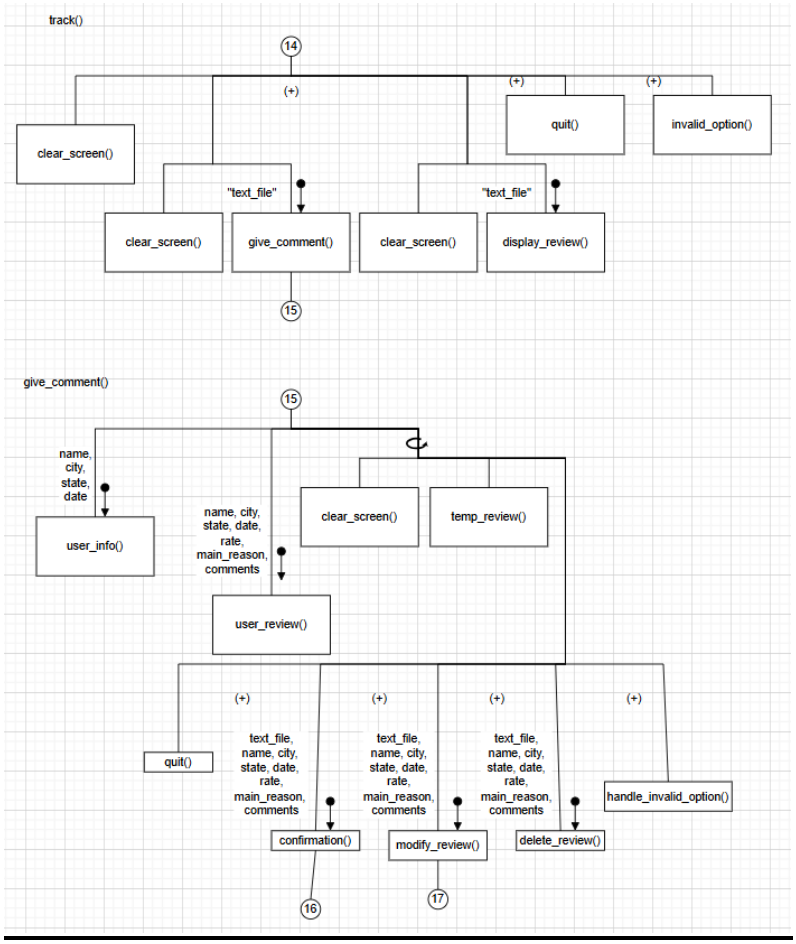

st.fast_food()
5
ch
clear_screen()
display_fast_food_details()
select ch
"burger.txt", 2, "8.00AM - 11.00 PM"
"domino.txt", 8, "10.30AM - 11.00 PM"
"snowy.txt", 2, "12:00PM - 09:00 PM"
"kfc.txt", 8, "08:00 AM - 11:00 PM"
"pizza.txt", 1, "10:00 AM - 11:00 PM"
track()
track()
track()
track()
track()
handle invalid option()
9
9
9
9
9

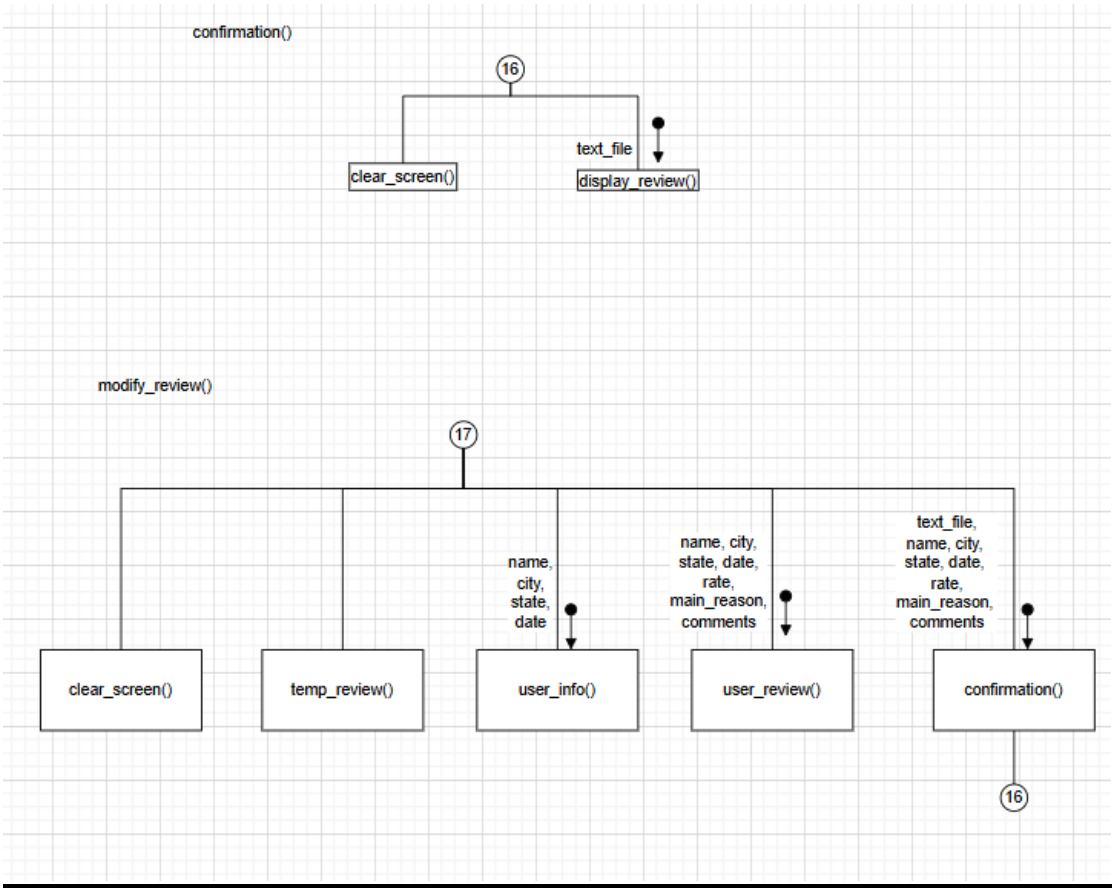

track()
9
(+)
(+)
(+)
(+)
clear_screen()
display_data()
quit()
add_data()
modify_data()
delete_data()
calculate()
text_file, code, food, f_price
text_file, code, food, f_price
text_file, code, food, f_price
text_file, code, quantity, f_price
text_file, code, food, f_price
text_file, code, food, f_price
text_file, code, food, f_price
text_file, code, quantity, f_price
10
11
12
13


10
text_file
clear_screen()
display_data()
(+)
(+)
text_file, code
(+)
quit()
check_code()
quit()
food_description()
food_price()
modify_data()
11
text_file
clear_screen()
display_data()
(+)
text_file, code
(+)
quit()
check_code()
(+)
text_file, code
quit()
check_code()
(+)
quit()
food_description()
(+)
quit()
food_price()
new_food_code()
delete_data()
12
text_file
clear_screen()
display_data()
(+)
text_file, code
(+)
quit()
check_code()
delete()
13
text_file
clear_screen()
display_data()
(+)
text_file, code
quit()
check_code()
calculate()

Review

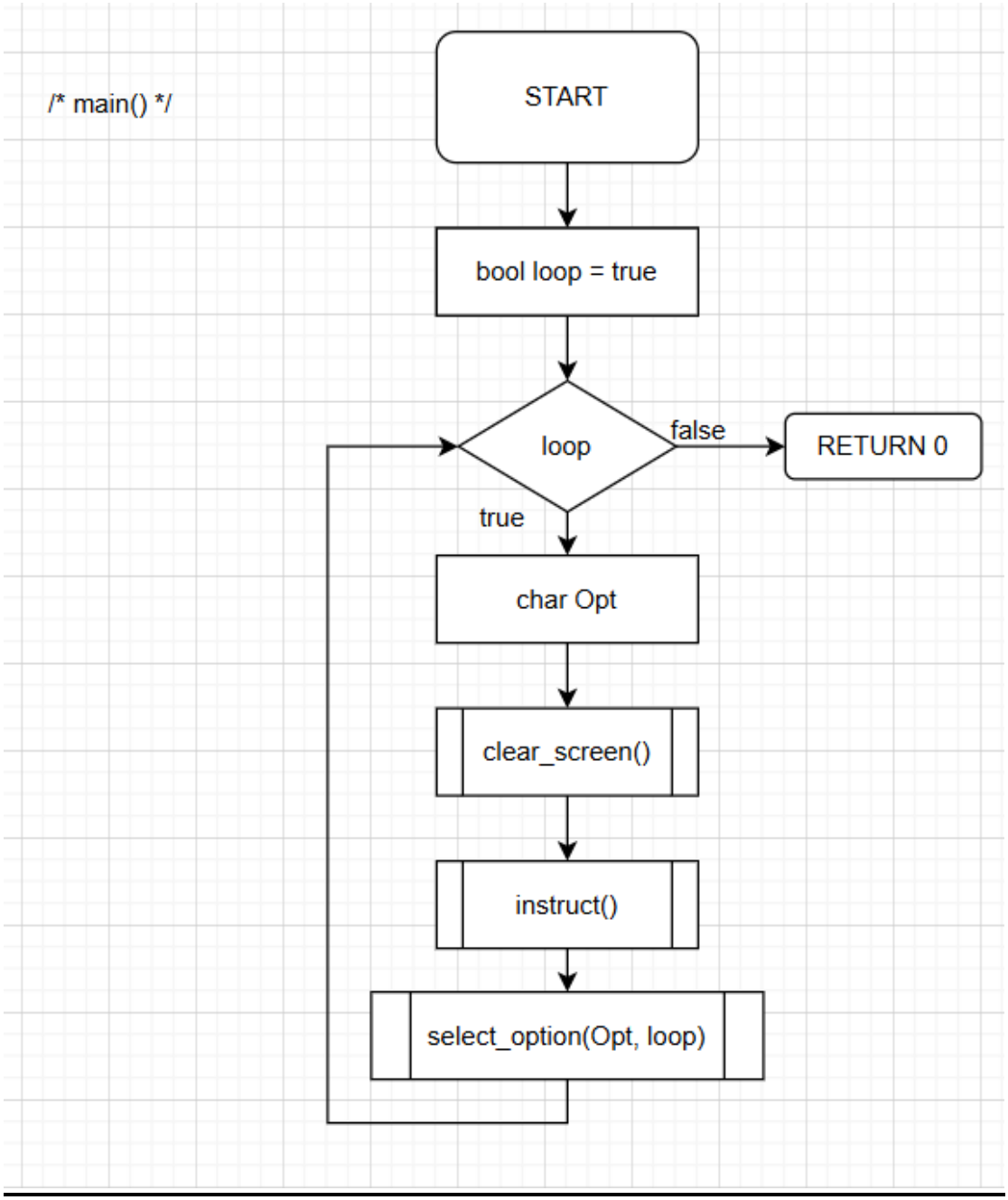

restaurant_r()
6
clearscreen()
ch
selech ch()
quit()
invalid_input()
(+)
"ah_ma_review.txt"
"fun_ review.txt"
"mrs_yau_ review.txt"
"taiwan_ review.txt"
"desa_review.txt"
"happy_thai_ review.txt"
"mye_review.txt"
"wei_duo_ review.txt"
track()
handle_invalid_option()
14

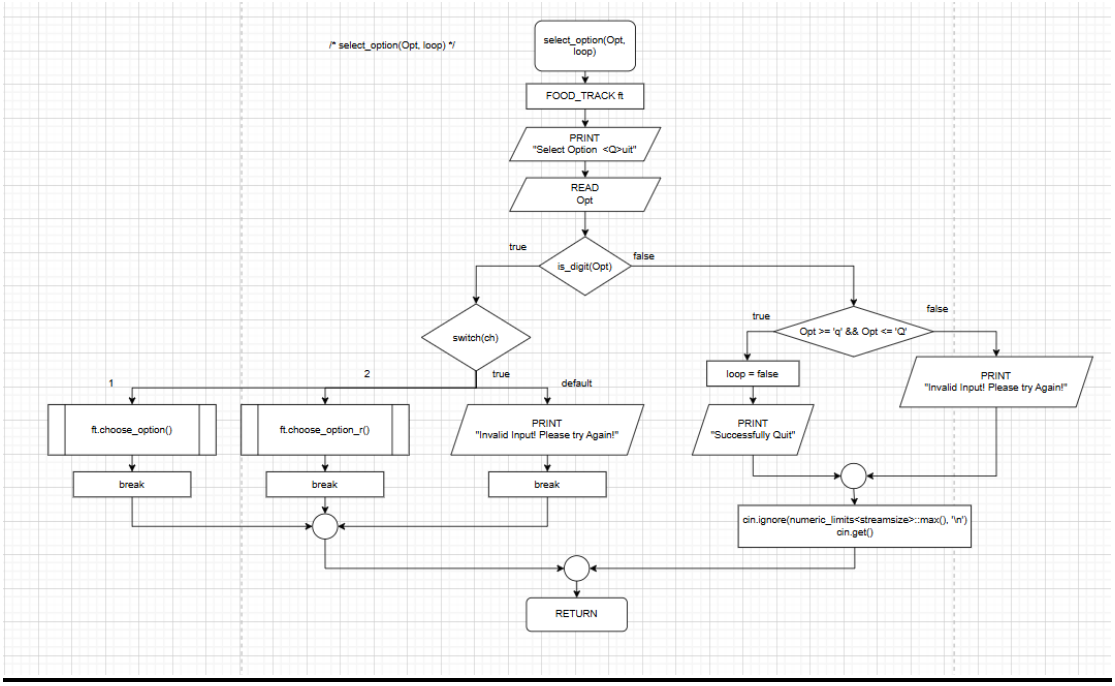

track()
14
(+)
(+)
(+)
clear_screen()
quit()
invalid_option()
"text_file"
"text_file"
clear_screen()
give_comment()
clear_screen()
display_review()
15
give_comment()
15
name,
city,
state,
date
user_info()
name, city,
state, date,
rate,
main_reason,
comments
clear_screen()
temp_review()
user_review()
(+)
(+)
(+)
(+)
quit()
text_file,
name, city,
state, date,
rate,
main_reason,
comments
text_file,
name, city,
state, date,
rate,
main_reason,
comments
text_file,
name, city,
state, date,
rate,
main_reason,
comments
handle_invalid_option()
confirmation()
modify_review()
delete_review()
16
17

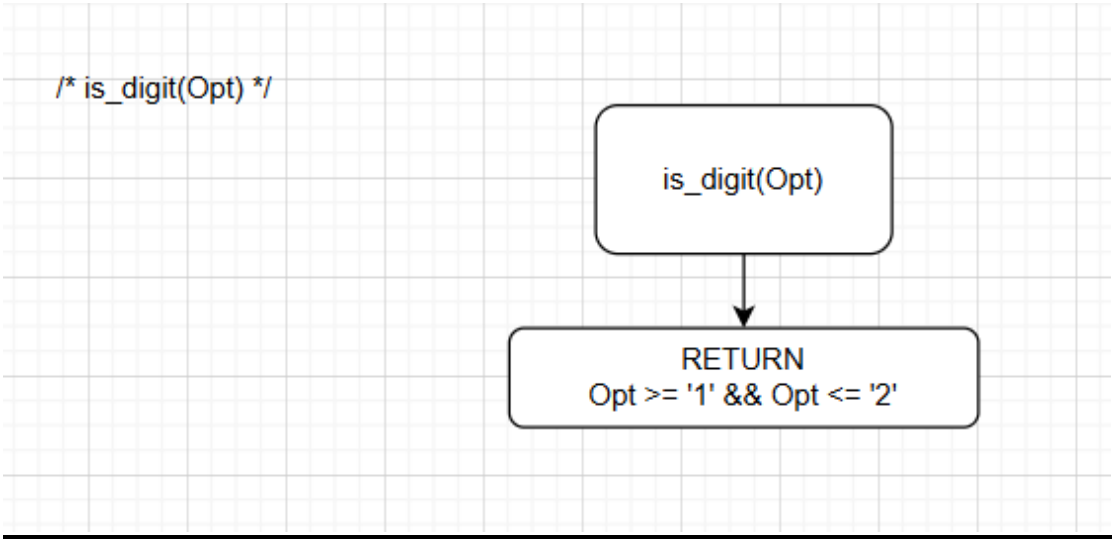

confirmation()
16
clear_screen()
text_file
display_review()
modify_review()
17
clear_screen()
temp_review()
name,
city,
state,
date
user_info()
name, city,
state, date,
rate,
main_reason,
comments
user_review()
text_file,
name, city,
state, date,
rate,
main_reason,
comments
confirmation()
16

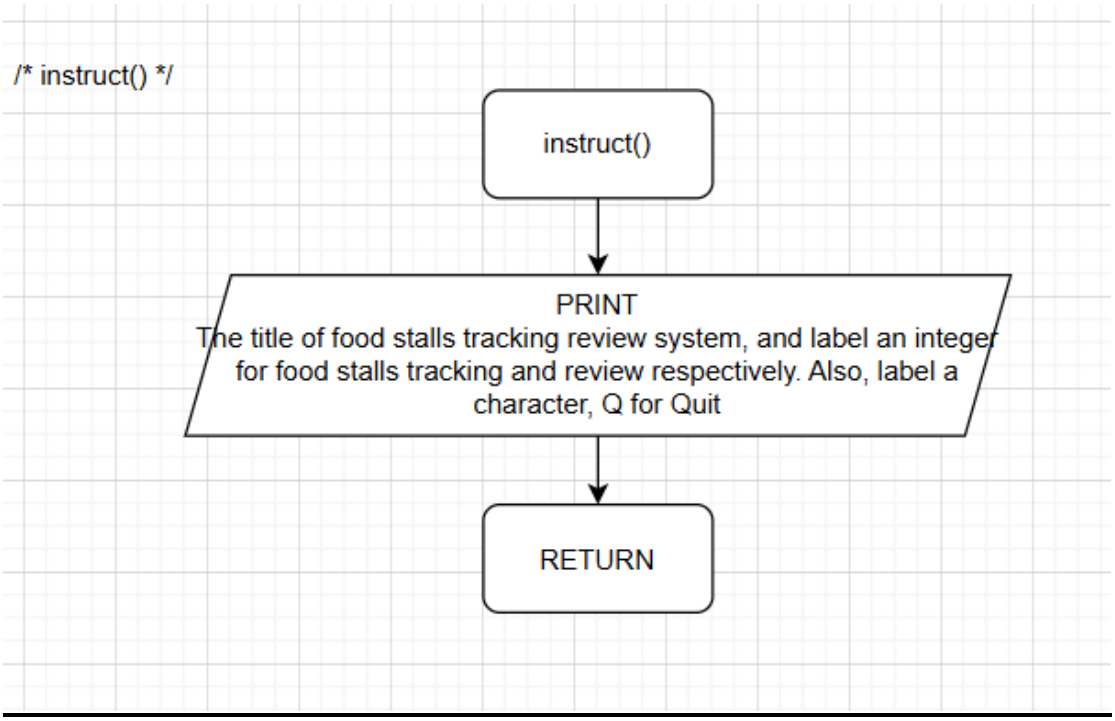
1.4 Flowchart

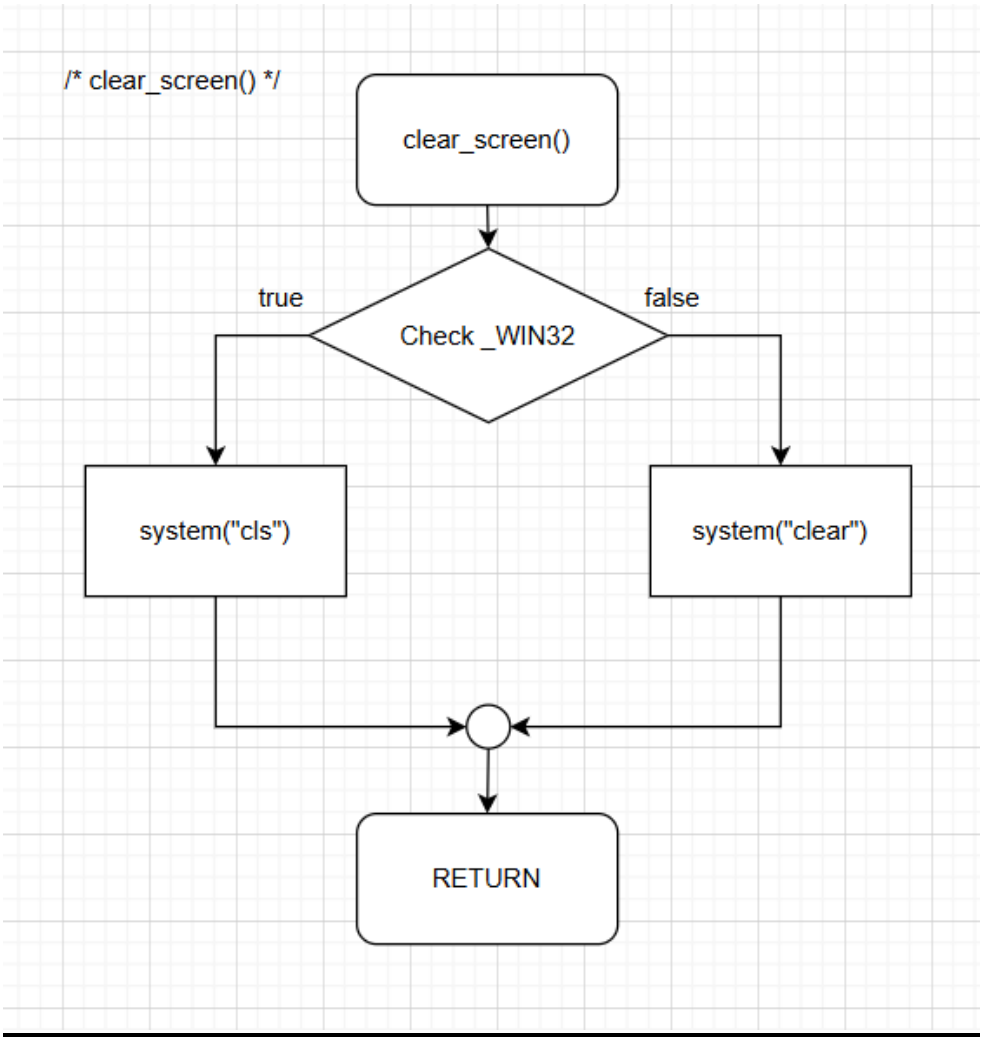
Main menu



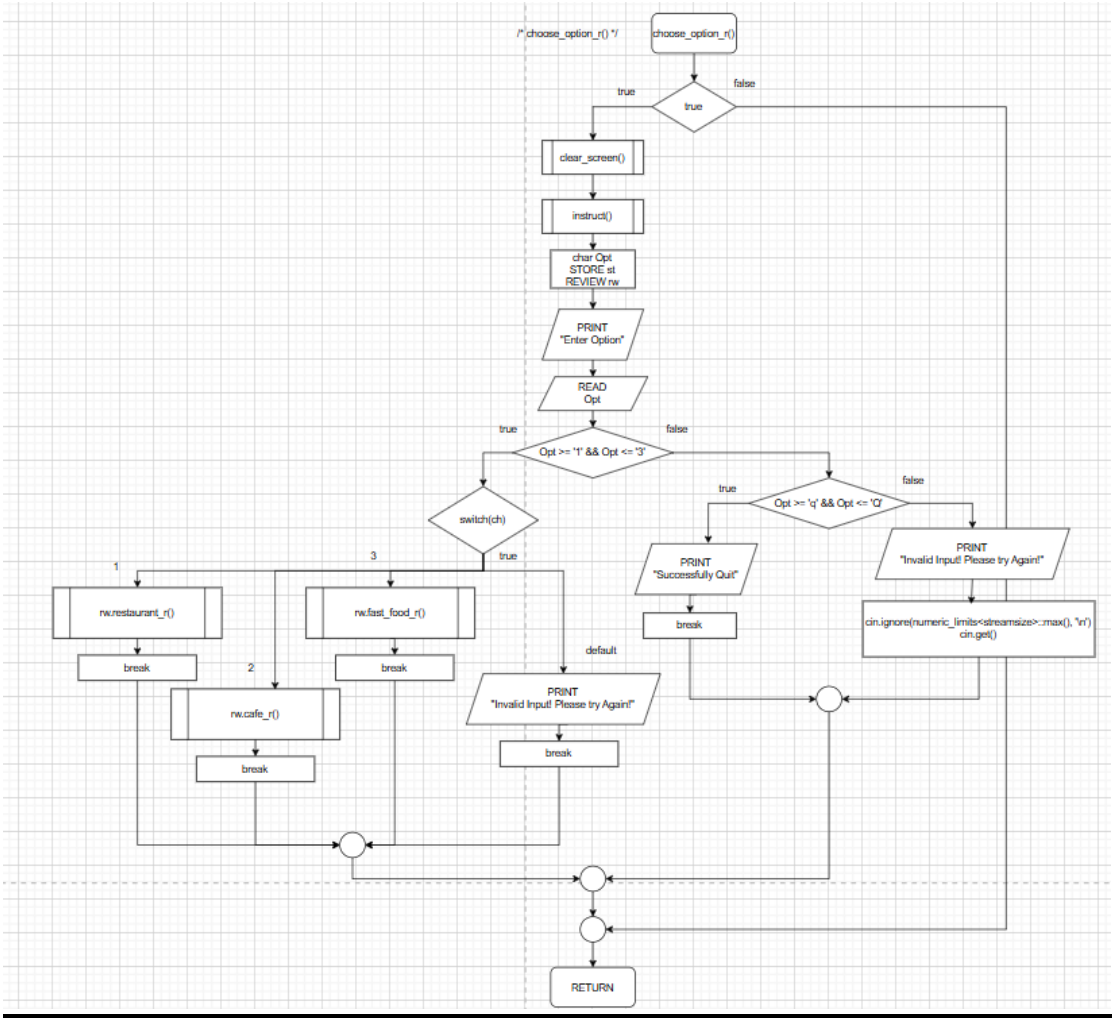

/* select_option(Opt, loop) */
select_option(Opt, loop)
FOOD_TRACK ft
PRINT "Select Option <Q>uit"
READ Opt
is_digit(Opt)
true
false
switch(ch)
1
2
true
default
ft.choose_option()
ft.choose_option_r()
PRINT "Invalid Input! Please try Again!"
break
break
break
Opt >= 'q' && Opt <= 'Q'
true
false
loop = false
PRINT "Invalid Input! Please try Again!"
PRINT "Sucessfully Quit"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
RETURN

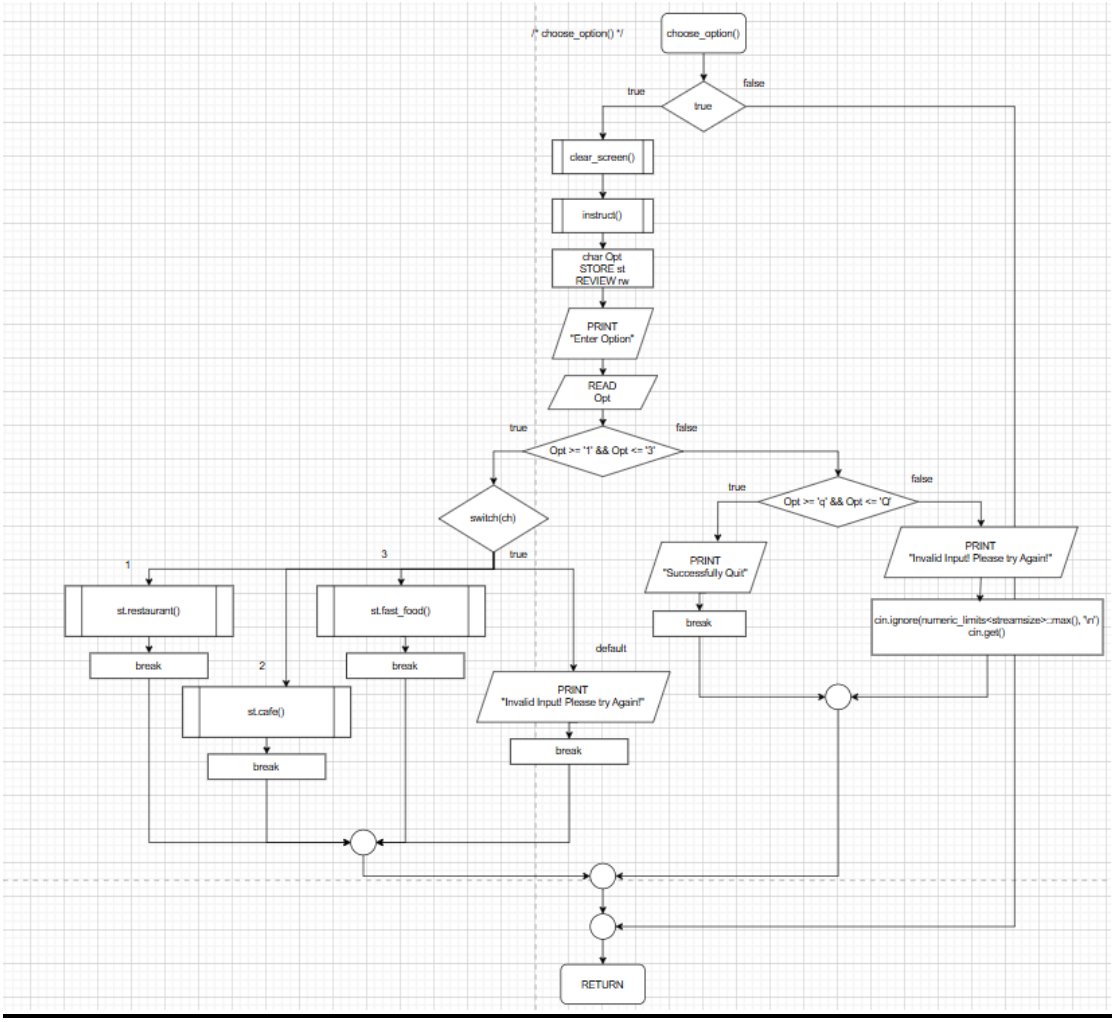

/* is_digit(Opt) */
is_digit(Opt)
RETURN
Opt >= '1' && Opt <= '2'

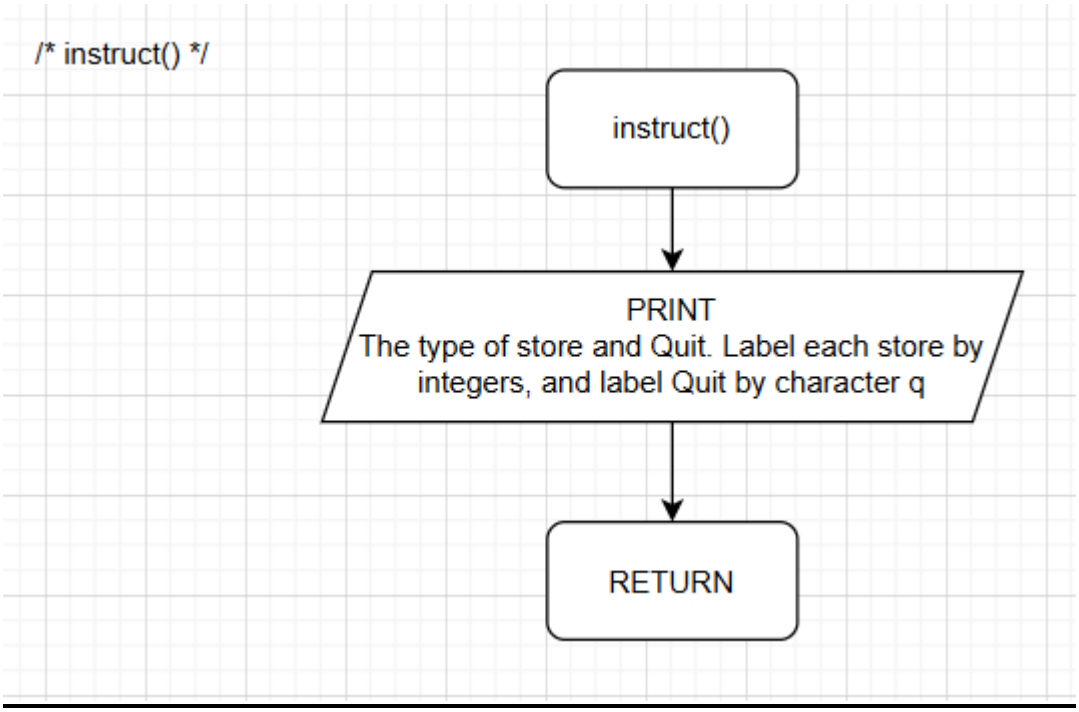

/* instruct() */
instruct()
PRINT
The title of food stalls tracking review system, and label an integer for food stalls tracking and review respectively. Also, label a character, Q for Quit
RETURN

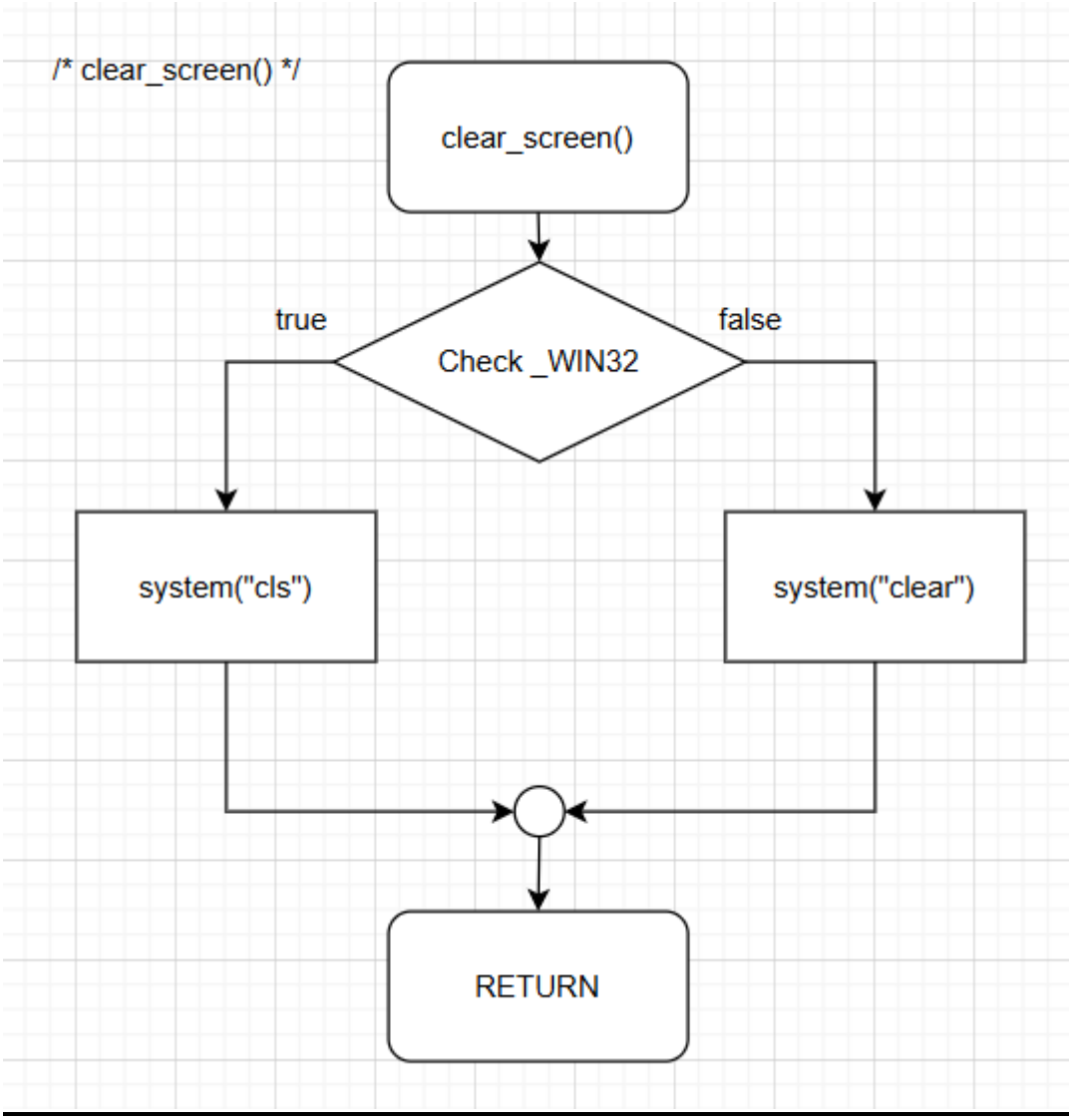

/* clear_screen() */
clear_screen()
true
Check _WIN32
false
system("cls")
system("clear")
RETURN

Food_track

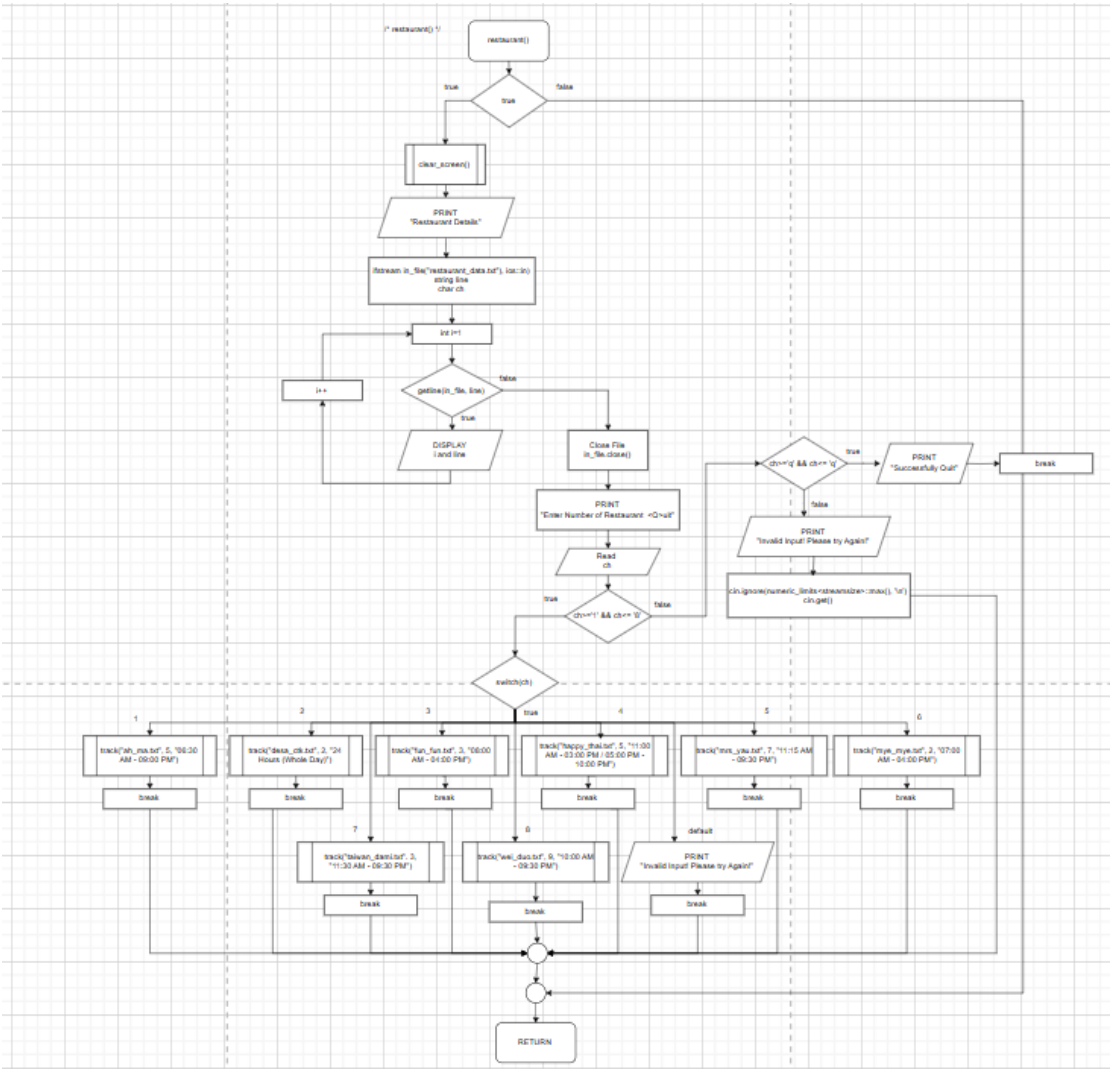

/* choose_option_r() */
choose_option_r()
true
false
true
clear_screen()
instruct()
char Opt
STORE st
REVIEW rw
PRINT
"Enter Option"
READ
Opt
true
false
Opt >= '1' && Opt <= '3'
switch(ch)
true
false
Opt >= 'q' && Opt <= 'Q'
1
3
rw.restaurant_r()
rw.fast_food_r()
break
break
2
rw.cafe_r()
break
default
PRINT
"Invalid Input! Please try Again!"
break
PRINT
"Successfully Quit"
break
PRINT
"Invalid Input! Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
RETURN

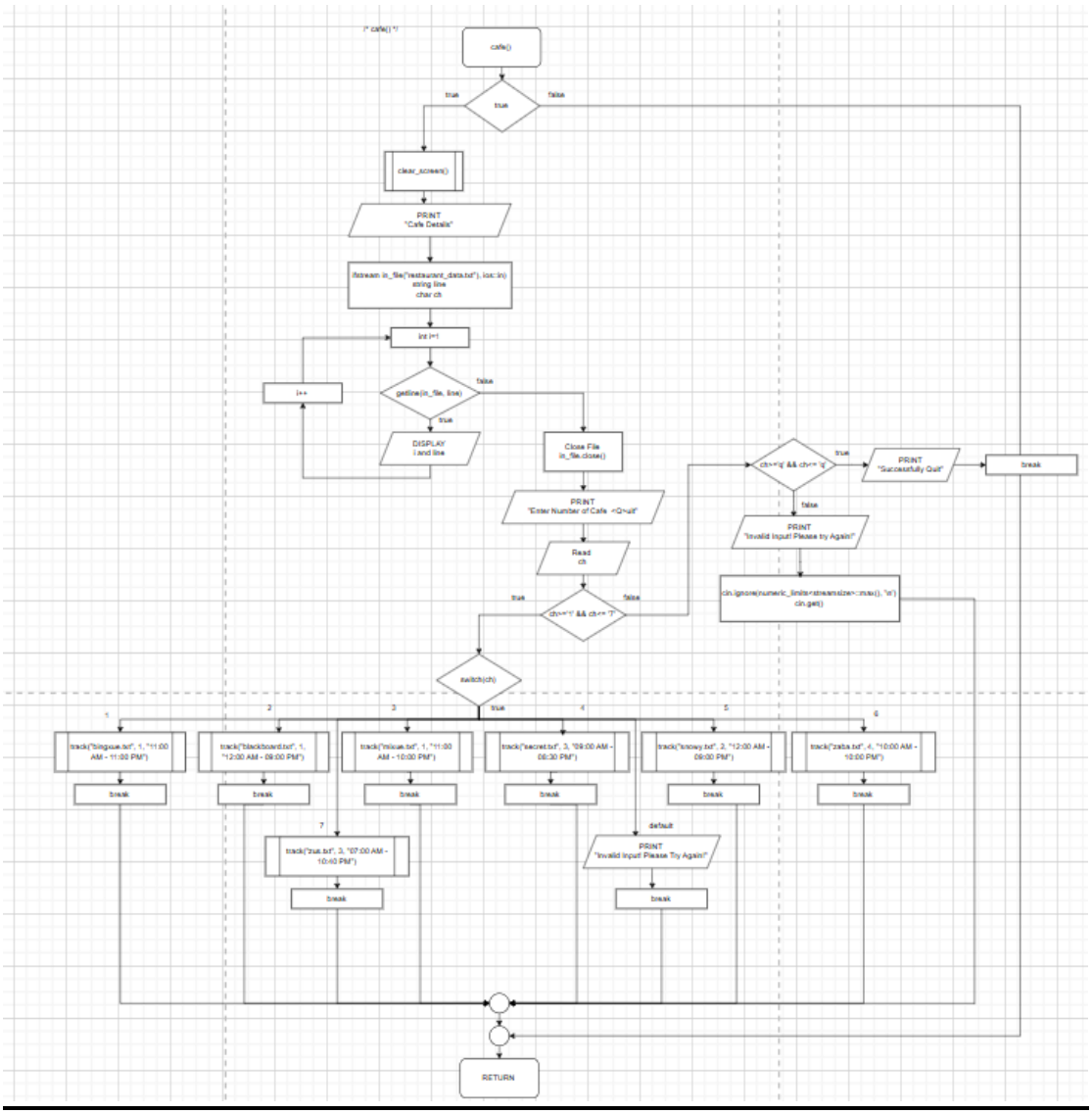

/* choose_option() */
choose_option()
true
false
clear_screen()
instruct()
char Opt
STORE st
REVIEW rw
PRINT
"Enter Option"
READ
Opt
Opt >= '1' && Opt <= '3'
switch(ch)
1
3
2
default
st.restaurant()
st.fast_food()
st.cafe()
break
PRINT
"Invalid Input! Please try Again!"
Opt >= 'q' && Opt <= 'Q'
PRINT
"Successfully Quit"
PRINT
"Invalid Input! Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
RETURN

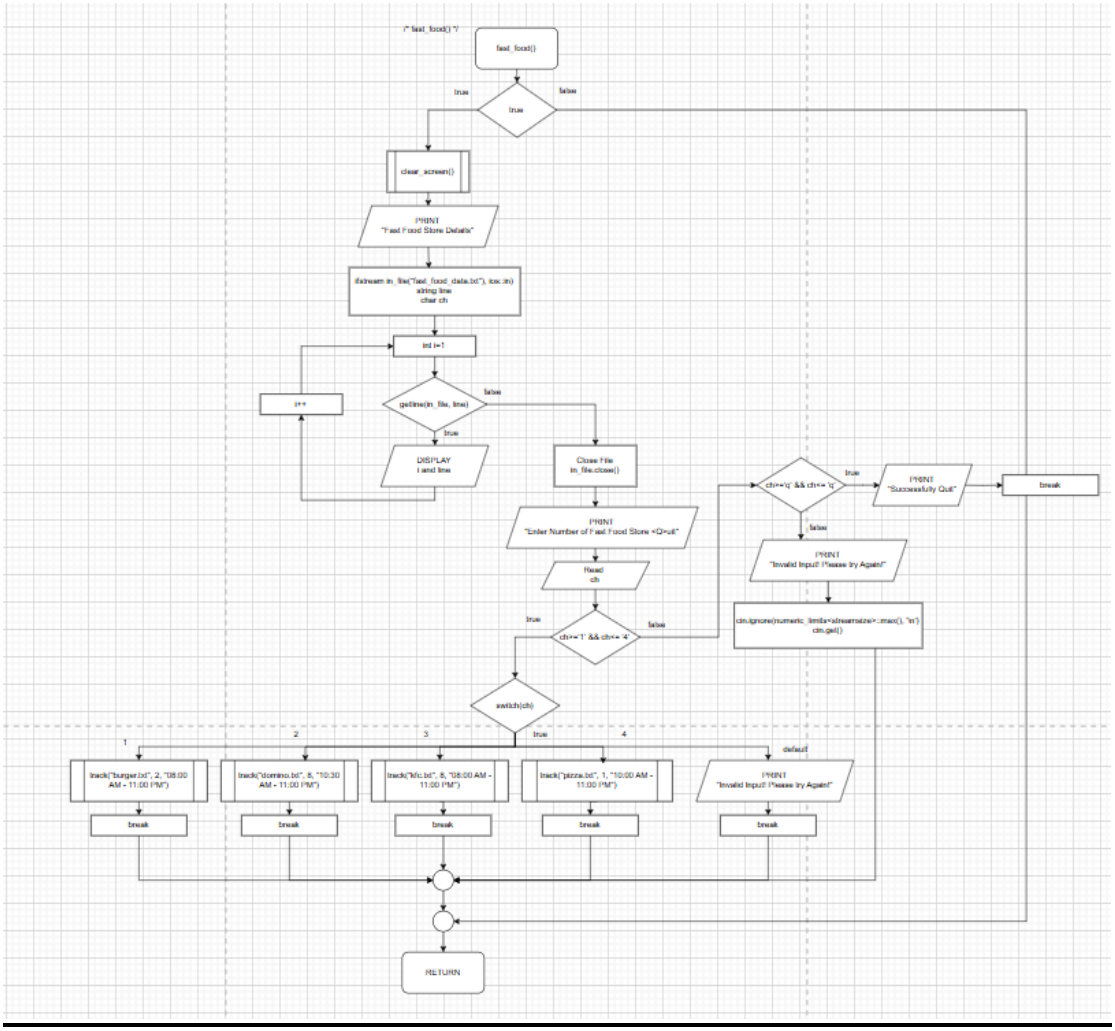

/* instruct() */
instruct()
PRINT
The type of store and Quit. Label each store by integers, and label Quit by character q
RETURN

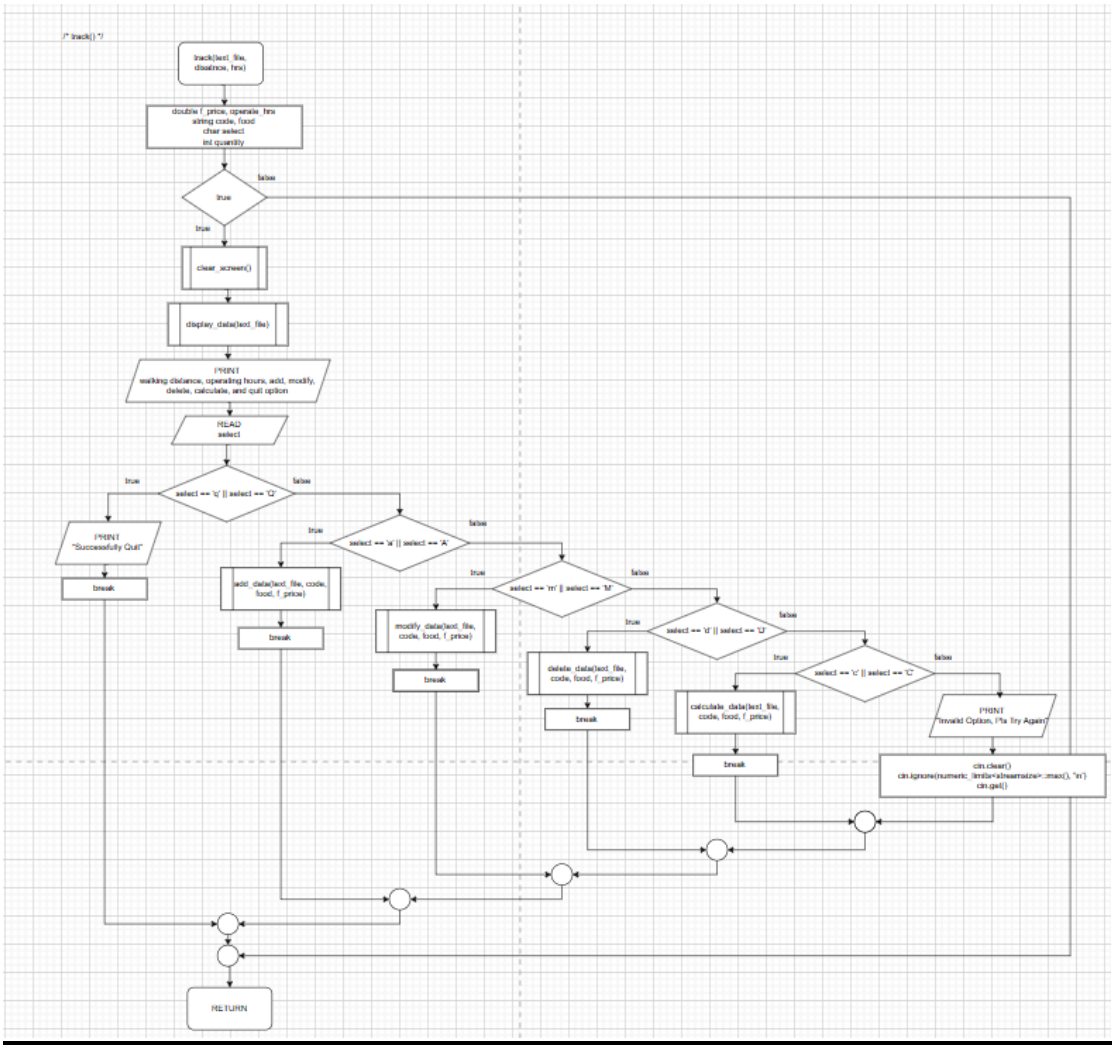

/* clear_screen() */
clear_screen()
true
Check _WIN32
false
system("cls")
system("clear")
RETURN

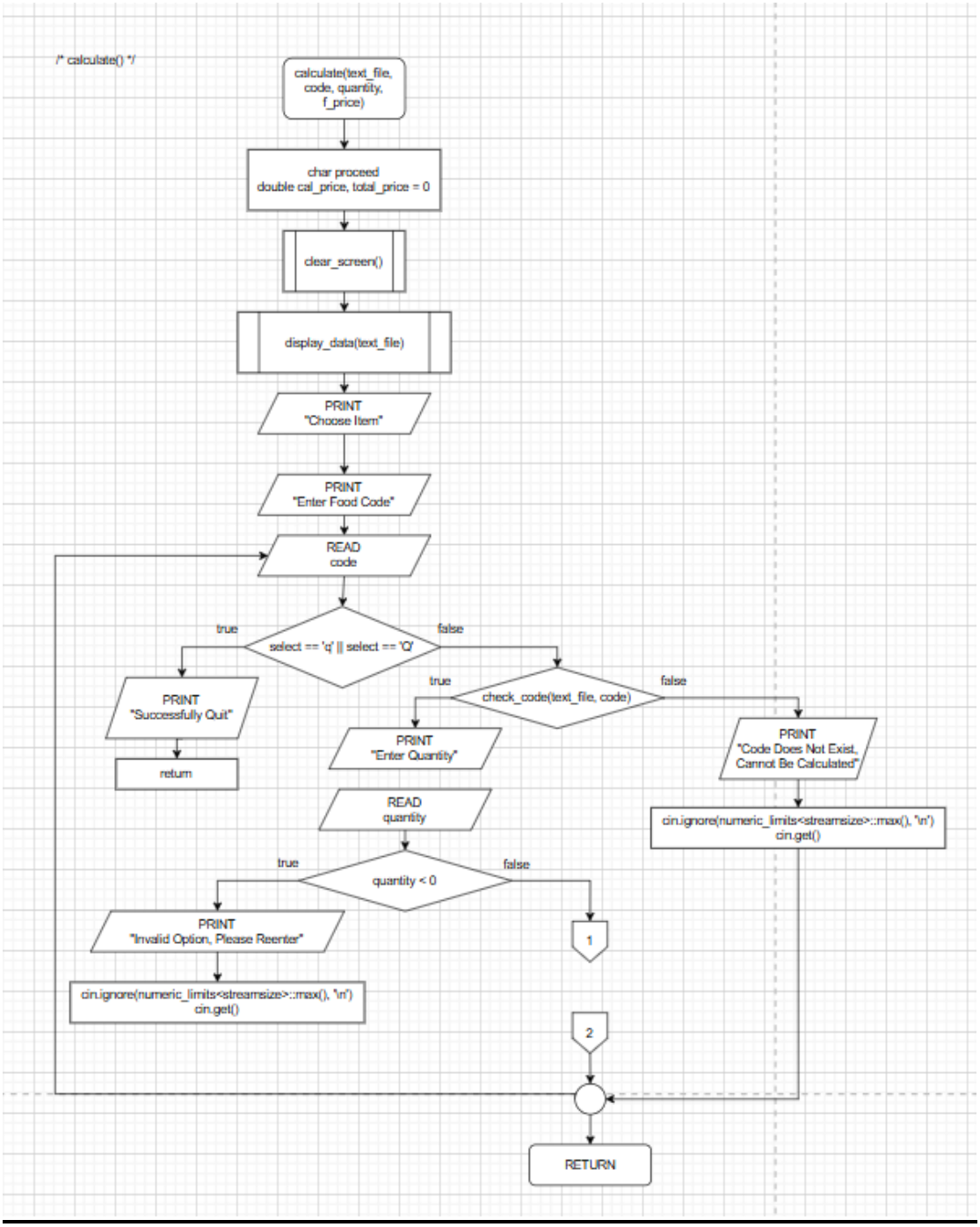
Store

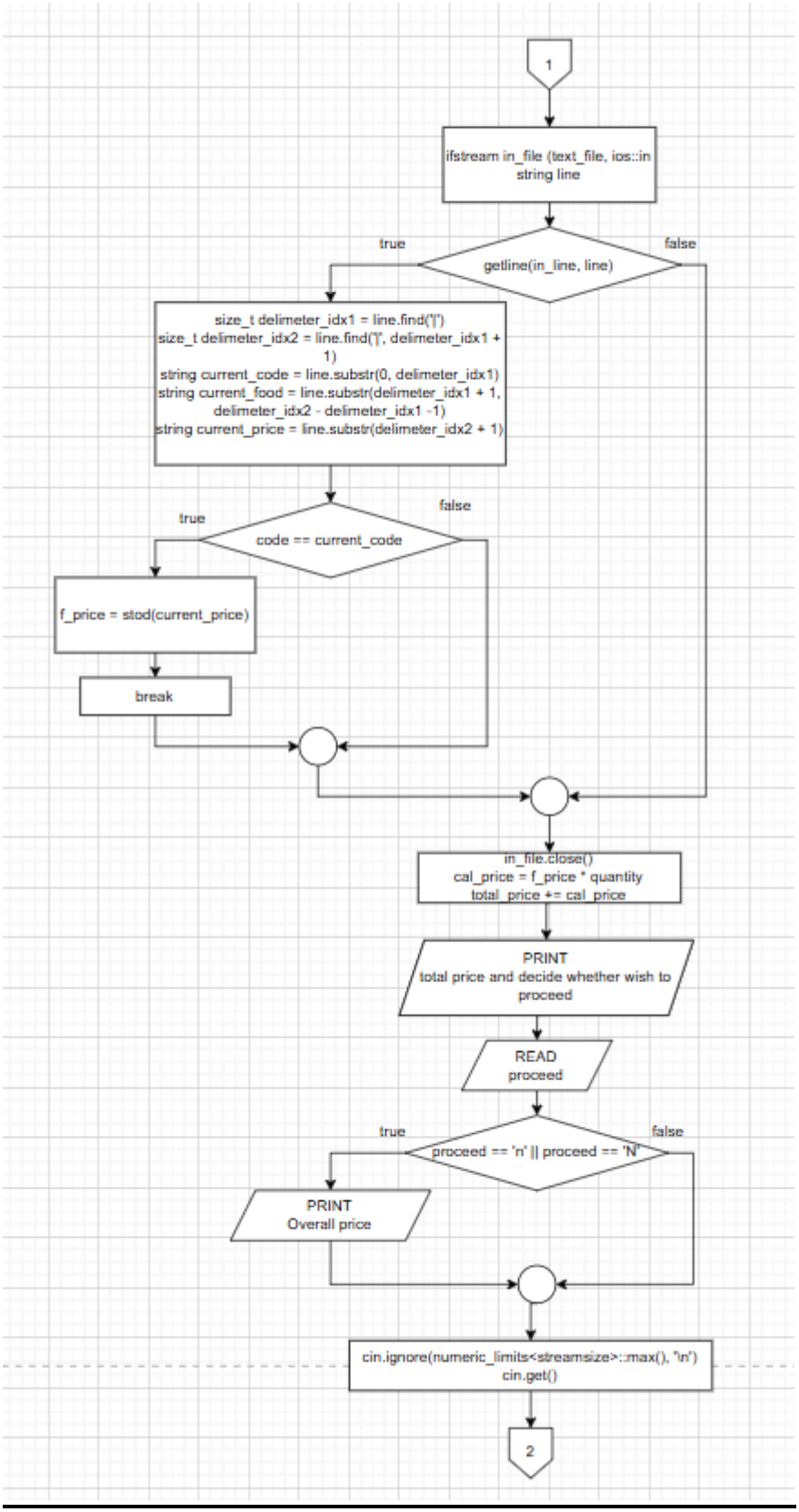

restaurant()
true
false
clear_screen()
PRINT
"Restaurant Details"
ifstream in_file("restaurant_data.txt"), ios::in)
string line
char ch
int i=1
i++
getline(in_file, line)
DISPLAY
i and line
Close File
in_file.close()
PRINT
"Enter Number of Restaurant +Q+ult"
Read
ch
ch>='1' && ch<= '8'
ch>='q' && ch<= 'q'
PRINT
"Successfully Quit"
break
PRINT
"Invalid Input Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
switch(ch)
1
2
3
4
5
6
7
8
default
track("ah_ma.txt", 5, "09:30 AM - 09:00 PM")
track("desa_cik.txt", 2, "24 Hours (Whole Day)")
track("fun_fun.txt", 3, "08:00 AM - 04:00 PM")
track("happy_thai.txt", 5, "11:00 AM - 03:00 PM / 05:00 PM - 10:00 PM")
track("mrs_yau.txt", 7, "11:15 AM - 09:30 PM")
track("mye_mye.txt", 2, "07:00 AM - 04:00 PM")
track("taiwan_dami.txt", 3, "11:30 AM - 09:30 PM")
track("wei_duo.txt", 9, "10:00 AM - 09:30 PM")
PRINT
"Invalid Input Please try Again!"
break
RETURN

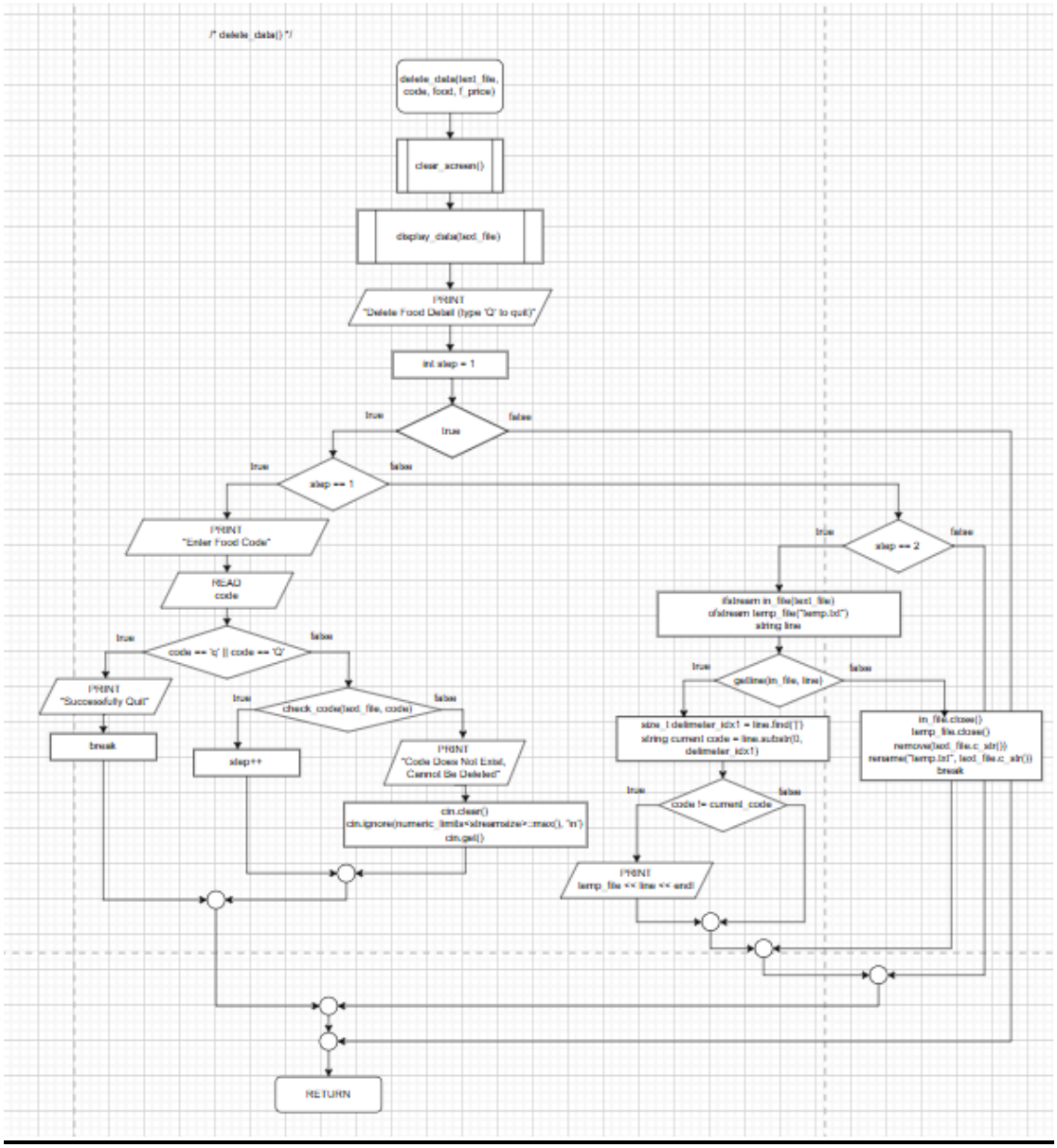

/* cafe() */
cafe()
true
true
false
clear_screen()
PRINT
"Cafe Details"
ifstream in_file("restaurant_data.txt"), loc::in)
string line
char ch
int i=1
i++
getline(in_file, line)
false
true
DISPLAY
i and line
Close File
in_file.close()
PRINT
"Enter Number of Cafe -Q=out"
Read
ch
ch=='1' && ch<= '7'
true
false
switch(ch)
ch=='q' && ch== 'q'
true
PRINT
"Successfully Quit"
break
false
PRINT
"Invalid Input! Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
1
2
3
4
5
6
track("bilguun.txt", 1, "11:00 AM - 11:00 PM")
track("blackboard.txt", 1, "12:00 AM - 09:00 PM")
track("mikuo.txt", 1, "11:00 AM - 10:00 PM")
track("secret.txt", 3, "09:00 AM - 06:30 PM")
track("snowy.txt", 2, "12:00 AM - 09:00 PM")
track("zaba.txt", 4, "10:00 AM - 10:00 PM")
break
7
track("zua.txt", 3, "07:00 AM - 10:40 PM")
default
PRINT
"Invalid Input Please Try Again!"
RETURN

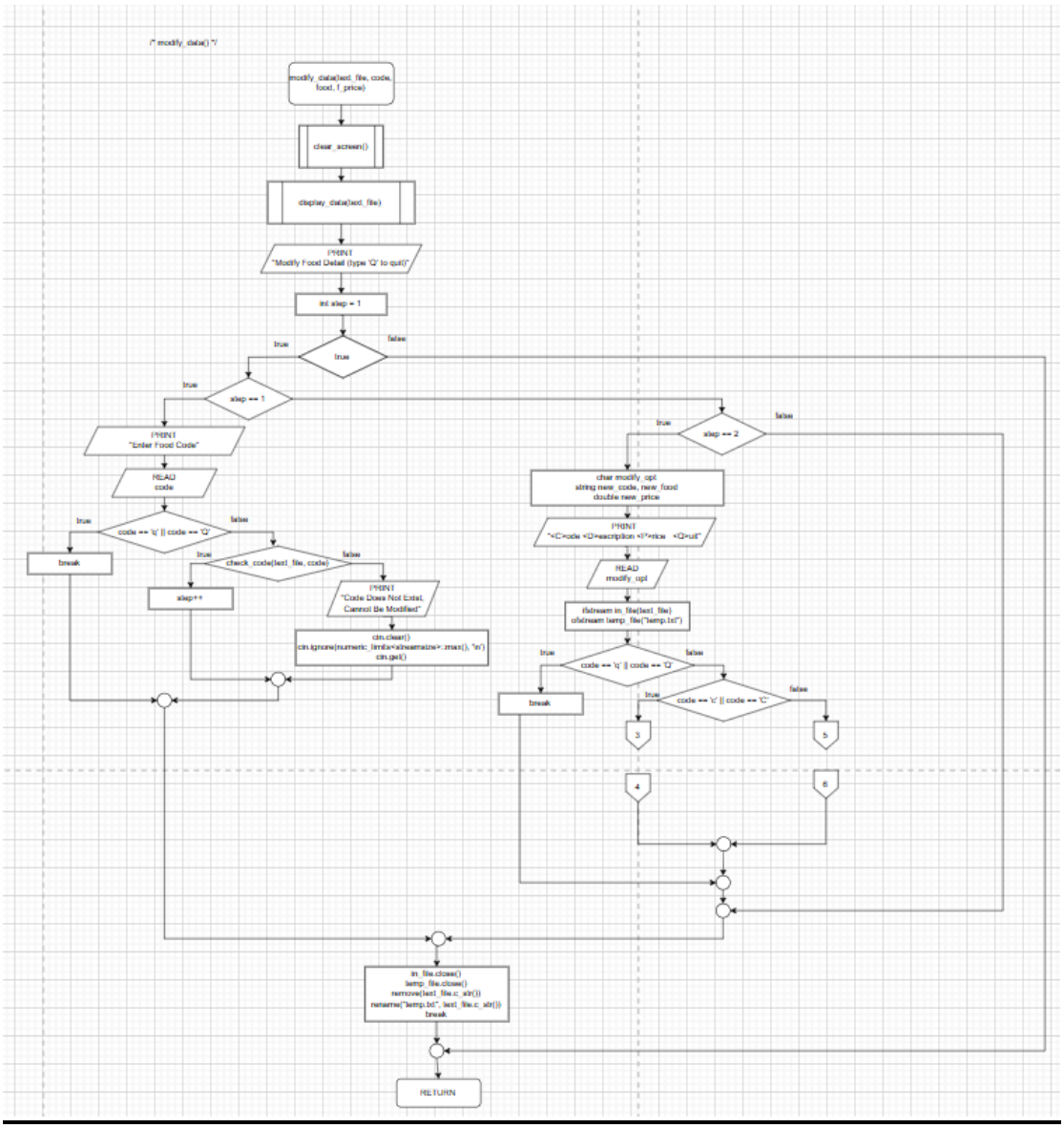

/* fast_food() */
fast_food()
true
false
true
clear_screen()
PRINT
"Fast Food Store Details"
ifstream in_file("fast_food_data.txt"), ios::in)
string line
char ch
int i=1
i++
getline(in_file, line)
false
true
DISPLAY
i and line
Close File
in_file.close()
PRINT
"Enter Number of Fast Food Store <Q=uit"
Read
ch
true
false
ch>='1' && ch<= '4'
ch=='q' && ch== 'q'
true
PRINT
"Successfully Quit"
break
false
PRINT
"Invalid Input! Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
switch(ch)
1
2
3
true
4
default
track("burger.txt", 2, "08:00 AM - 11:00 PM")
track("domino.txt", 8, "10:30 AM - 11:00 PM")
track("kfc.txt", 8, "08:00 AM - 11:00 PM")
track("pizza.txt", 1, "10:00 AM - 11:00 PM")
PRINT
"Invalid Input! Please try Again!"
break
break
break
break
break
RETURN

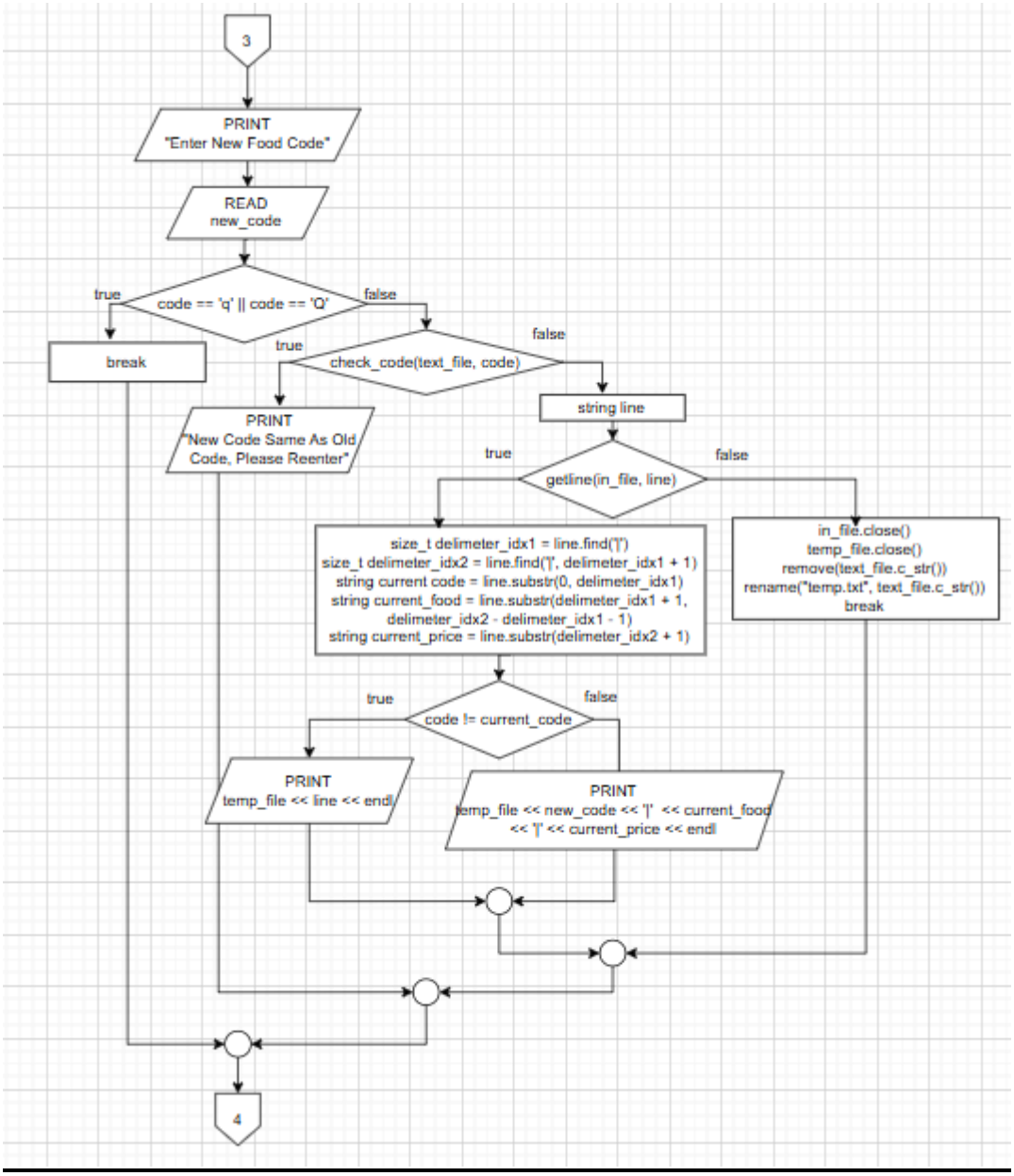

/* track() */
track(text_file, distance, hrs)
double f_price, operate_hrs
string code, food
char select
int quantity
true
false
true
clear_screen()
display_data(text_file)
PRINT
walking distance, operating hours, add, modify, delete, calculate, and quit option
READ
select
select == 'q' || select == 'Q'
true
false
PRINT
"Successfully Quit"
break
select == 'a' || select == 'A'
true
false
add_data(text_file, code, food, f_price)
break
select == 'm' || select == 'M'
true
false
modify_data(text_file, code, food, f_price)
break
select == 'd' || select == 'D'
true
false
delete_data(text_file, code, food, f_price)
break
select == 'c' || select == 'C'
true
false
calculate_data(text_file, code, food, f_price)
break
PRINT
"Invalid Option, I'ts Try Again!"
cin.clear()
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
RETURN

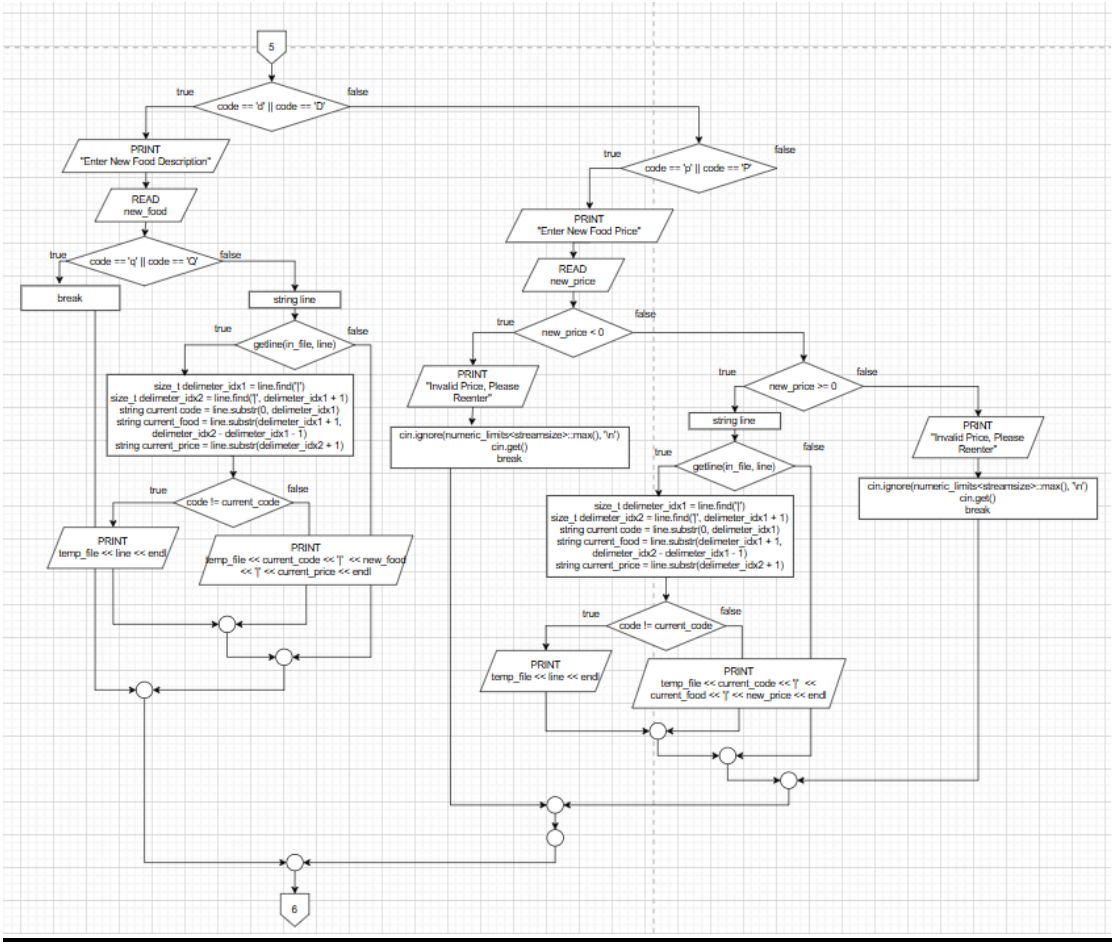

/* calculate() */
calculate(text_file, code, quantity, f_price)
char proceed
double cal_price, total_price = 0
clear_screen()
display_data(text_file)
PRINT "Choose Item"
PRINT "Enter Food Code"
READ code
true
select == 'q' || select == 'Q'
false
PRINT "Successfully Quit"
return
true
check_code(text_file, code)
false
PRINT "Enter Quantity"
PRINT "Code Does Not Exist, Cannot Be Calculated"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
READ quantity
true
quantity < 0
false
PRINT "Invalid Option, Please Reenter"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
1
2
RETURN

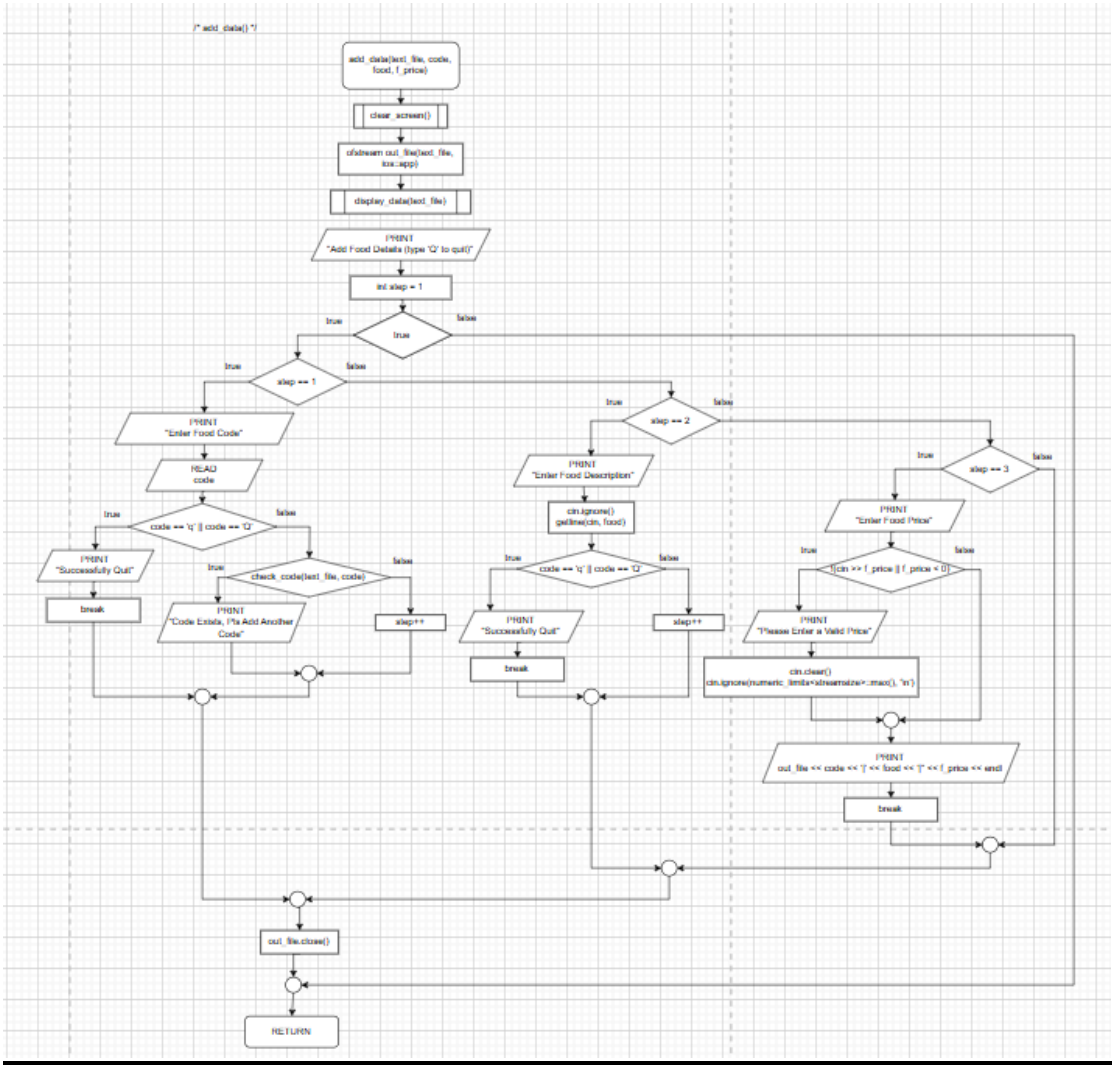

1
ifstream in_file (text_file, ios::in
string line
true
getline(in_line, line)
false
size_t delimeter_idx1 = line.find('|')
size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1)
string current_code = line.substr(0, delimeter_idx1)
string current_food = line.substr(delimeter_idx1 + 1, delimeter_idx2 - delimeter_idx1 -1)
string current_price = line.substr(delimeter_idx2 + 1)
true
code == current_code
false
f_price = stod(current_price)
break
in_file.close()
cal_price = f_price * quantity
total_price += cal_price
PRINT
total price and decide whether wish to proceed
READ
proceed
true
proceed == 'n' || proceed == 'N'
false
PRINT
Overall price
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
2

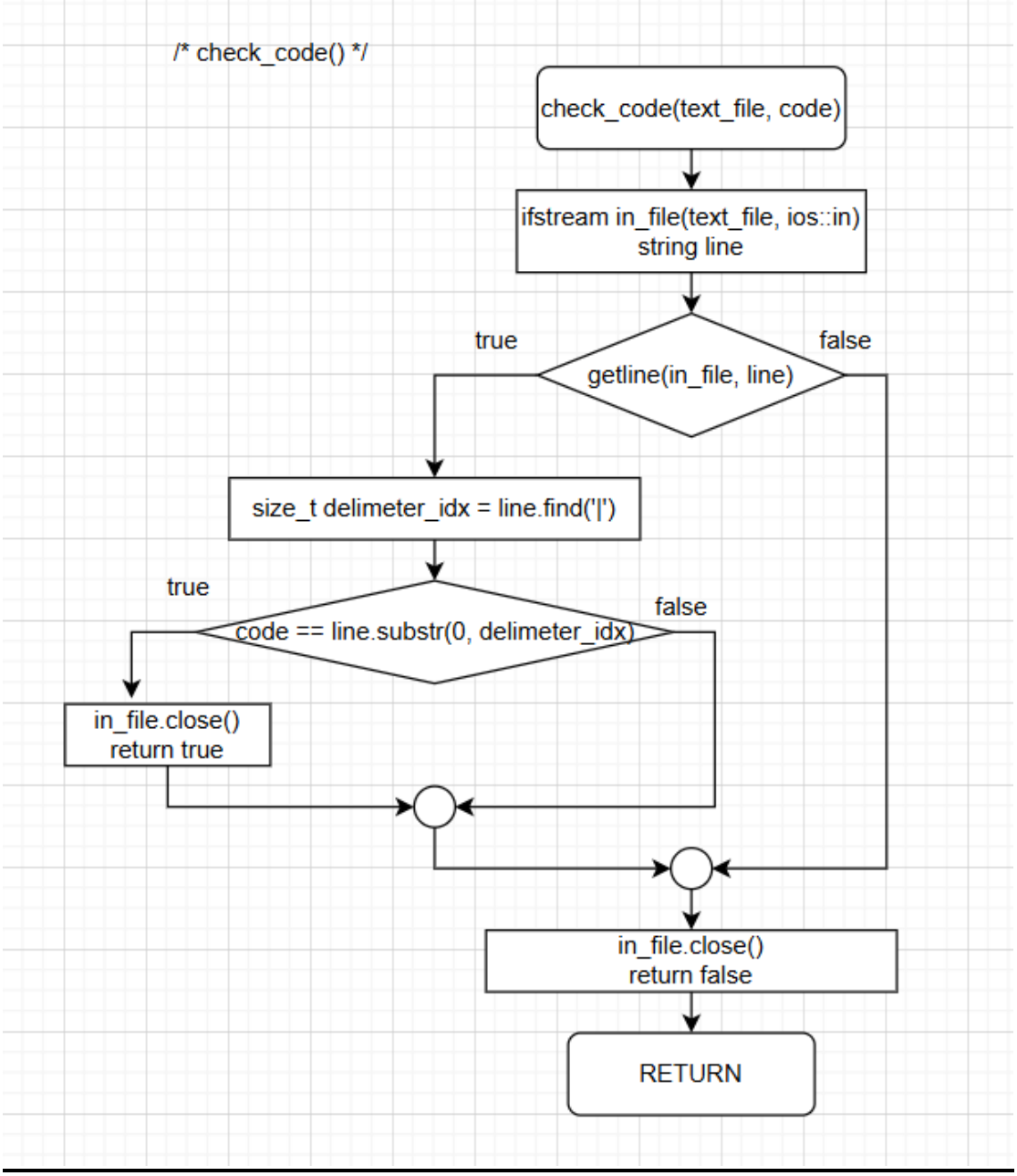

/* delete_data() */
delete_data(text_file, code, food, f_price)
clear_screen()
display_data(text_file)
PRINT
"Delete Food Detail (type 'Q' to quit)"
int step = 1
true
false
step == 1
step == 2
PRINT
"Enter Food Code"
READ
code
code == 'q' || code == 'Q'
PRINT
"Successfully Quit"
break
check_code(text_file, code)
step++
PRINT
"Code Does Not Exist, Cannot Be Deleted"
cin.clear()
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
ifstream in_file(text_file)
ofstream temp_file("temp.txt")
string line
getline(in_file, line)
size_t delimeter_idx1 = line.find('|')
string current_code = line.substr(0, delimeter_idx1)
code != current_code
PRINT
temp_file << line << endl
in_file.close()
temp_file.close()
remove(text_file.c_str())
rename("temp.txt", text_file.c_str())
break
RETURN

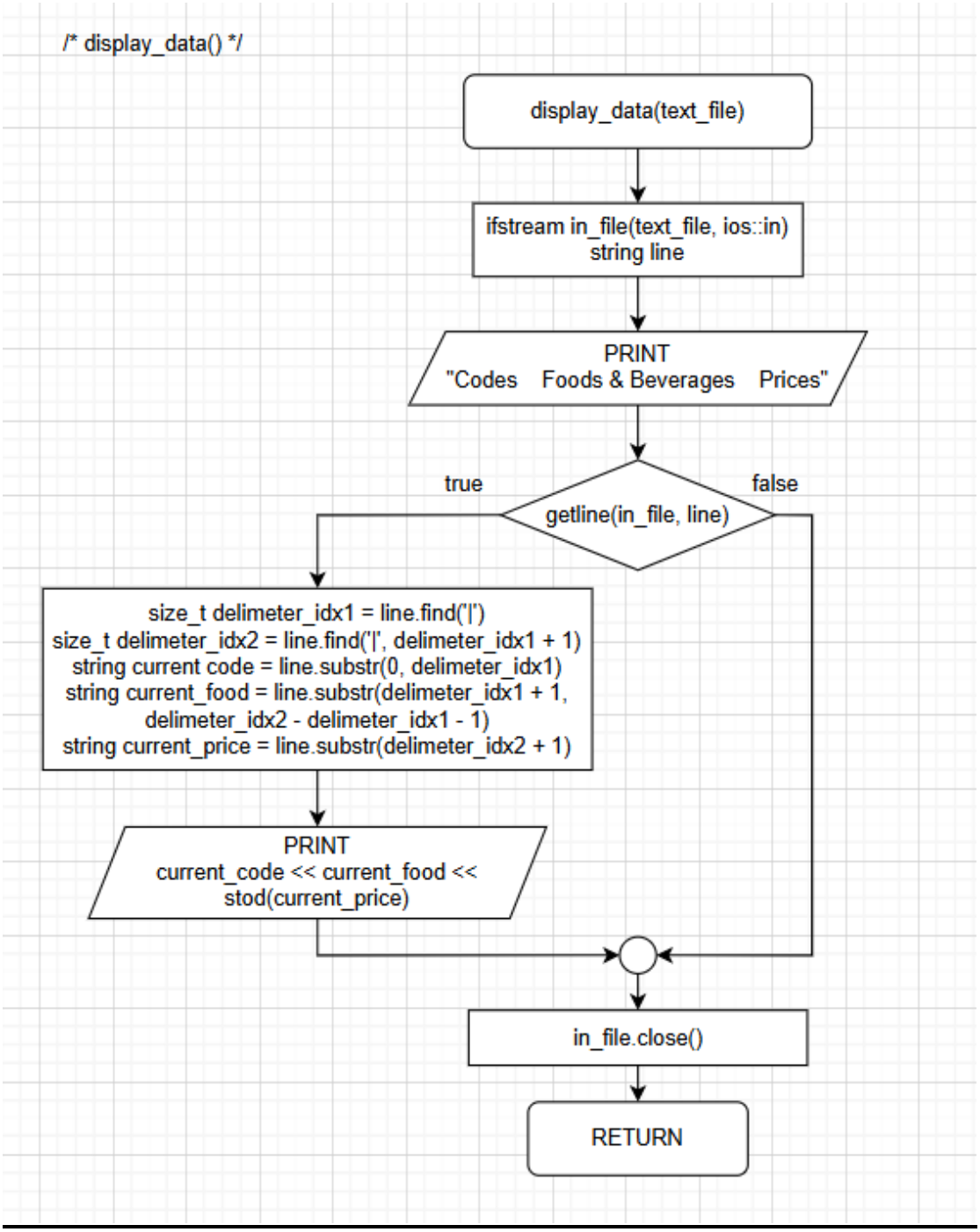

/* modify_data() */
modify_data(text_file, code, food, f_price)
clear_screen()
display_data(text_file)
PRINT
"Modify Food Detail (type 'Q' to quit)"
int step = 1
true
false
step == 1
PRINT
"Enter Food Code"
READ
code
code == 'q' || code == 'Q'
break
check_code(text_file, code)
step++
PRINT
"Code Does Not Exist,
Cannot Be Modified"
cin.clear()
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
step == 2
char modify_opt
string new_code, new_food
double new_price
PRINT
"<C>ode <D>escription <P>rice <Q>uit"
READ
modify_opt
fstream in_file(text_file)
ofstream temp_file("temp.txt")
code == 'q' || code == 'Q'
break
code == 'c' || code == 'C'
3
5
4
6
in_file.close()
temp_file.close()
remove(text_file.c_str())
rename("temp.txt", text_file.c_str())
break
RETURN

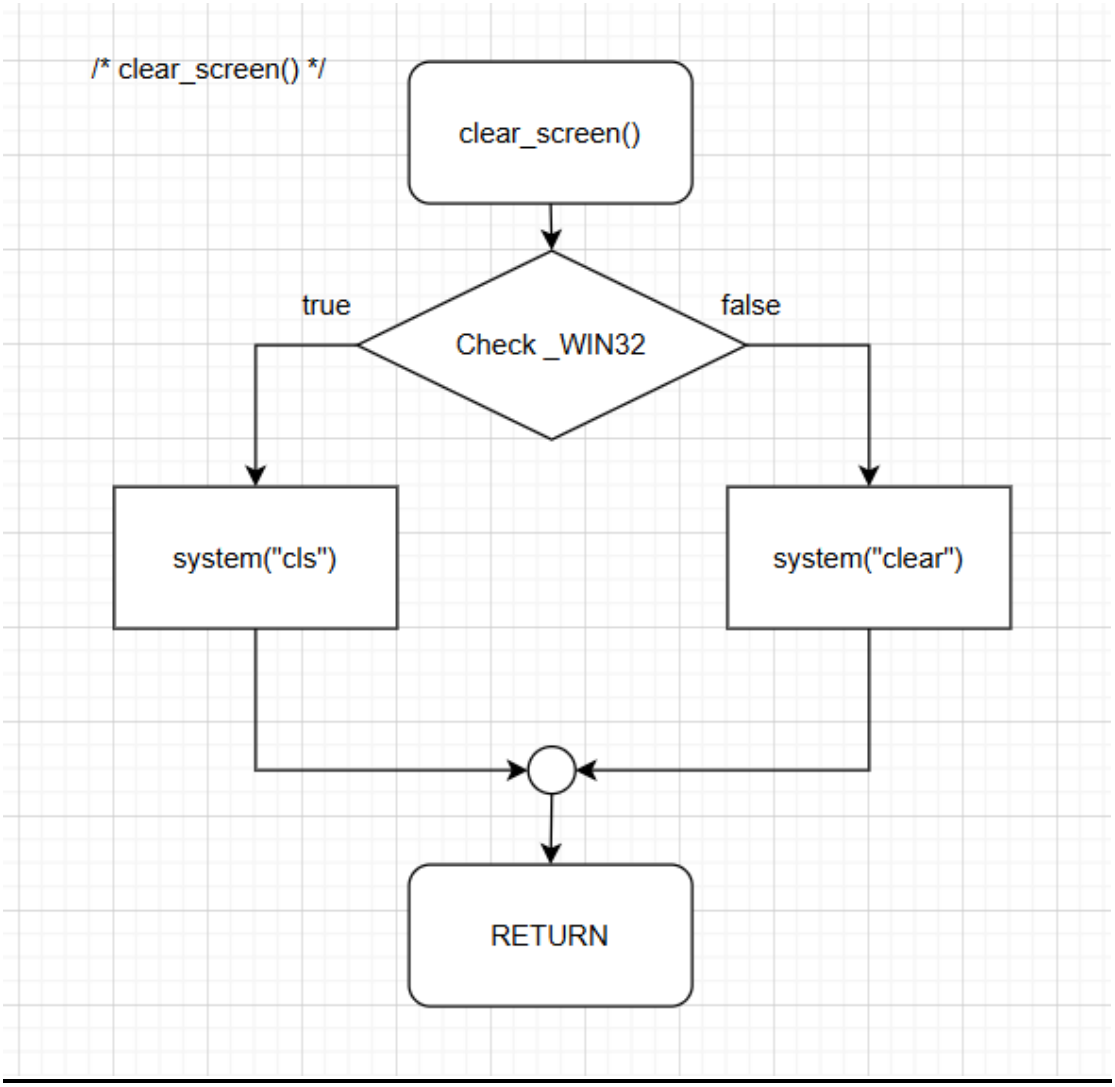

3
PRINT
"Enter New Food Code"
READ
new_code
true
code == 'q' || code == 'Q'
false
break
true
check_code(text_file, code)
false
PRINT
"New Code Same As Old
Code, Please Reenter"
string line
true
getline(in_file, line)
false
size_t delimeter_idx1 = line.find('|')
size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1)
string current_code = line.substr(0, delimeter_idx1)
string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1)
string current_price = line.substr(delimeter_idx2 + 1)
in_file.close()
temp_file.close()
remove(text_file.c_str())
rename("temp.txt", text_file.c_str())
break
true
code != current_code
false
PRINT
temp_file << line << endl
PRINT
temp_file << new_code << '|' << current_food
<< '|' << current_price << endl
4


5
code == 'd' || code == 'D'
true
false
PRINT
"Enter New Food Description"
READ
new_food
code == 'q' || code == 'Q'
true
false
break
string line
getline(in_file, line)
true
false
size_t delimeter_idx1 = line.find('|')
size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1)
string current_code = line.substr(0, delimeter_idx1)
string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1)
string current_price = line.substr(delimeter_idx2 + 1)
code != current_code
true
false
PRINT
temp_file << line << endl
PRINT
temp_file << current_code << '|' << new_food
<< '|' << current_price << endl
code == 'p' || code == 'P'
true
false
PRINT
"Enter New Food Price"
READ
new_price
new_price < 0
true
false
PRINT
"Invalid Price, Please
Reenter"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
break
new_price >= 0
true
false
string line
getline(in_file, line)
true
false
size_t delimeter_idx1 = line.find('|')
size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1)
string current_code = line.substr(0, delimeter_idx1)
string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1)
string current_price = line.substr(delimeter_idx2 + 1)
code != current_code
true
false
PRINT
temp_file << line << endl
PRINT
temp_file << current_code << '|' <<
current_food << '|' << new_price << endl
PRINT
"Invalid Price, Please
Reenter"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
break
6

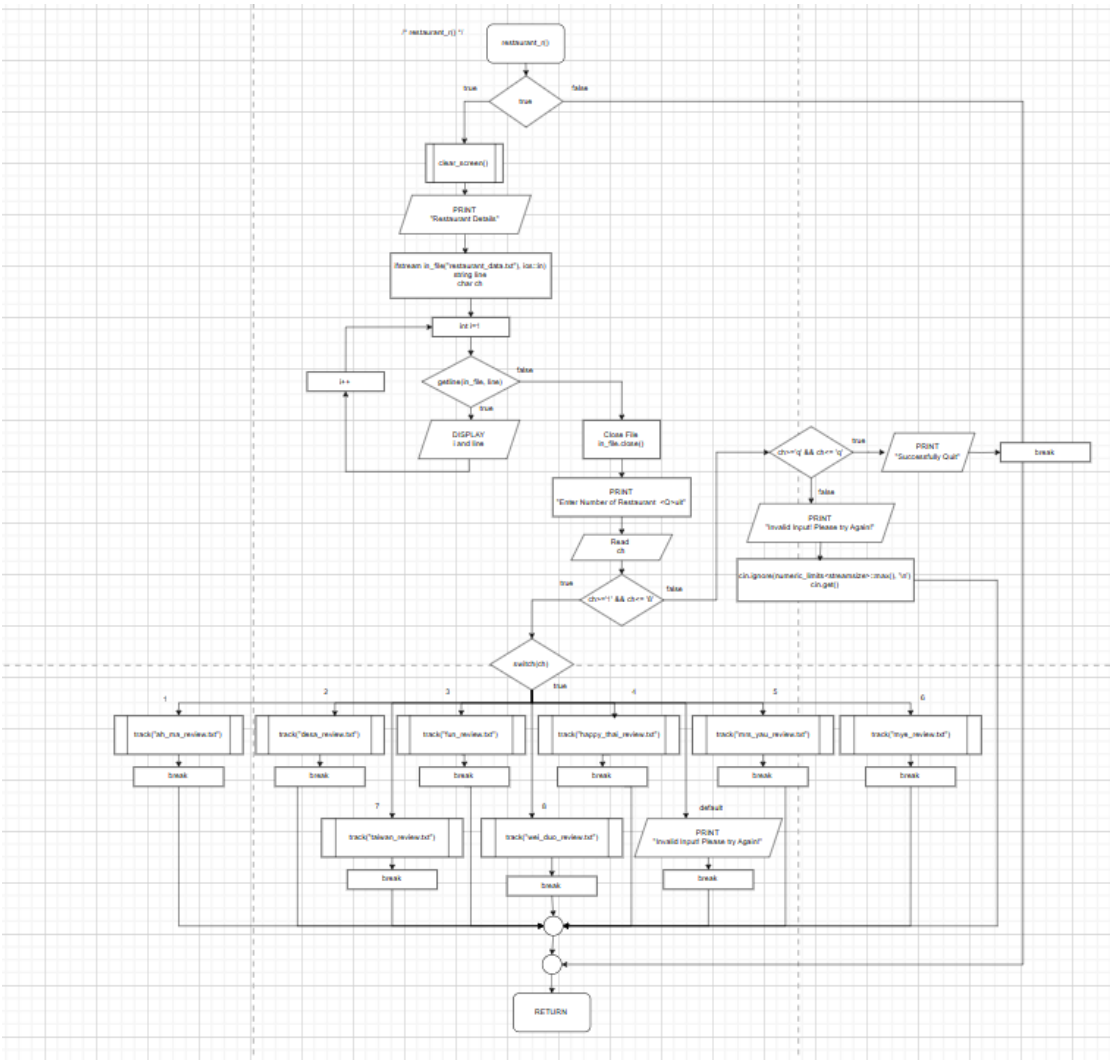

/* add_data() */
add_data(text_file, code, food, f_price)
clear_screen()
ofstream out_file(text_file, ios::app)
display_data(text_file)
PRINT "Add Food Details (type 'Q' to quit)"
int step = 1
true
step == 1
PRINT "Enter Food Code"
READ code
code == 'q' || code == 'Q'
PRINT "Successfully Quit"
break
check_code(text_file, code)
PRINT "Code Exists, Pls Add Another Code"
step++
step == 2
PRINT "Enter Food Description"
cin.ignore()
getline(cin, food)
code == 'q' || code == 'Q'
PRINT "Successfully Quit"
break
step++
step == 3
PRINT "Enter Food Price"
!(cin >> f_price || f_price < 0)
PRINT "Please Enter a Valid Price"
cin.clear()
cin.ignore(numeric_limits<streamsize>::max(), '\n')
PRINT out_file << code << '|' << food << '|' << f_price << endl
break
out_file.close()
RETURN

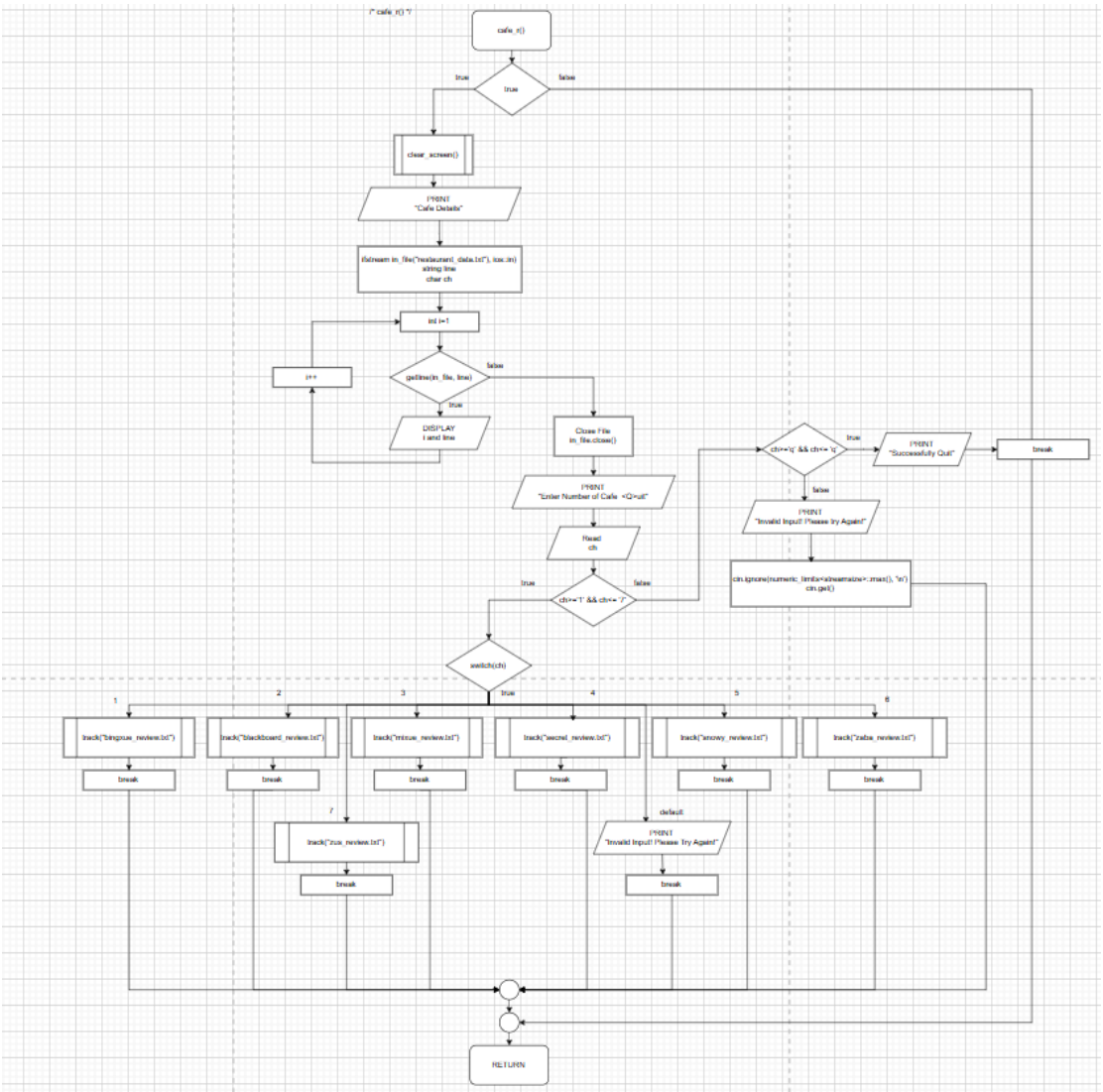

/* check_code() */
check_code(text_file, code)
ifstream in_file(text_file, ios::in)
string line
true
false
getline(in_file, line)
size_t delimeter_idx = line.find('|')
true
false
code == line.substr(0, delimeter_idx)
in_file.close()
return true
in_file.close()
return false
RETURN

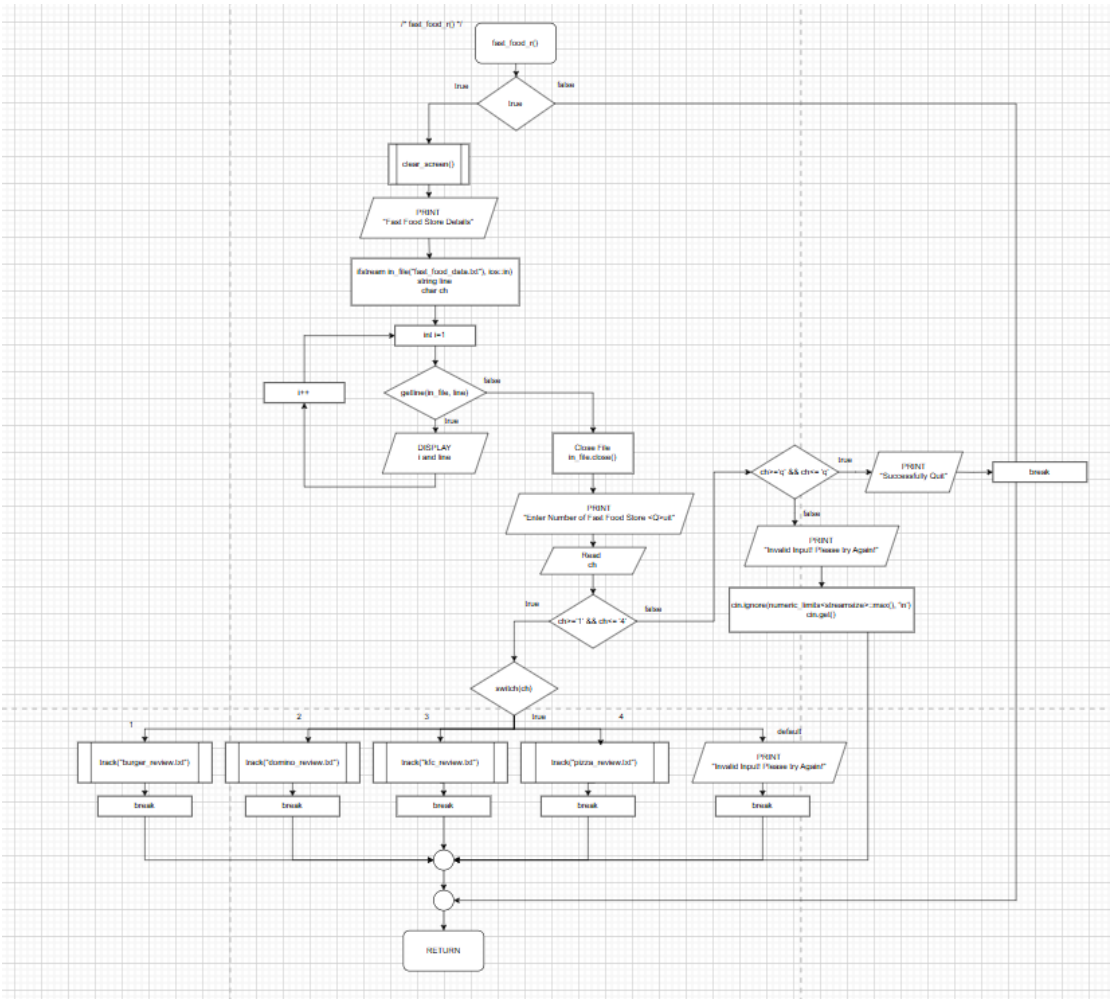

/* display_data() */
display_data(text_file)
ifstream in_file(text_file, ios::in)
string line
PRINT
"Codes Foods & Beverages Prices"
true
false
getline(in_file, line)
size_t delimeter_idx1 = line.find('|')
size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1)
string current code = line.substr(0, delimeter_idx1)
string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1)
string current_price = line.substr(delimeter_idx2 + 1)
PRINT
current_code << current_food <<
stod(current_price)
in_file.close()
RETURN

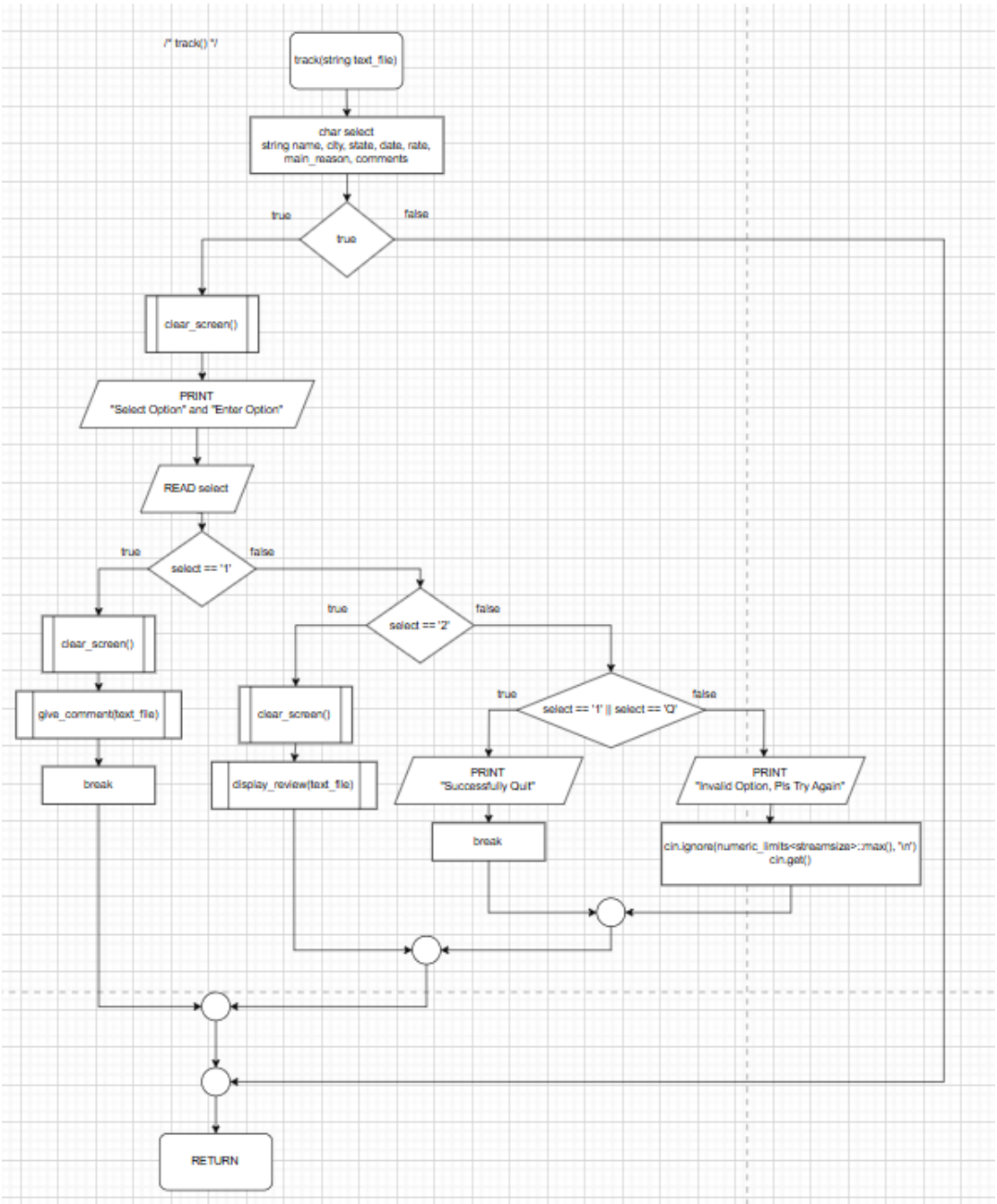

/* clear_screen() */
clear_screen()
true
Check _WIN32
false
system("cls")
system("clear")
RETURN

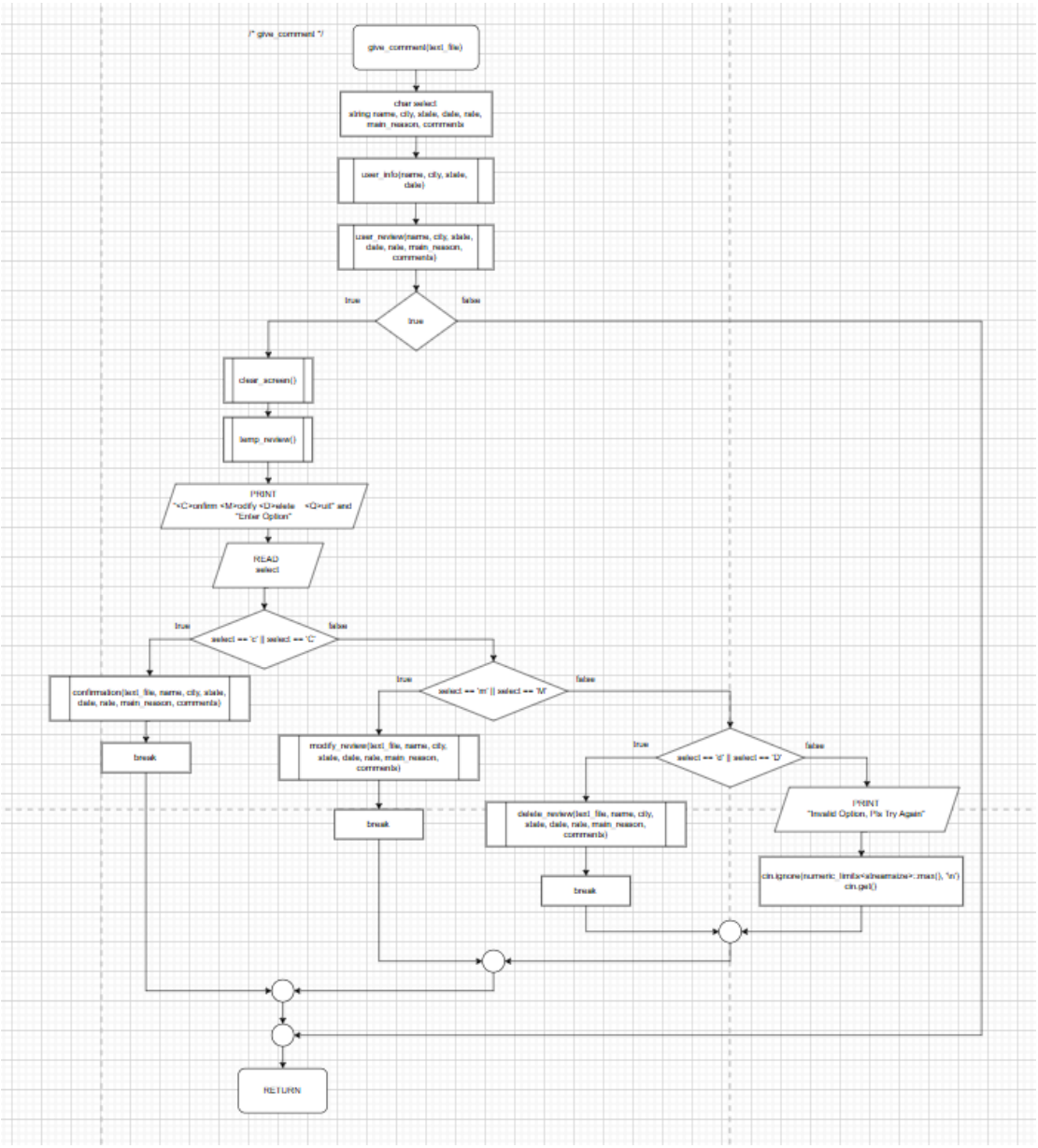
Review

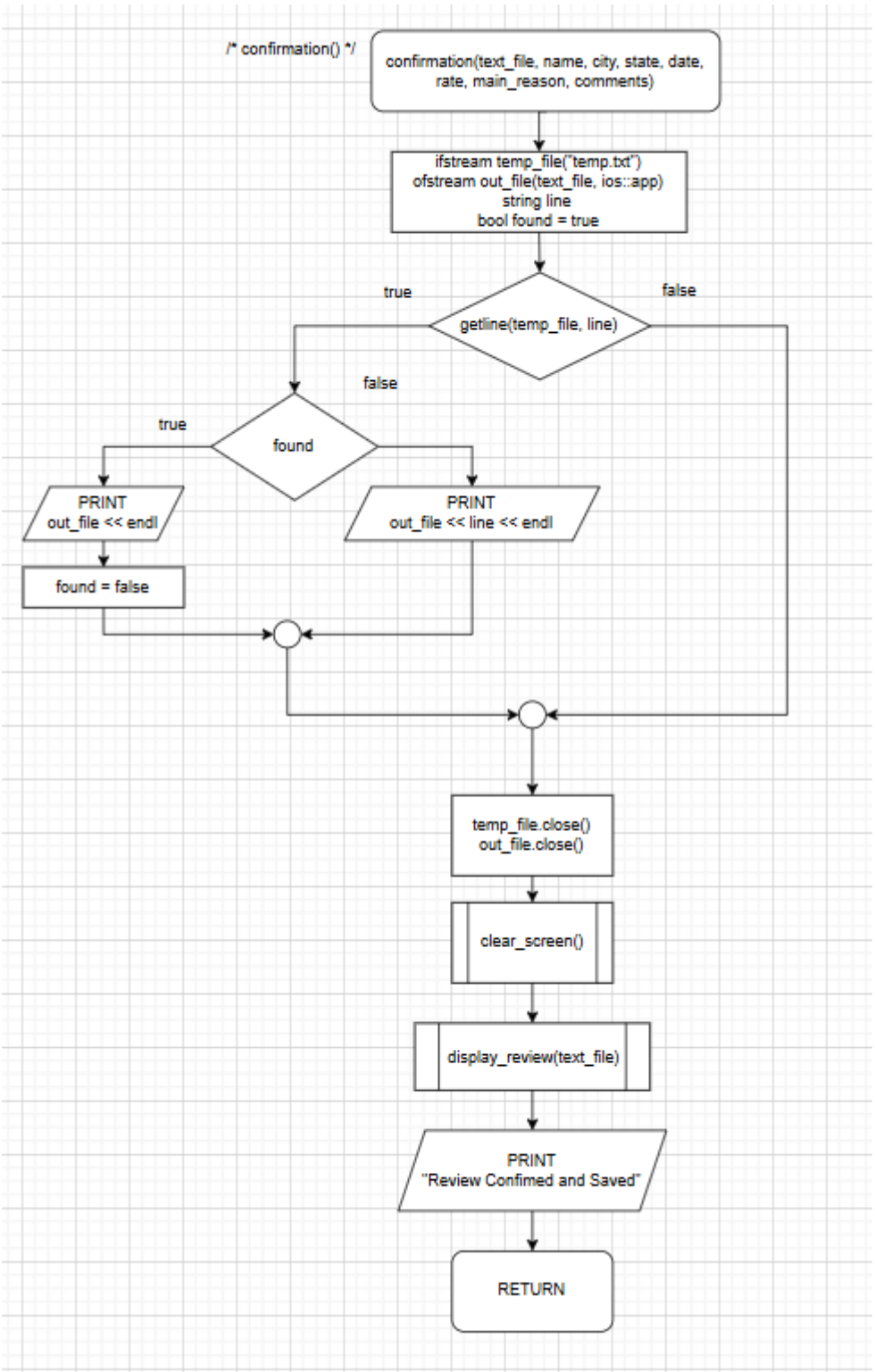

restaurant_r()
true
clear_screen()
PRINT
"Restaurant Details"
DISPLAY
i and line
Close File
in_file.close()
PRINT
"Enter Number of Restaurant <Q>uit"
Read
ch
PRINT
"Successfully Quit"
break
PRINT
"Invalid Input Please try Again!"
switch(ch)
track("ah_ma_review.txt")
track("dasa_review.txt")
track("fun_review.txt")
track("happy_thai_review.txt")
track("mm_you_review.txt")
track("mye_review.txt")
track("taiwan_review.txt")
track("wei_duo_review.txt")
default
PRINT
"Invalid Input Please try Again!"
RETURN

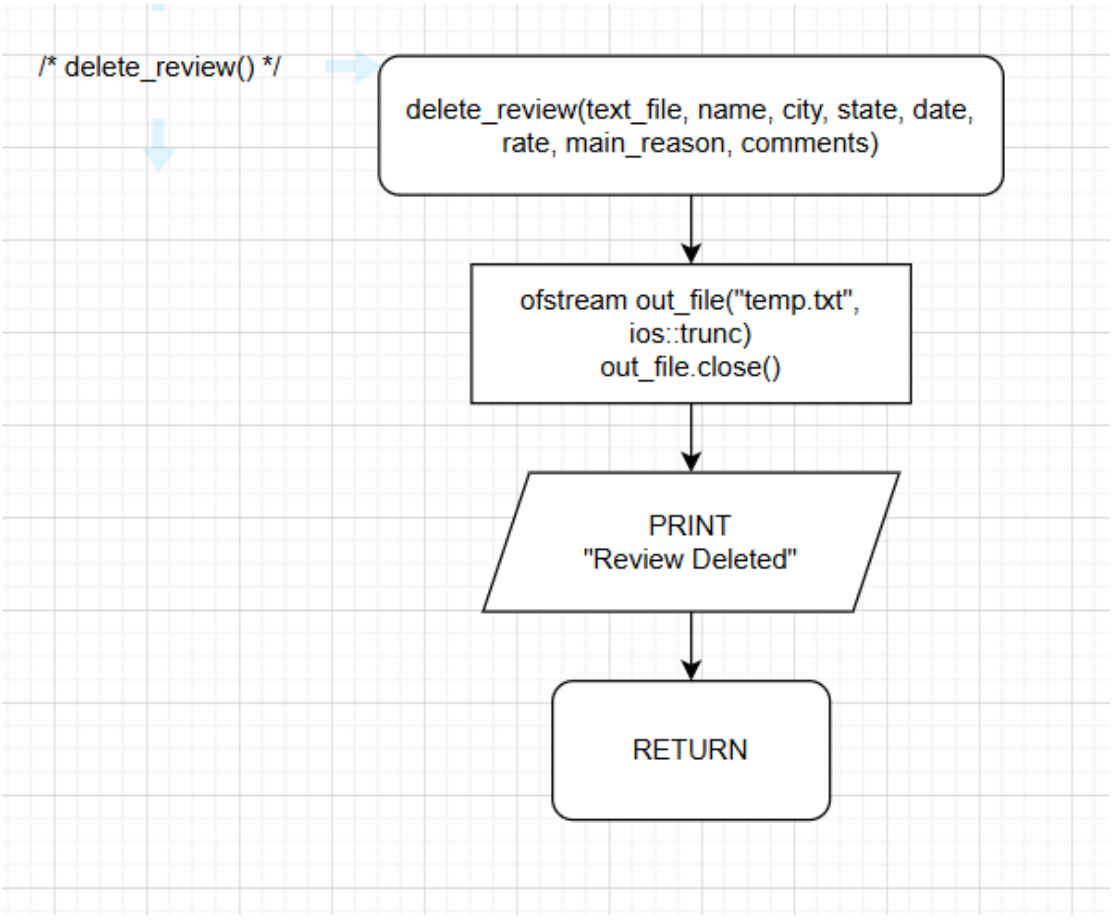

/* cafe_r() */
cafe_r()
true
true
false
clear_screen()
PRINT
"Cafe Details"
ifstream in_file("restaurant_data.txt"), inac.in)
string line
char ch
int i=1
i++
getline(in_file, line)
false
true
DISPLAY
i and line
Close File
in_file.close()
PRINT
"Enter Number of Cafe +Q=out"
Read
ch
ch>='1' && ch<= '7'
true
false
switch(ch)
true
1
2
3
4
5
6
track("bingzuw_reviews.txt")
track("blackboard_reviews.txt")
track("mixue_review.txt")
track("secret_review.txt")
track("snowy_review.txt")
track("zaiba_review.txt")
break
7
track("zuu_reviews.txt")
default
PRINT
"Invalid Input! Please Try Again!"
RETURN
ch>='q' && ch<= 'q'
true
PRINT
"Successfully Quit"
false
PRINT
"Invalid Input! Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()

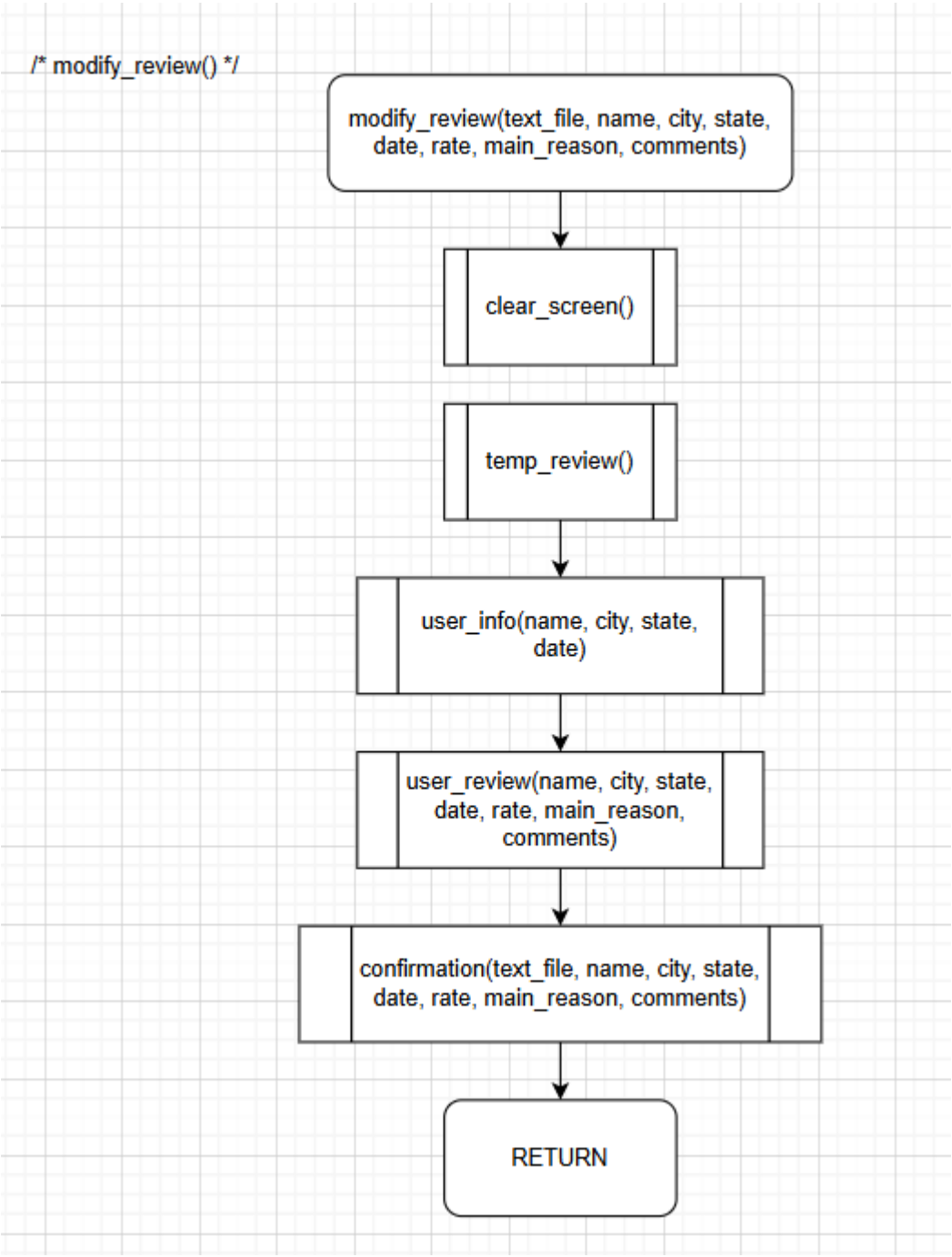

/* fast_food_r() */
fast_food_r()
true
false
true
clear_screen()
PRINT
"Fast Food Store Details"
ifstream in_file("fast_food_data.txt"), ios::in)
string line
char ch
int i=1
i++
getline(in_file, line)
false
true
DISPLAY
i and line
Close File
in_file.close()
PRINT
"Enter Number of Fast Food Store <Q=uit"
Read
ch
true
false
ch>='1' && ch<= '4'
ch>='q' && ch<= 'q'
true
PRINT
"Successfully Quit"
break
false
PRINT
"Invalid Input! Please try Again!"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
switch(ch)
1
2
3
4
true
default
track("burger_review.txt")
track("domino_review.txt")
track("kfc_review.txt")
track("pizza_review.txt")
PRINT
"Invalid Input! Please try Again!"
break
break
break
break
break
RETURN

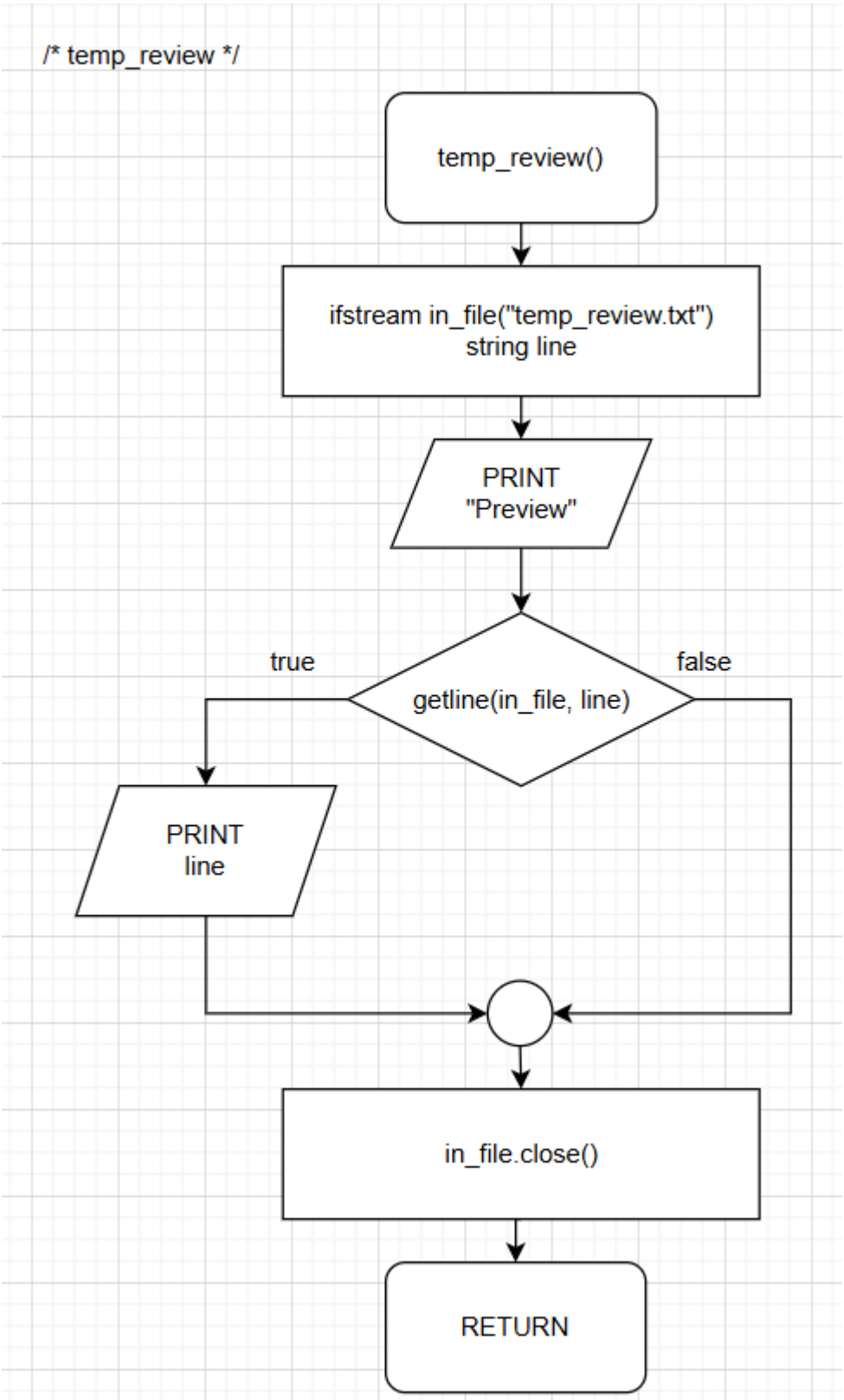

/* track() */
track(string text_file)
char select
string name, city, state, date, rate,
main_reason, comments
true
false
true
clear_screen()
PRINT
"Select Option" and "Enter Option"
READ select
true
false
select == '1'
clear_screen()
give_comment(text_file)
break
true
false
select == '2'
clear_screen()
display_review(text_file)
true
false
select == '1' || select == 'Q'
PRINT
"Successfully Quit"
break
PRINT
"Invalid Option, Pls Try Again"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
RETURN

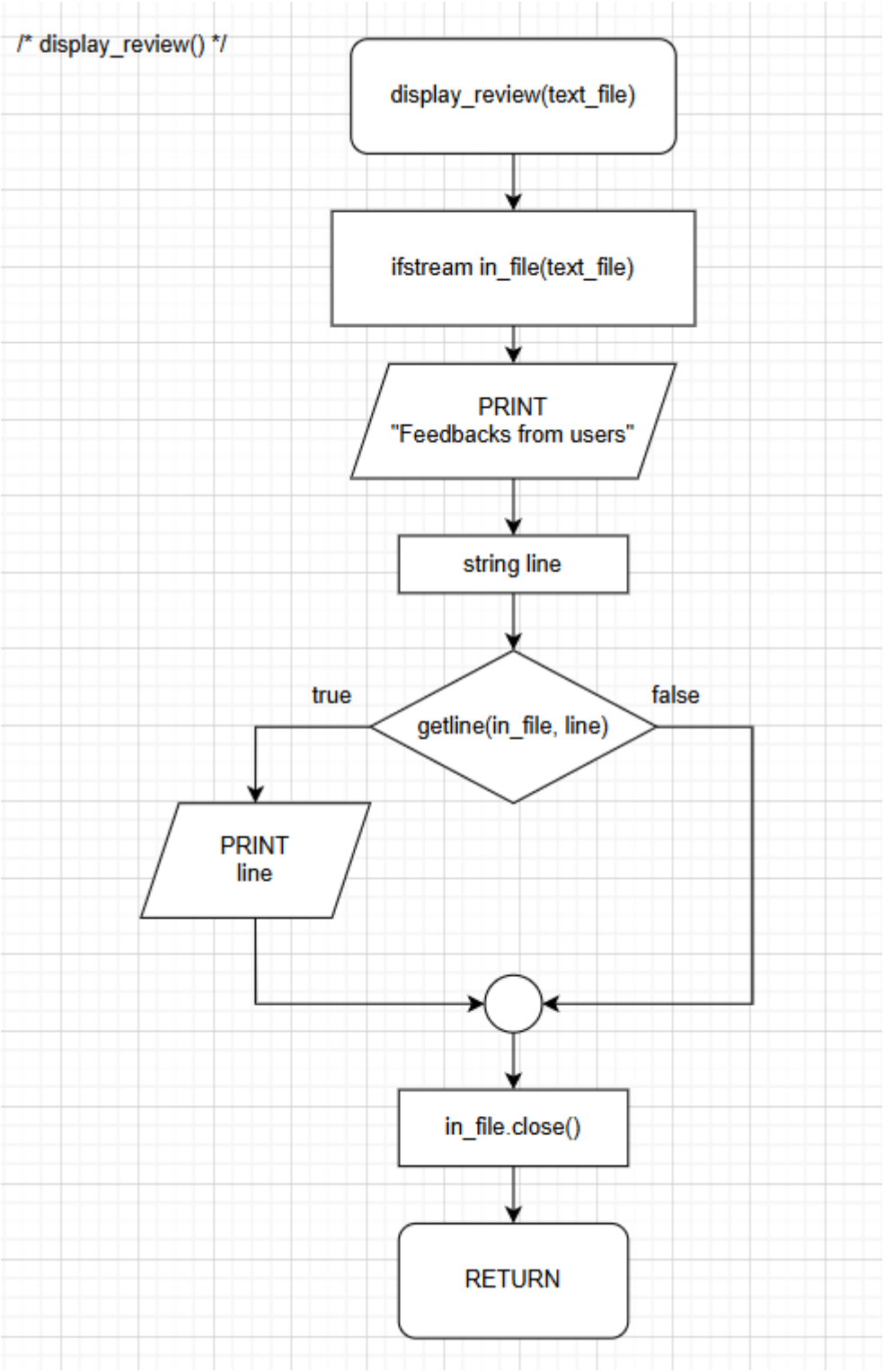

/* give_comment */
give_comment(acct_file)
char select
string name, city, state, date, rate,
main_reason, comments
user_info(name, city, state,
date)
user_review(name, city, state,
date, rate, main_reason,
comments)
true
false
true
clear_screen()
temp_review()
PRINT
"<C>onfirm <M>odify <D>elete <Q>uit" and
"Enter Option"
READ
select
true
false
select == 'c' || select == 'C'
confirmation(acct_file, name, city, state,
date, rate, main_reason, comments)
break
true
false
select == 'm' || select == 'M'
modify_review(acct_file, name, city,
state, date, rate, main_reason,
comments)
break
true
false
select == 'd' || select == 'D'
delete_review(acct_file, name, city,
state, date, rate, main_reason,
comments)
break
PRINT
"Invalid Option, Pls Try Again"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
cin.get()
RETURN

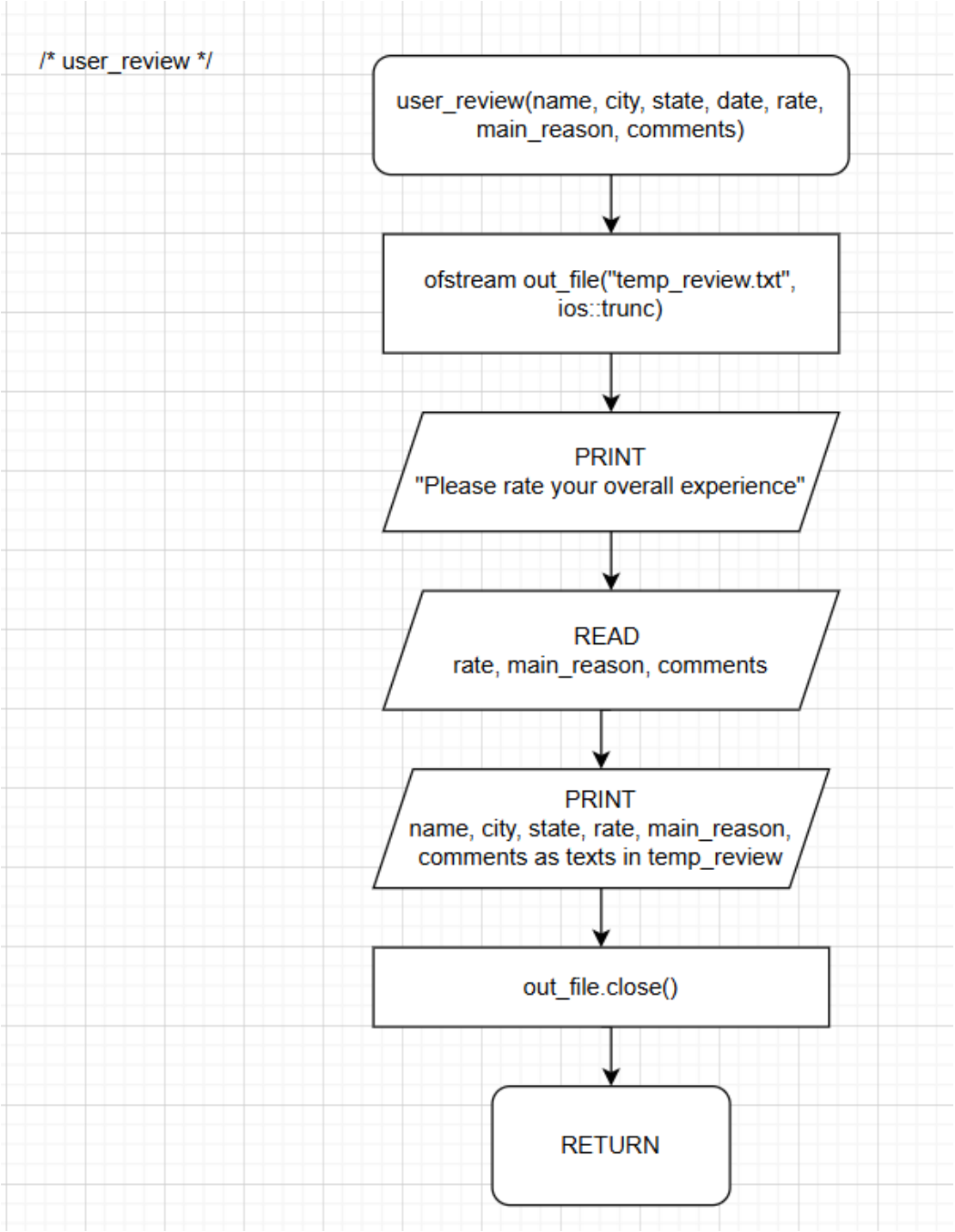

/* confirmation() */
confirmation(text_file, name, city, state, date, rate, main_reason, comments)
ifstream temp_file("temp.txt")
ofstream out_file(text_file, ios::app)
string line
bool found = true
true
false
getline(temp_file, line)
false
true
found
PRINT
out_file << endl
found = false
PRINT
out_file << line << endl
temp_file.close()
out_file.close()
clear_screen()
display_review(text_file)
PRINT
"Review Confimed and Saved"
RETURN

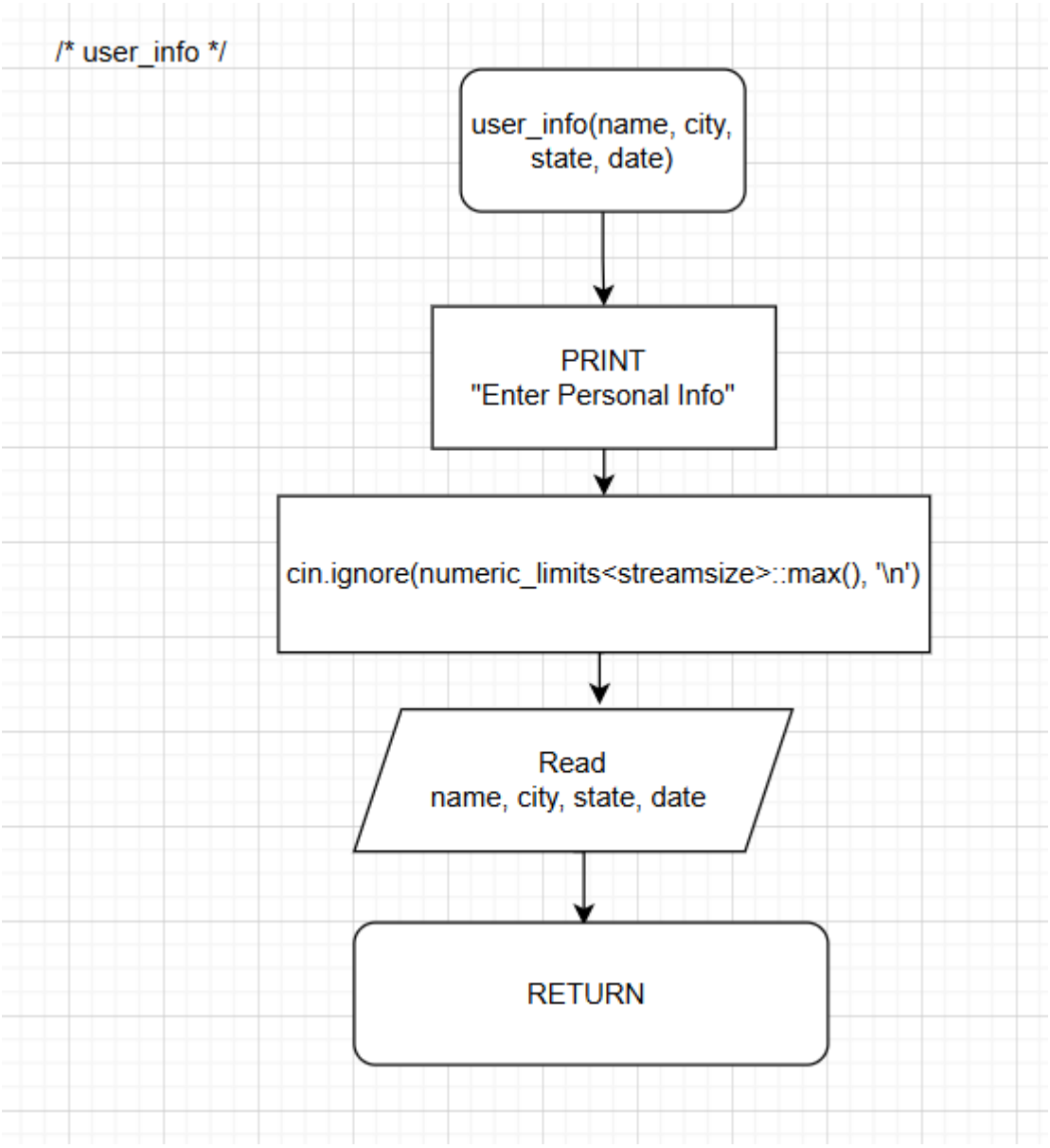

/* delete_review() */
delete_review(text_file, name, city, state, date, rate, main_reason, comments)
ofstream out_file("temp.txt", ios::trunc)
out_file.close()
PRINT
"Review Deleted"
RETURN

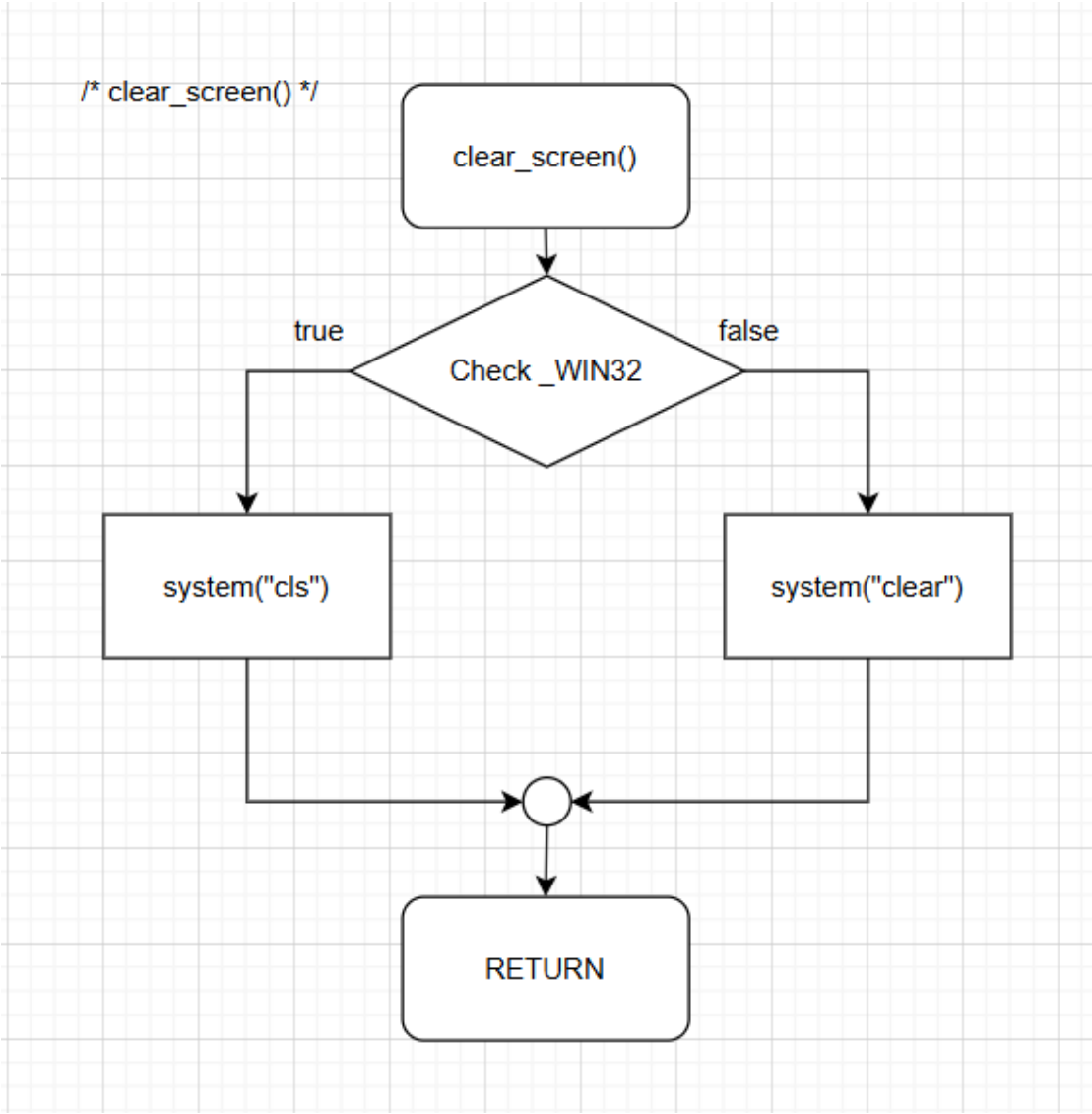

/* modify_review() */
modify_review(text_file, name, city, state, date, rate, main_reason, comments)
clear_screen()
temp_review()
user_info(name, city, state, date)
user_review(name, city, state, date, rate, main_reason, comments)
confirmation(text_file, name, city, state, date, rate, main_reason, comments)
RETURN


/* temp_review */
temp_review()
ifstream in_file("temp_review.txt")
string line
PRINT
"Preview"
true
false
getline(in_file, line)
PRINT
line
in_file.close()
RETURN


/* display_review() */
display_review(text_file)
ifstream in_file(text_file)
PRINT
"Feedbacks from users"
string line
true
getline(in_file, line)
false
PRINT
line
in_file.close()
RETURN


/* user_review */
user_review(name, city, state, date, rate, main_reason, comments)
ofstream out_file("temp_review.txt", ios::trunc)
PRINT "Please rate your overall experience"
READ rate, main_reason, comments
PRINT name, city, state, rate, main_reason, comments as texts in temp_review
out_file.close()
RETURN


/* user_info */
user_info(name, city, state, date)
PRINT "Enter Personal Info"
cin.ignore(numeric_limits<streamsize>::max(), '\n')
Read name, city, state, date
RETURN


/* clear_screen() */
clear_screen()
true
Check _WIN32
false
system("cls")
system("clear")
RETURN

2.0 C++ program

Main_menu

```
/*File name: main_menu.cpp
Tool (IDE) : VS CODE */



// Import food_track Module
#include "food_track.h"
#include "review.h"
FOOD_TRACK ft;

#include <iostream>
#include <cstdlib> // system
#include <string>
#include <limits> // cin.ignore
using namespace std;



// Clear The Screen Before Displaying Data
void clear_screen() {
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}

void instruct() {
    cout << string(60, '-') << endl;
        cout << string(12, ' ') << "Food Stalls Tracking & Review System" << endl;
```

```
        cout << string(60, '-') << endl;
        cout << "<1> Food Stalls Tracking" << endl;
        cout << "<2> Review" << endl;
        cout << "<Q>uit" << endl;
        cout << string(60, '-') << endl;
}

bool is_digit(char Opt) {
    return Opt >= '1' && Opt <= '2';
}

void select_option(char &Opt, bool &loop) {
    cout << "Select Option <Q>uit >> ";
    cin >> Opt;

    if (is_digit(Opt)) {
        switch (Opt) {
            case '1':
                // Import choose_option() Function From food_track.h File
                ft.choose_option();
                break;
            case '2':
                // Import choose_option_r() Function From food_track.h File
                ft.choose_option_r();
                break;
            default:
                cout << "--------------Invalid Input! Please Try Again!--------------" <<
endl;
                break;
```

```
        }
    } else if (Opt == 'q' || Opt == 'Q') {
        loop = false;
        cout << "----------------------Successfully Quit----------------------" << endl;
    } else {
        cout << "--------------Invalid Input! Please Try Again!--------------" << endl;
    }

    // Handle Buffer
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    // Read Single Character (including whitespace) ---> input()
    cin.get();
}

int main() {
    bool loop = true;
    while (loop) {
        char Opt;

        clear_screen();
        instruct();
        select_option(Opt, loop);
    }

    return 0;
}
```

```
/* File name: food_track.cpp */

#include "food_track.h"
#include "store.h"
#include "review.h"
STORE st;
REVIEW rw;

#include <iostream>
#include <string>
#include <iomanip> // system, setw()
#include <limits>   // cin.ignore
#include <fstream> //ifstream, ofstream
using namespace std;


void FOOD_TRACK::clear_screen() {
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}

void FOOD_TRACK::instruct() {
    cout << string(60, '-') << endl;
    cout << string(23, ' ') << "Types of Store" << endl;
    cout << string(60, '-') << endl;
    cout << "<1> Restaurant" << endl;
```

```
    cout << "<2> Cafe" << endl;
    cout << "<3> Fast Food Store" << endl;
    cout << "<Q>uit" << endl;
    cout << string(60, '-') << endl;
}

void FOOD_TRACK::choose_option() {
    while(true) {
        clear_screen();
        instruct();

        char Opt;
        cout << "Enter Option <Q>uit >> ";
        cin >> Opt;
        if (Opt >= '1' && Opt <= '3') {
            switch (Opt) {
                case '1':
                    st.restaurant();
                    break;
                case '2':
                    st.cafe();
                    break;
                case '3':
                    st.fast_food();
                    break;
                default:
                    cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
```

```
                break;
            }
        } else if (Opt == 'q' || Opt == 'Q') {
            cout << "----------Successfully Quit----------" << endl;
            break;

        } else {
            cout << "----------Invalid Input! Please Try Again!----------" << endl;
              cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignore incorrect
input
            cin.get(); // Read single character (including whitespace) ---> input()
        }
    }
}

void FOOD_TRACK::choose_option_r() {
    while(true) {
        clear_screen();
        instruct();

        char Opt;
        cout << "Enter Option <Q>uit >> ";
        cin >> Opt;
        if (Opt >= '1' && Opt <= '3') {
            switch (Opt) {
                case '1':
                    rw.restaurant_r();
                    break;
                case '2':
```

```
                rw.cafe_r();
                break;
            case '3':
                rw.fast_food_r();
                break;
            default:
                cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                break;
        }

    } else if (Opt == 'q' || Opt == 'Q') {
        cout << "----------------------Successfully Quit ----------------------" << endl;
        break;

    } else {
        cout << "--------------Invalid Input! Please Try Again!--------------" << endl;
          cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignore incorrect
input
        cin.get(); // Read single character (including whitespace) ---> input()
    }
  }
}
```

Food_track.h

```
class FOOD_TRACK {
    public:
    // Data Management Function
    void instruct();

    // Core Functions
    void choose_option();
    void choose_option_r();

    // Clear All Data
    void clear_screen();
};
```

Store

```
/* File name: store.cpp */

#include "food_track.h"
#include "store.h"

#include <iostream>
#include <string>
#include <iomanip>
#include <limits>
#include <fstream>
using namespace std;



void STORE::clear_screen() {
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}

void STORE::display_data (string text_file) {
    ifstream in_file(text_file, ios::in);
    string line;

    cout << string(60, '-') << endl;
    cout << "Codes" << setw(21) << "Foods & Beverages" << setw(34) << "Prices" <<
endl;
```

```
    cout << string(60, '-') << endl;

    while (getline(in_file, line)) {
        size_t delimeter_idx1 = line.find('|');
        size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1);
        string current_code = line.substr(0, delimeter_idx1);
          string current_food = line.substr(delimeter_idx1 + 1, delimeter_idx2 -
delimeter_idx1 - 1);
        string current_price = line.substr(delimeter_idx2 + 1);

         cout << left << setw(9) << current_code << setw(41) << current_food <<
setw(10)
        << right << fixed << setprecision(2) << stod(current_price) << endl;
    }

    in_file.close();
}

bool STORE::check_code(string text_file, string code) {
    ifstream in_file(text_file, ios::in);

    string line;
    while (getline(in_file, line)) {
        size_t delimeter_idx = line.find('|'); // Get The 1st Delimeter From Line
        if (code == line.substr(0, delimeter_idx)) {
            in_file.close(); // Avoid Non-Void Function Error
            return true;
        }
    }
```

```
    in_file.close();
    return false;
}

void STORE::add_data(string text_file, string code, string food, double f_price) {
    clear_screen();
    ofstream out_file(text_file, ios::app);

    display_data(text_file);
    cout << string(60, '-') << endl;
    cout << "Add Food Details (type 'Q' to quit)\n";
    cout << string(60, '-') << endl;

    int step = 1;
    while (true) {
        if (step == 1) {
            cout << "Enter Food Code: ";
            cin >> code;
            if (code == "q" || code == "Q") {
                break;

            } else if (check_code(text_file, code)) {

                cout << "----------Code Exist, Pls Add Another Code----------\n";
            } else {
                step++;
            }
        }
```

```
        if (step == 2) {
            cout << "Enter Food Description: ";
            cin.ignore();
            getline(cin, food);
            if (food == "q" || food == "Q") {
                cout << "----------------------Successfully Quit----------------------" << endl;
                break;
            } else {
                step++;
            }
        }

        if (step == 3) {
            cout << "Enter Food Price: ";
            while (!(cin >> f_price) || f_price < 0) {
                cout << "Please Enter a Valid Price: ";

                cin.clear(); // Clear The Error Input (Avoid Infinite Loop)
                cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            }

            out_file << code << "|" << food << "|" << f_price << endl;
            break;
        }
    }
    out_file.close();
}

void STORE::modify_data(string text_file, string code, string food, double f_price) {
```

```
clear_screen();

display_data(text_file);
cout << string(60, '-') << endl;
cout << "Modify Food Details (type 'Q' to quit)\n";
cout << string(60, '-') << endl;

int step = 1;
while (true) {
    if (step == 1) {
        cout << "Enter Food Code: ";
        cin >> code;
        if (code == "q" || code == "Q") {
            break;

        } else if (check_code(text_file, code)) {
            step++;

        } else {
            cout << "----------Code Does Not Exist, Cannot Be Modified----------";

            cin.clear(); // Clear The Error Input (Avoid Infinite Loop)
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            cin.get();
        }
    }

    if (step == 2) {
        char modify_opt;
```

```
            string new_code, new_food;
            double new_price;

            cout << "<C>ode <D>escription <P>rice     <Q>uit: ";
            cin >> modify_opt;
            cout << string(60, '-') << endl;

            ifstream in_file(text_file);
            ofstream temp_file("temp.txt"); // Create a Temporary File to Store Original
Data

            if (modify_opt == 'q' || modify_opt == 'Q') {
                break;

            } else if (modify_opt == 'c' || modify_opt == 'C') {
                cout << "Enter New Food Code: ";
                cin >> new_code;
                if (new_code == "q" || new_code == "Q") {
                    break;

                } else if (check_code(text_file, new_code)) {
                    cout << "----------New Code Same As Old Code, Please Reenter----
-----\n\n";

                } else {
                    string line;

                    while (getline(in_file, line)) {
                        size_t delimeter_idx1 = line.find('|');
```

```
                    size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1);
                    string current_code = line.substr(0, delimeter_idx1);
                        string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1);
                    string current_price = line.substr(delimeter_idx2 + 1);

                    if (code != current_code) {
                        temp_file << line << endl;
                    } else {
                        temp_file << new_code << "|" << current_food << "|" <<
current_price << endl;
                    }
                }
            }

        } else if (modify_opt == 'd' || modify_opt == 'D') {
            cout << "Enter New Food Description: ";
            cin >> new_food;
            if (new_food == "q" || new_food == "Q") {
                break;

            } else {
                string line;
                while (getline(in_file, line)) {
                    size_t delimeter_idx1 = line.find('|');
                    size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1);
                    string current_code = line.substr(0, delimeter_idx1);
                        string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1);
```

```
                    string current_price = line.substr(delimeter_idx2 + 1);

                    if (code != current_code) {
                        temp_file << line << endl;
                    } else {
                        temp_file << current_code << "|" << new_food << "|" <<
current_price << endl;
                    }
                }
            }

        } else if (modify_opt == 'p' || modify_opt == 'P') {
            cout << "Enter New Food Price: ";
            cin >> new_price;

            if (new_price < 0) {
                cout << "---------------Invalid Price, Please Reenter----------------";

                cin.ignore(numeric_limits<streamsize>::max(),  '\n');  //  Handle
Buffer
                cin.get(); // Pause
                break;

            } else if (new_price >= 0) {
                string line;
                while (getline(in_file, line)) {
                    size_t delimeter_idx1 = line.find('|');
                    size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1);
                    string current_code = line.substr(0, delimeter_idx1);
```

```
                        string current_food = line.substr(delimeter_idx1 + 1,
delimeter_idx2 - delimeter_idx1 - 1);
                        string current_price = line.substr(delimeter_idx2 + 1);

                        // If not modifying, write the original line to temp file
                        if (code != current_code) {
                            temp_file << line << endl;
                        } else {
                            temp_file << current_code << "|" << current_food << "|" <<
new_price << endl;
                        }
                    }

                } else {
                    cout << "---------------Invalid Price, Please Reenter----------------";

                    cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle
Buffer
                    cin.get(); // Pause
                    break;
                }
            }

            in_file.close();
            temp_file.close();

            remove(text_file.c_str());
            rename("temp.txt", text_file.c_str());
            break;
```

```
        }
    }
}

void STORE::delete_data(string text_file, string code, string food, double f_price) {
    clear_screen();

    display_data(text_file);
    cout << string(60, '-') << endl;
    cout << "Delete Food Details (type 'Q' to quit)\n";
    cout << string(60, '-') << endl;

    int step = 1;
    while (true) {
        if (step == 1) {
            cout << "Enter Food Code: ";
            cin >> code;
            if (code == "q" || code == "Q") {
                cout << "----------Successfully Quit----------" << endl;
                break;

            } else if (check_code(text_file, code)) {
                step++;

            } else {
                cout << "----------Code Does Not Exist, Cannot Be Deleted----------";

                cin.clear(); // Clear The Error Input (Avoid Infinite Loop)
                cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
```

```
            cin.get();
        }
    }


    if (step == 2) {
        ifstream in_file(text_file);
        ofstream temp_file("temp.txt"); // Create a Temporary File to Store Original
Data

        string line;
        while (getline(in_file, line)) {
            size_t delimeter_idx1 = line.find('|');
            string current_code = line.substr(0, delimeter_idx1);

            // Rewrite all lines except the one to be deleted/chosen
            if (code != current_code) {
                temp_file << line << endl;
            }
        }

        in_file.close();
        temp_file.close();

        remove(text_file.c_str());
        rename("temp.txt", text_file.c_str());

        break;
    }
```

```
    }
}

void STORE::calculate(string text_file, string code, int quantity, double f_price) {
    char proceed;
    double cal_price, total_price = 0;
    clear_screen();

    display_data(text_file);
    cout << string(60, '-') << endl;
    cout << "\t\t\t Choose Items" << endl;
    cout << string(60, '-') << endl;



    do {
        cout << "Enter Food Code: ";
        cin >> code;

        if (code == "q" || code == "Q") {
            cout << "----------------------Successfully Quit----------------------" << endl;
            return;

        } else if (check_code(text_file, code)) {
            cout << "Enter Quantity: ";
            cin >> quantity;
            if (quantity < 0) {
                cout << "--------------Invalid Quantity, Please Reenter---------------";

                cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
```

```
            cin.get();

        } else {
            ifstream in_file(text_file, ios::in);
            string line;
            while (getline(in_file, line)) {
                size_t delimeter_idx1 = line.find('|');
                size_t delimeter_idx2 = line.find('|', delimeter_idx1 + 1);
                string current_code = line.substr(0, delimeter_idx1);
                string current_food = line.substr(delimeter_idx1 + 1, delimeter_idx2
- delimeter_idx1 - 1);
                string current_price = line.substr(delimeter_idx2 + 1);

                if (code == current_code) {
                    f_price = stod(current_price); // Convert string to double
                    break;
                }
            }
            in_file.close();

            cal_price = f_price * quantity;
            total_price += cal_price;
            cout << "Total Price: RM" << cal_price << endl;

            cout << "Wish to proceed? <Y>es/<N>o: ";
            cin >> proceed;
            if (proceed == 'n' || proceed == 'N') {
                cout << string(60, '=') << endl;
                cout << "Overall Price: RM" << total_price << endl;
```

```
                cout << string(60, '=') << endl;
            }

            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            cin.get();
        }

    } else {
        cout << "---------Code Does Not Exist, Cannot Be Calculated----------";

        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
        cin.get();
    }
} while (proceed == 'y' || proceed == 'Y');
}

void STORE::track(string text_file, int distance, string hrs) {
    double f_price, operate_hrs;
    string code, food;
    char select;
    int quantity;

    while (true) {
        clear_screen();

        display_data(text_file);
        cout << string(60, '-') << endl;
        cout << "Walking Distance: " << distance << "km" << endl;
        cout << "Operating Hours: " << hrs << endl;
```

```
        cout << string(60, '-') << endl;
        cout << "<A>dd <M>odify <D>elete <C>alculate      <Q>uit" << endl;
        cout << string(60, '-') << endl;

        cout << "Enter Option: ";
        cin >> select;
        if (select == 'q' || select == 'Q') {
            cout << "----------------------Successfully Quit----------------------" << endl;
            break;
        } else if (select == 'a' || select == 'A') {
            add_data(text_file, code, food, f_price);
        } else if (select == 'm' || select == 'M') {
            modify_data(text_file, code, food, f_price);
        } else if (select == 'd' || select == 'D') {
            delete_data(text_file, code, food, f_price);
        } else if (select == 'c' || select == 'C') {
            calculate(text_file, code, quantity, f_price);
        } else {
            cout << "----------------Invalid Option, Pls Try Again----------------" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            cin.get();
        }
    }
}

void STORE::restaurant() {
    while (true) {
        clear_screen();
```

```
cout << string(60, '-') << endl;
cout << "\t\t       Restaurant Details\n";
cout << string(60, '-') << endl;

ifstream in_file("restaurant_data.txt", ios::in);
string line;
int i=1;
char ch;

while (getline(in_file, line)) {
    cout << "<" << i << "> " << line << endl;
    i++;
}
cout << string(60, '-') << endl;
in_file.close();

cout << "Enter Number of Restaurant    <Q>uit: ";
cin >> ch;
if (ch>='1' && ch<='8') {
    switch (ch) {
        case '1':
            track("ah_ma.txt", 5, "06:30 AM - 09:00 PM");
            break;
        case '2':
            track("desa_ctk.txt", 2, "24 Hours (Whole Day)");
            break;
        case '3':
            track("fun_fun.txt", 3, "08:00 AM - 04:00 PM");
```

```
                break;
            case '4':
                track("happy_thai.txt", 5, "11:00 AM - 03:00 PM / 05:00 PM - 10:00
PM");
                break;
            case '5':
                track("mrs_yau.txt", 7, "11:15 AM - 09:30 PM");
                break;
            case '6':
                track("mye_mye.txt", 2, "07:00 AM - 04:00 PM");
                break;
            case '7':
                track("taiwan_dami.txt", 3, "11:30 AM - 09:30 PM");
                break;
            case '8':
                track("wei_duo.txt", 9, "10:00 AM - 09:30 PM");
                break;
            default:
                cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                break;
        }
    } else if (ch == 'q' || ch == 'Q') {
        cout << "----------------------Successfully Quit ----------------------" << endl;
        break;
    } else {
        cout << "--------------Invalid Input! Please Try Again!--------------" << endl;
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
        cin.get(); // Pause
```

```
        }
    }
}

void STORE::cafe() {
    while (true) {
        clear_screen();

        cout << string(60, '-') << endl;
        cout << "\t\t\t Cafe Details\n";
        cout << string(60, '-') << endl;

        ifstream in_file("cafe_data.txt", ios::in);
        string line;
        int i=1;
        char ch;

        while (getline(in_file, line)) {
            cout << "<" << i << "> " << line << endl;
            i++;
        }
        cout << string(60, '-') << endl;
        in_file.close();

        cout << "Enter Number of Cafe     <Q>uit: ";
        cin >> ch;
        if (ch>='1' && ch<='7') {
            switch (ch) {
                case '1':
```

```
                track("bingxue.txt", 1, "11:00 AM - 11:00 PM");
                break;
            case '2':
                track("blackboard.txt", 1, "12:00PM - 09:00 PM");
                break;
            case '3':
                track("mixue.txt", 1, "11:00 AM - 10:00 PM");
                break;
            case '4':
                        track("secret_penang.txt", 3, "09:00 AM - 08:30
PM");
                break;
            case '5':
                track("snowy.txt", 2, "12:00PM - 09:00 PM");
                break;
            case '6':
                track("zaba_long.txt", 4, "10:00 AM - 10:00 PM");
                break;
            case '7':
                track("zus_coffee.txt", 3, "07:00 AM - 10:40 PM");
                break;
            default:
                cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                break;
        }
    } else if (ch == 'q' || ch == 'Q') {
        cout << "----------------------Successfully Quit ----------------------" << endl;
        break;
```

```
        } else {
            cout << "--------------Invalid Input! Please Try Again!--------------" << endl;
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            cin.get(); // Pause
        }
    }
}

void STORE::fast_food() {
    while (true) {
        clear_screen();

        cout << string(60, '-') << endl;
        cout << "\t\t      Fast Food Store Details\n";
        cout << string(60, '-') << endl;

        ifstream in_file("fast_food_data.txt", ios::in);
        string line;
        int i=1;
        char ch;

        while (getline(in_file, line)) {
            cout << "<" << i << "> " << line << endl;
            i++;
        }

        cout << string(60, '-') << endl;
        in_file.close();
```

```
        cout << "Enter Number of Fast Food Store     <Q>uit: ";
        cin >> ch;
        if (ch>='1' && ch<='4') {
            switch (ch) {
                case '1':
                    track("burger.txt", 2, "8.00AM - 11.00 PM");
                    break;
                case '2':
                    track("domino.txt", 8, "10.30AM - 11.00 PM");
                    break;
                case '3':
                    track("kfc.txt", 8, "08:00 AM - 11:00 PM");
                    break;
                case '4':
                    track("pizza.txt", 1, "10:00 AM - 11:00 PM");
                    break;
                default:
                    cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                    break;
            }

        } else if (ch == 'q' || ch == 'Q') {
            cout << "----------------------Successfully Quit ----------------------" << endl;
            break;

        } else {
            cout << "--------------Invalid Input! Please Try Again!--------------" << endl;
```

```
            cin.ignore(numeric_limits<streamsize>::max(), '\n');   // Handle Buffer
            cin.get(); // Pause
        }
    }
}
```

Store.h

```
#include <string>
using namespace std;

class STORE {
    public:
    // Core Functions
    void restaurant();
    void cafe();
    void fast_food();

    // Data Management Functions
    void display_data(string text_file);
    bool check_code(string text_file, string code);
    void add_data(string text_file, string code, string food, double f_price);
    void modify_data(string text_file, string code, string food, double f_price);
    void delete_data(string text_file, string code, string food, double f_price);
    void calculate(string text_file, string code, int quantity, double f_price);
    void track(string text_file, int distance, string hrs);

    // Clear All Data
    void clear_screen();
};
```

Review

```
/* File name: review.cpp */

#include "food_track.h"
#include "store.h"
#include "review.h"

#include <iostream>
#include <string>
#include <limits>
#include <fstream>
using namespace std;


void REVIEW::clear_screen() {
#ifdef _WIN32
    // Only for Window user
    system("cls");
#else
    // For Linux/Mac... user
    system("clear");
#endif
}

void REVIEW::user_info(string &name, string &city, string &state, string &date) {
    cout << string(60, '-') << endl;
    cout << string(20, ' ') << "Enter Personal Info" << endl;
    cout << string(60, '-') << endl;
```

```
    // Flush buffer
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    cout << "Name (Joey Chong): ";
    // To read a line of text
    getline(cin, name);
    cout << "City (Bandar Sungai Long): ";
    getline(cin, city);
    cout << "State (Selangor): ";
    getline(cin, state);
    cout << "Date (01/01/2025): ";
    getline(cin, date);
}

void REVIEW::user_review(string name, string city, string state, string date, string &rate, string &main_reason, string &comments) {
    ofstream out_file("temp_review.txt", ios::trunc);

    cout << string(60, '-') << endl;
    cout << string(12, ' ') << "Please rate your overall experience" << endl;
    cout << string(60, '-') << endl;

    cout << "Rating (*****): ";
    getline(cin, rate);
    cout << "Main Reason (Exceptional Service): ";
    getline(cin, main_reason);
    cout << "Comments/Feedbacks: ";
    getline(cin, comments);
```

```
    // Save it into temparary file for preview purpose
    out_file << name << ", from " << city << ", " << state
    << endl << rate << " for " << main_reason << endl << comments;

    out_file.close();
}

void REVIEW::display_review(string text_file) {
    ifstream in_file(text_file);

    cout << string(60, '-') << endl;
    cout << string(20, ' ') << "Feedbacks from Users" << endl;
    cout << string(60, '-') << endl;

    string line;
    while (getline(in_file, line)) {
        cout << line << endl;
    }
    in_file.close();
}

void REVIEW::temp_review() {
    ifstream in_file("temp_review.txt");

    cout << string(60, '-') << endl;
    cout << string(27, ' ') << "Preview" << endl;
    cout << string(60, '-') << endl;

    string line;
```

```
    while (getline(in_file, line)) {
        cout << line << endl;
    }
    in_file.close();
}

void REVIEW::modify_review(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments) {
    clear_screen();

    temp_review();

    user_info(name, city, state, date);
    user_review(name, city, state, date, rate, main_reason, comments);
    confirmation(text_file, name, city, state, date, rate, main_reason, comments);
}

void REVIEW::delete_review(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments) {
    // Overwrite Contents
    ofstream out_file("temp_review.txt", ios::trunc);
    out_file.close();
    cout << "----------------------Review Deleted----------------------" << endl;
    cin.ignore();
    cin.get();
}

void REVIEW::confirmation(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments) {
```

```
    // Append temp_review to review.txt
    ifstream temp_file("temp_review.txt");
    ofstream out_file(text_file, ios::app);

    string line;
    bool found=true;
    while (getline(temp_file, line)) {
        // Save The Data In Next Line
        if (found) {
            out_file << endl;
            found = false;
        }
        out_file << line << endl;
    }
    temp_file.close();
    out_file.close();

    clear_screen();
    // Now Show All Confirmed Reviews
    display_review(text_file);
    cout << "------------------Review Confirmed and Saved------------------" << endl;
    cin.ignore();
    cin.get();
}

void REVIEW::give_comment(string text_file) {
    char select;
    string name, city, state, date, rate, main_reason, comments;
```

```
    user_info(name, city, state, date);
    user_review(name, city, state, date, rate, main_reason, comments);

    while (true) {
        clear_screen();

        temp_review();
        cout << string(60, '-') << endl;
        cout << "<C>onfirm <M>odify <D>elete      <Q>uit" << endl;
        cout << string(60, '-') << endl;

        cout << "Enter Option: ";
        cin >> select;
        if (select == 'q' || select == 'Q') {
            cout << "----------------------Successfully Quit----------------------" << endl;
            break;

        } else if (select == 'c' || select == 'C') {
            confirmation(text_file, name, city, state, date, rate, main_reason, comments);
            break;

        } else if (select == 'm' || select == 'M') {
                modify_review(text_file, name, city, state, date, rate, main_reason,
comments);
            break;

        } else if (select == 'd' || select == 'D') {
                delete_review(text_file, name, city, state, date, rate, main_reason,
comments);
```

```
            break;

        } else {
            cout << "----------------Invalid Option, Pls Try Again---------------" << endl;

            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle buffer
            cin.get();
        }
    }
}

void REVIEW::track(string text_file) {
    char select;
    string name, city, state, date, rate, main_reason, comments;

    while (true) {
        clear_screen();

        cout << string(60, '-') << endl;
        cout << "\t\t   Select Option" << endl;
        cout << string(60, '-') << endl;
        cout << "<1> " << "Give Comment" << endl;
        cout << "<2> " << "View Review" << endl;
        cout << "<Q> " << "Quit" << endl;
        cout << string(60, '-') << endl;

        cout << "Enter Option: ";
        cin >> select;
        if (select == '1') {
```

```
            clear_screen();
            give_comment(text_file);
            break;

        } else if (select == '2') {
            clear_screen();
            display_review(text_file);
            cin.ignore();
            cin.get();

        } else if (select == 'q' || select == 'Q') {
            cout << "----------------------Successfully Quit----------------------" << endl;
            break;

        } else {
            cout << "----------------Invalid Option, Pls Try Again----------------" << endl;

            cin.clear(); // Clear The Error Input (Avoid Infinite Loop)
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle buffer
            cin.get();
        }
    }
}

void REVIEW::restaurant_r() {
    while (true) {
        clear_screen();

        cout << string(60, '-') << endl;
```

```
cout << "\t\tRestaurant Details\n";
cout << string(60, '-') << endl;

ifstream in_file("restaurant_data.txt", ios::in);
string line;
int i=1;
char ch;

while (getline(in_file, line)) {
    cout << "<" << i << "> " << line << endl;
    i++;
}
cout << string(60, '-') << endl;
in_file.close();

cout << "Enter Number of Restaurant    <Q>uit: ";
cin >> ch;
if (ch>='1' && ch<='8') {
    switch (ch) {
        case '1':
            track("ah_ma_review.txt");
            break;
        case '2':
            track("desa_review.txt");
            break;
        case '3':
            track("fun_review.txt");
            break;
        case '4':
```

```
                track("happy_thai_review.txt");
                break;
            case '5':
                track("mrs_yau_review.txt");
                break;
            case '6':
                track("mye_review.txt");
                break;
            case '7':
                track("taiwan_review.txt");
                break;
            case '8':
                track("wei_duo_review.txt");
                break;
            default:
                cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                break;
        }

    } else if (ch == 'q' || ch == 'Q') {
        cout << "----------------------Successfully Quit----------------------" << endl;
        break;

    } else {
        cout << "--------------Invalid Input! Please Try Again!--------------" << endl;

        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
        cin.get(); // Pause
```

```
        }
    }
}

void REVIEW::cafe_r() {
    while (true) {
        clear_screen();

        cout << string(60, '-') << endl;
        cout << "\t\t   Cafe Details\n";
        cout << string(60, '-') << endl;

        ifstream in_file("cafe_data.txt", ios::in);
        string line;
        int i=1;
        char ch;

        while (getline(in_file, line)) {
            cout << "<" << i << "> " << line << endl;
            i++;
        }
        cout << string(60, '-') << endl;
        in_file.close();

        cout << "Enter Number of Cafe     <Q>uit: ";
        cin >> ch;
        if (ch>='1' && ch<='7') {
            switch (ch) {
                case '1':
```

```
                track("bingxue_review.txt");
                break;
            case '2':
                track("blackboard_review.txt");
                break;
            case '3':
                track("mixue_review.txt");
                break;
            case '4':
                track("secret_review.txt");
                break;
            case '5':
                track("snowy_review.txt");
                break;
            case '6':
                track("zaba_review.txt");
                break;
            case '7':
                track("zus_review.txt");
                break;
            default:
                cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                break;
        }

    } else if (ch == 'q' || ch == 'Q') {
        cout << "----------------------Successfully Quit ----------------------" << endl;
        break;
```

```
        } else {
            cout << "--------------Invalid Input! Please Try Again!--------------" << endl;

            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            cin.get(); // Pause
        }
    }
}

void REVIEW::fast_food_r() {
    while (true) {
        clear_screen();

        cout << string(60, '-') << endl;
        cout << "\t\t        Fast Food Store Details\n";
        cout << string(60, '-') << endl;

        ifstream in_file("fast_food_data.txt", ios::in);
        string line;
        int i=1;
        char ch;

        while (getline(in_file, line)) {
            cout << "<" << i << "> " << line << endl;
            i++;
        }
        cout << string(60, '-') << endl;
        in_file.close();
```

```
        cout << "Enter Number of Fast Food Store     <Q>uit: ";
        cin >> ch;
        if (ch>='1' && ch<='4') {
            switch (ch) {
                case '1':
                    track("burger_review.txt");
                    break;
                case '2':
                    track("domino_review.txt");
                    break;
                case '3':
                    track("kfc_review.txt");
                    break;
                case '4':
                    track("pizza_review.txt");
                    break;
                default:
                    cout << "--------------Invalid Input! Please Try Again!--------------"
<< endl;
                    break;
            }

        } else if (ch == 'q' || ch == 'Q') {
            cout << "----------------------Successfully Quit----------------------" << endl;
            break;

        } else {
            cout << "--------------Invalid Input! Please Try Again!--------------" << endl;
```

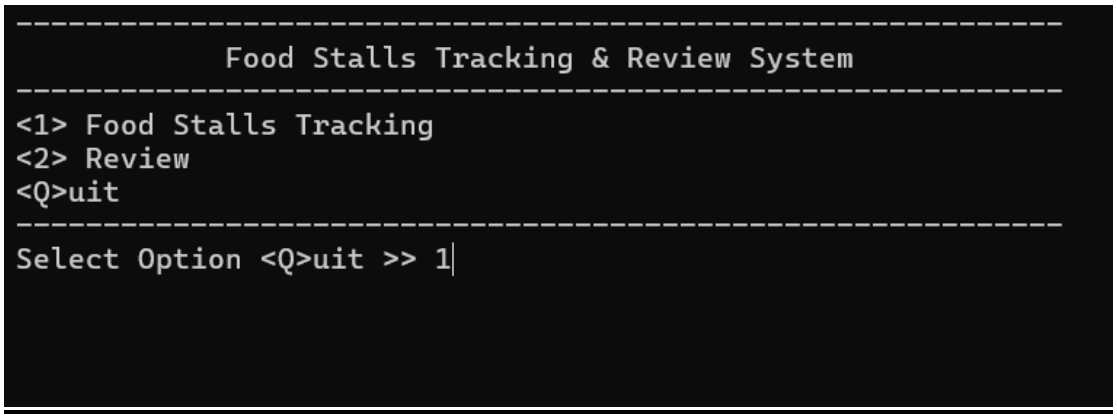
```
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Handle Buffer
            cin.get(); // Pause
        }
    }
}
```

Review.h

```
#include <string>
using namespace std;

class REVIEW {
    public:
    // Core Functions
    void restaurant_r();
    void cafe_r();
    void fast_food_r();

    // Data Management Functions
    void user_info(string &name, string &city, string &state, string &date);
    void user_review(string name, string city, string state, string date, string &rate, string &main_reason, string &comments);
    void display_review(string text_file);
    void temp_review();
    void modify_review(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments);
    void delete_review(string text_file, string name, string city, string state, string date, string rate, string main_reason, string comments);
    void confirmation(string text_file, string name, string city, string state, string date,
```

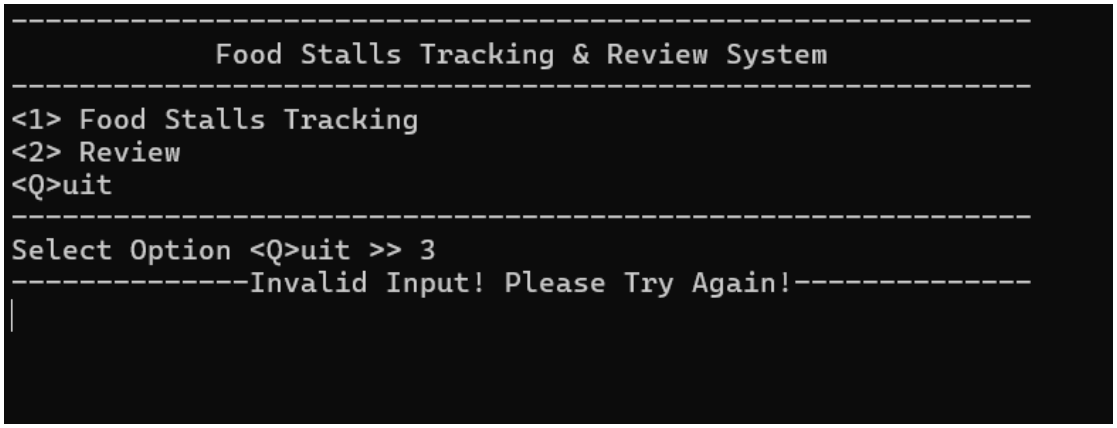
```
string rate, string main_reason, string comments);
    void give_comment(string text_file);
    void track(string text_file);

    // Clear All Data
    void clear_screen();
}
```

3.0 Sample output

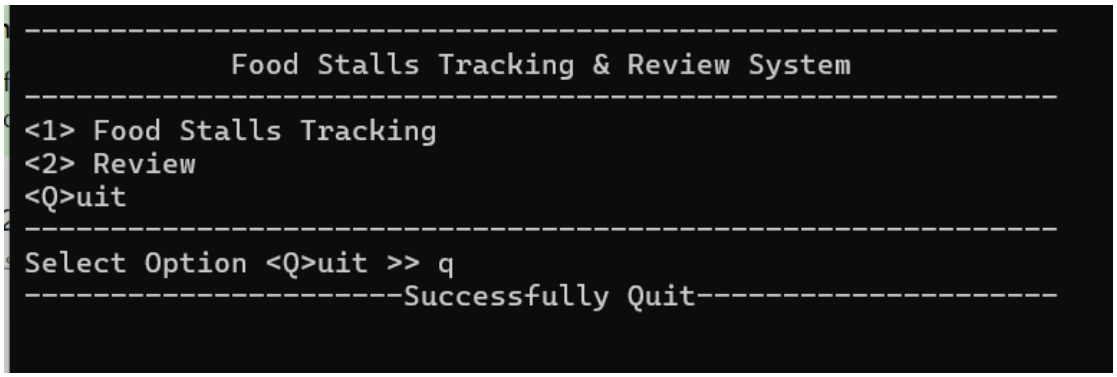


This is the main menu for user to choose from 1(Food stalls tracking) and 2 (Review).

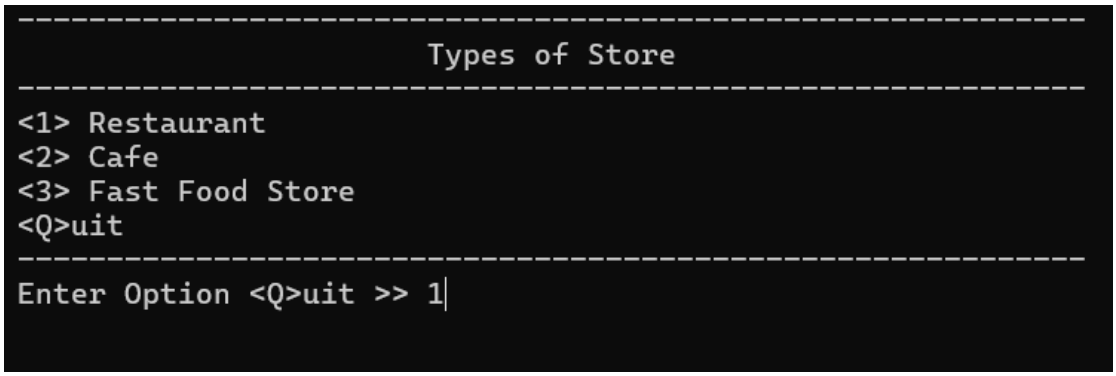
Here go through food stalls tracking



This for invalid input in main menu.



Quit option.



Input 1 for restaurant

Here we choose first restaurant to go to tracking system.

Handle invalid input.

```
------------------------------------------------------------
Codes    Foods & Beverages                            Prices
------------------------------------------------------------
001      Kuching Laksa (S)                             13.00
002      Fish Ball Noodle (S)                          11.00
003      Shredded Chicken Hor Fun (S)                  13.00
004      Nasi Lemak                                    13.90
005      Lu Rou Rice                                   13.90
006      loklok                                        10.00
------------------------------------------------------------
                        Choose Items
------------------------------------------------------------
Enter Food Code: 001
Enter Quantity: 1
Total Price: RM13.00
Wish to proceed? <Y>es/<N>o: y

Enter Food Code: 002
Enter Quantity: 1
Total Price: RM11.00
Wish to proceed? <Y>es/<N>o: n
============================================================
Overall Price: RM24.00
============================================================
```

Then, user can choose what food he prefers, and the system will done the calculation and display the overall price.

```
------------------------------------------------------------
Codes    Foods & Beverages                            Prices
------------------------------------------------------------
001      Kuching Laksa (S)                             13.00
002      Fish Ball Noodle (S)                          11.00
003      Shredded Chicken Hor Fun (S)                  13.00
004      Nasi Lemak                                    13.90
005      Lu Rou Rice                                   13.90
006      Lok Lok                                        8.00
------------------------------------------------------------
Walking Distance: 5km
Operating Hours: 06:30 AM - 09:00 PM
------------------------------------------------------------
<A>dd <M>odify <D>elete <C>alculate    <Q>uit
------------------------------------------------------------
Enter Option:
```

Staff can add, modify, delete or proceed to calculation

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
------------------------------------------------------------
Add Food Details (type 'Q' to quit)
------------------------------------------------------------
Enter Food Code: 006
Enter Food Description: Lok Lok
Enter Food Price: 8
```

Staff can add other food under add option

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
006      loklok                                          10.00
------------------------------------------------------------
Delete Food Details (type 'Q' to quit)
------------------------------------------------------------
Enter Food Code: 006
```

Staff can delete the food

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
------------------------------------------------------------
Walking Distance: 5km
Operating Hours: 06:30 AM - 09:00 PM
------------------------------------------------------------
<A>dd <M>odify <D>elete <C>alculate    <Q>uit
------------------------------------------------------------
Enter Option:
```

After deleted

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
007      loklok                                          10.00
------------------------------------------------------------
Delete Food Details (type 'Q' to quit)
------------------------------------------------------------
Enter Food Code: 006
----------Code Does Not Exist, Cannot Be Deleted------------
```

Validation for delete

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
006      loklok                                          10.00
------------------------------------------------------------
Modify Food Details (type 'Q' to quit)
------------------------------------------------------------
Enter Food Code: 006
<C>ode <D>escription <P>rice   <Q>uit: c
------------------------------------------------------------
Enter New Food Code: 007
```

Staff can modify food code

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
007      loklok                                          10.00
------------------------------------------------------------
Walking Distance: 5km
Operating Hours: 06:30 AM - 09:00 PM
------------------------------------------------------------
<A>dd <M>odify <D>elete <C>alculate    <Q>uit
------------------------------------------------------------
Enter Option:
```

After modify food code for lok lok

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
006      loklok                                          10.00
------------------------------------------------------------
Modify Food Details (type 'Q' to quit)
------------------------------------------------------------
Enter Food Code: 006
<C>ode <D>escription <P>rice   <Q>uit: p
------------------------------------------------------------
Enter New Food Price: 9
```

Staff can modify the food price

```
------------------------------------------------------------
Codes    Foods & Beverages                              Prices
------------------------------------------------------------
001      Kuching Laksa (S)                               13.00
002      Fish Ball Noodle (S)                            11.00
003      Shredded Chicken Hor Fun (S)                    13.00
004      Nasi Lemak                                      13.90
005      Lu Rou Rice                                     13.90
006      loklok                                           9.00
------------------------------------------------------------
Walking Distance: 5km
Operating Hours: 06:30 AM - 09:00 PM
------------------------------------------------------------
<A>dd <M>odify <D>elete <C>alculate    <Q>uit
------------------------------------------------------------
Enter Option:
```

After modify food price

```
------------------------------------------------------------
Codes    Foods & Beverages                            Prices
------------------------------------------------------------
001      Kuching Laksa (S)                             13.00
002      Fish Ball Noodle (S)                          11.00
003      Shredded Chicken Hor Fun (S)                  13.00
004      Nasi Lemak                                    13.90
005      Lu Rou Rice                                   13.90
006      loklok                                        10.00
------------------------------------------------------------
Modify Food Details (type 'Q' to quit)
------------------------------------------------------------
Enter Food Code: 006
<C>ode <D>escription <P>rice   <Q>uit: p
------------------------------------------------------------
Enter New Food Price: -1
---------------Invalid Price, Please Reenter----------------
```

Invalid price validation

```
------------------------------------------------------------
                  Select Option
------------------------------------------------------------
<1> Give Comment
<2> View Review
<Q> Quit
------------------------------------------------------------
Enter Option:
```

User can go in review system to choose either give comment or view review

```
------------------------------------------------------------
                  Select Option
------------------------------------------------------------
<1> Give Comment
<2> View Review
<Q> Quit
------------------------------------------------------------
Enter Option: 3
----------------Invalid Option, Pls Try Again---------------
```

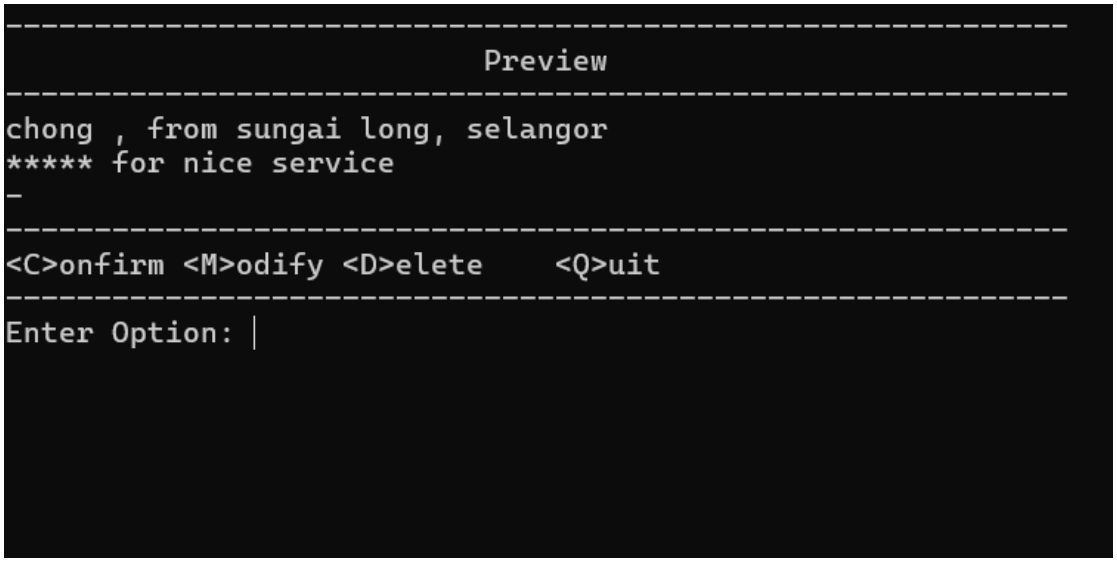
Handle invalid option

```
------------------------------------------------------------
                    Feedbacks from Users
------------------------------------------------------------
Carolyn Chong, from Bandar Sungai Long, Selangor
***** for Exceptional Service
Excellent Service

Lesley Lin, from Shah Alam, Selangor
*** for Poor Cleanliness
Bad Environment

Adam Lai, from Bandar Sungai Long, Selangor
**** for Nice Taste of Food
-
```

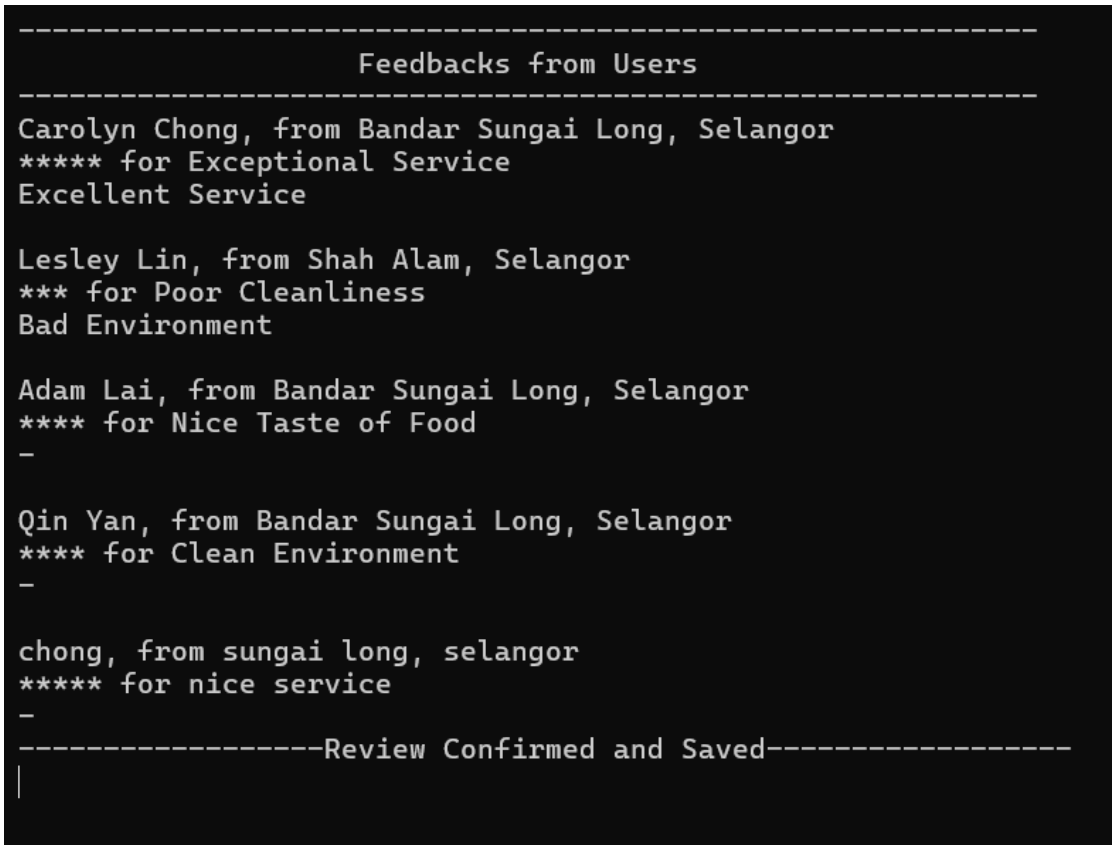
This is for view review option

```
------------------------------------------------------------
                    Enter Personal Info
------------------------------------------------------------
Name (Joey Chong): chong
City (Bandar Sungai Long): sungai long
State (Selangor): selangor
Date (01/01/2025): 01/02/2024
------------------------------------------------------------
            Please rate your overall experience
------------------------------------------------------------
Rating (*****): *****
Main Reason (Exceptional Service): nice service
Comments/Feedbacks: -
```

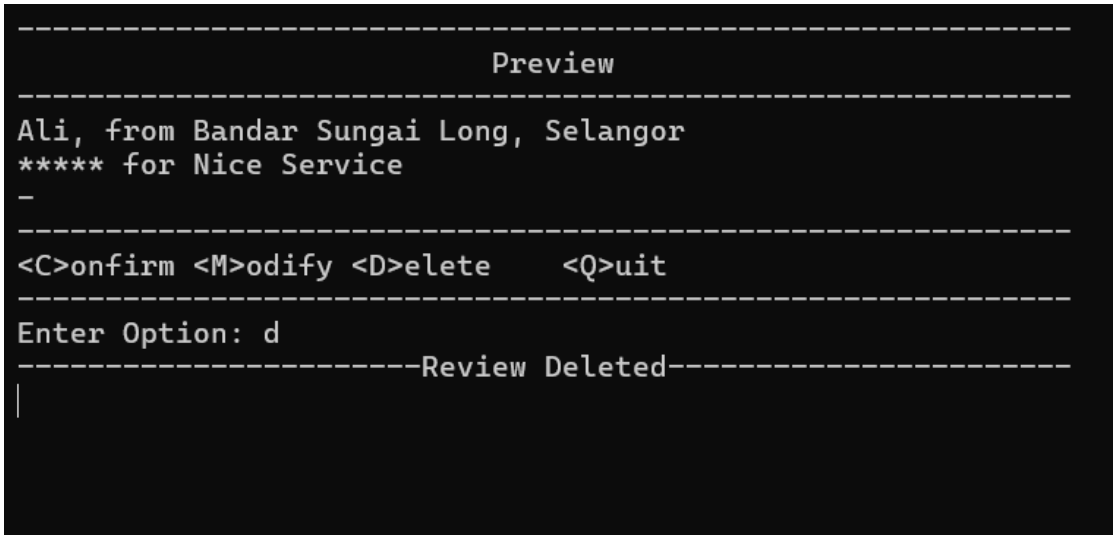
Under option 1, user can give their comments



Then, user can choose whether to confirm, modify, delete or quit.

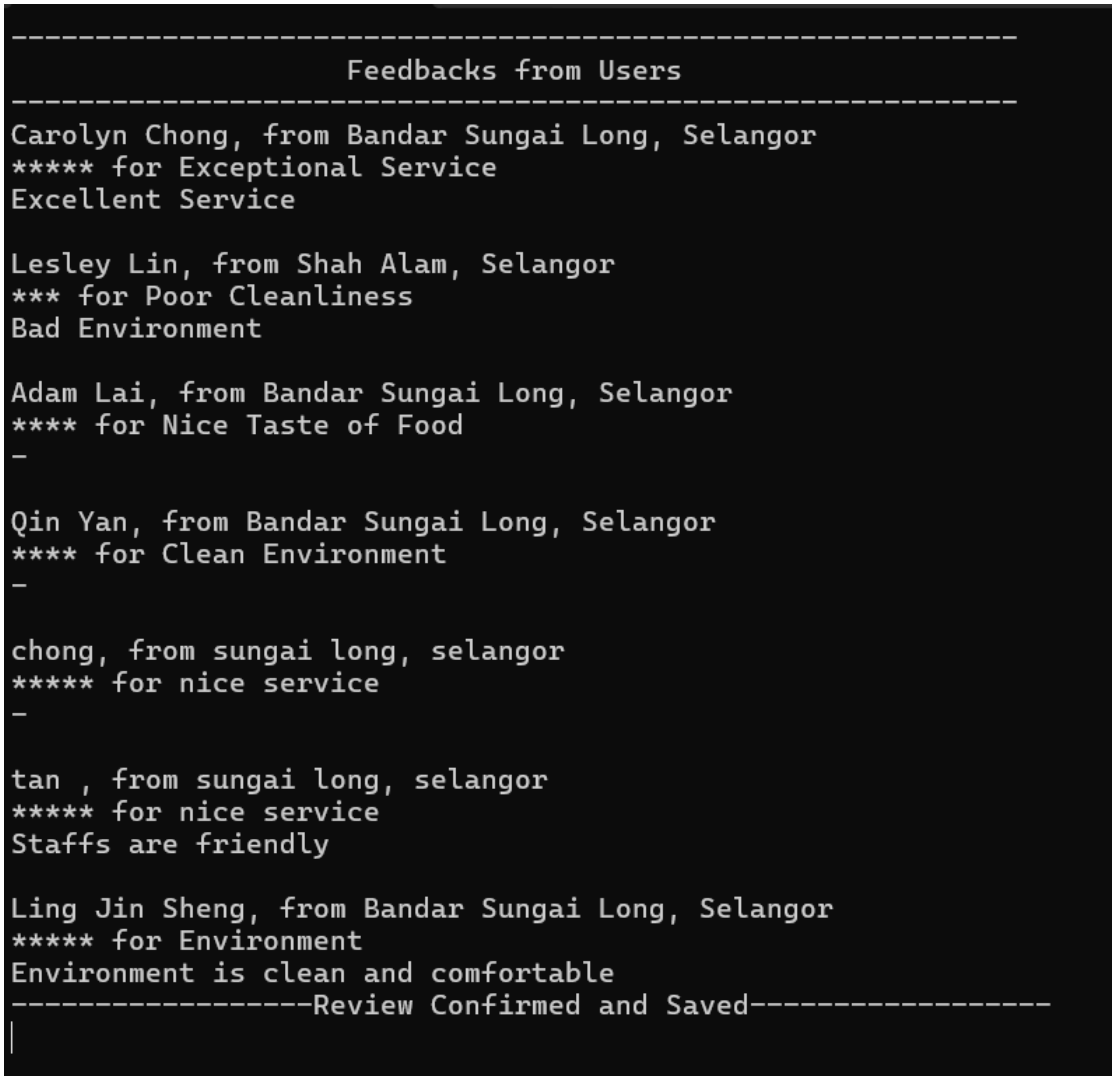


This is for confirm option



User can delete review

For modify option. User can choose to modify their rating and review if there is any error.



After modified, review is confirmed and saved in the text file.

4.0 Sample Input

Ah_ma.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

*** for Nice taste of foods

Services have to improve

Ah_ma_review.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

Lesley Lin, from Shah Alam, Selangor

*** for Poor Cleanliness

Bad Environment

Adam Lai, from Bandar Sungai Long, Selangor

**** for Nice Taste of Food

-

Qin Yan, from Bandar Sungai Long, Selangor

**** for Clean Environment

-

chong, from sungai long, selangor

***** for nice service

-

tan , from sungai long, selangor

***** for nice service

Staffs are friendly

Ling Jin Sheng, from Bandar Sungai Long, Selangor

***** for Environment

Environment is clean and comfortable

Bingxue.txt

001|Strawberry Sundae|6

002|Chocolate Sundae|6

003|Brown Sugar Bubble Tea|6.5

004|Honey Peach Milk Tea|8

005|Iced Passion Fruit|7

006|Lemon Black Tea|4.99

007|Grape Bucket|12

008|Mango Coconut Milk|9

009|Honey Peach Coconut Milk|9

010|Mango Milkshake|6

Bingxue_review.txt

Ling Qing Yan, from Bandar Sungai Long, Selangor

*** for Excellent Service

Staffs are friendly and well-behave

Ali, from Bandar Sungai Long, Selangor

***** for Nice Taste for Beverage

-

Blackboard.txt

001|Mongolian Chicken With Rice|14.9

002|Sweet & Sour Chicken With Rice|14.9

003|Mongolian Fish With Rice|14.9

004|Lemon Chicken With Rice|14.9

005|Chicken Maryland|25.9

006|Fried Chicken Chop With Carbonara Spaghetti|23.9

007|Fish & Chips|21.9

008|Grilled Chicken Chop With Baked Cheese|26.9

009|Kiwi Fruit Tea|7.9

010|Oolong Green Tea|5.9

Blackboard_review.txt

Adam, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

Burger.txt

001|Fish N Crisp|8.5

002|Chick N Crisp|7.3

003|Long Chicken|10.2

004|Tendercrisp|12.95

005|Tendergrill|12.2

006|Cheeseburger|8.5

007|Double cheeseburger|12.6

008|Whopper Jr|8.9

009|Whopper|14.9

010|Single BBQ Beafacon|11.3

burger_review.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

cafe_data.txt

BingXue

BlackBoard

MiXue

Secret Penang

Snowy Ice

Zaba Long

Zus Coffee

desa_ctk.txt

001|Nasi Goreng|5

002|Maggi Goreng|5

003|Roti Canai|1.5

004|Kuew Teow Goreng|5

005|Nasi Ayam, Sayur|8.5

006|French Toast|3.5

007|Milo Ice|3

008|Teh O Limau Ice|2.3

009|Nescafe O|2.5

010|Cham|2.5

desa_review.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

domino.txt

001|Chicken Pepperoni Pizza|35.9

002|Aloha Chicken Pizza (Regular)|35.9

003|Chocolate Lava Cake|12.9

004|Honey Garlic Roasted Chicken (6/pcs)|20.9

005|Bottle Pepsi Zero Sugar (1.5L)|7.9

006|Bottle 7 Up (1.5L)|7.9

domino_review.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

fast_food_data.txt

Burger King

Domino

KFC

Pizza Hut

fun_fun.txt

001|Traditional Pork Noodle (Soup/Dry)|10.5

002|Bitter Gourd Pork Noodle (Soup/Dry)|12

003|Herb Pork Noodle (Soup/Dry)|12

004|Seaweed Pork Noodle (Soup/Dry)|12

005|Egg Pork Noodle (Soup/Dry)|12

006|All-in-one Pork Noodle (Soup/Dry)|16.5

007|XO Pork Noodle|14.5

008|Fish Head Noodle|14.5

009|Spicy Pork Noodle (Soup/Dry)|12

010|Mushroom Pork Noodle (Soup/Dry)|12

fun_review.txt

Ali, from Bandar Sungai Long, Selangor

*** for Price

Price is expensive

happy_thai.txt

001|Belacan Fried Rice|19.9

002|Tomyam Fried Rice|15.9

003|Thai Style Fried Rice|13.9

004|Green Curry Fried Rice|15.9

005|Kailan Crispy Pork Rice|15.9

006|Pad Thai|15.9

007|Fried Mama Mee|15.9

008|Stir Fried Noodles|14.9

009|Kangkung Belacan|15.9

010|Bittergourd Egg|17.9

happy_thai_review.txt

Ashley Lee, from Bandar Sungai Long, Selangor

***** for Exceptional Services

Nice and friendly services from staffs

kfc.txt

001|3pc Portuguese Egg Tart|5.5

002|Snaker Box|16.49

003|Snack Plate (NO DRINK)|18.99

004|1pc Rice Combo|15.99

005|Spicy Mala Burger Combo|14.88

006|Zinger Classic Combo|19.99

007|Colonel Classic Combo|15.49

008|6pc Nuggets Combo|15.99

009|Loaded Cheezy Fries|8.99

010|Cheezy Wedges (L)|8.99

kfc_review.txt

Li Wei, from Bandar Sungai Long, Selangor

*** for Friendly price

Services are too slow, and the table is dirty

mixue.txt

001|Ice Cream|2

002|Fresh Lemonade|3

003|Super Boba Sundae|5

004|O-CoCo Milk Tea|6

005|Ice Cream Toffee Hazelnut Latte|5.5

006|Creamy Mango Boba|7

007|Passion Fruit Bubble Tea|6

008|Peach Mi-Shake|5

009|Kiwi Jasmine Tea|5

010|Classical Milk Tea|5

mixue_review.txt

Ashley Chin, from Bandar Sungai Long, Selangor

**** for Exceptional Service

Overall, the environemnt is clean and comfortable,

price is friendly for students.

However, payment method need to improve.

mrs_yau.txt

001|Beef Noodles Soup|10.5

002|Baked Chicken Cheese Rice|22.9

003|Pork Soup With Rice|10.5

004|Taiwan Sausage (3/pcs)|10.9

005|Crispy Fry Dumpling|15.9

006|Iced Latte |10.9

007|Iced Lime Tea|5.9

008|Iced Blueberry Yogurt|7.5

mrs_yau_review.txt

John, from Bandar Sungai Long, Selangor

*** for Exceptional Service

Environment need to improve

mye_mye.txt

001|White Coffee|3

002|Oat Cereal|2.8

003|Kopi O|2.6

004|Teh|2.6

005|Teh O|2.6

006|Teh C|2.8

007|Cham|3

008|Horlicks|3.2

009|Nescafe|3.5

010|Kopi|2.8

mye_review.txt

John, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

pizza.txt

001|Ramandan special-2 regular pizzas|19.8

002|Ramadan Deal 4 Pax Combo|39.6

003|Mybox Pizza duo|30

004|Hut's meal pasta 2(2Pax)|31.9

005|Hut's meal 2(2 pax)|31.9

006|Double Box large (5-6Pax)|58.9

007|Aloha chicken with thousand island (L)|35.9

008|Island Tuna with thousand island (L)|35.9

009|Beef pepperoni with thousand island (L)|35.9

010|BBQ chicken with thousand island (L)|35.9

pizza_review.txt

Zhang Jie, from Bandar Sungai Long, Selangor

**** for Comfortable Environment

Foods and beverages are quite expensive

restaurant_data.txt

Ah Ma Sarawak Kampua Mee

Desa CTK

Fun Fun Kitchen

Happy Thai Kitchen

Mrs Yau Hong Kong Style

Mye Mye

Taiwan Dami

Wei Duo Mei Food

secret_penang.txt

001|Penang Famous Fried Kuey Teow|11

002|Penang White Curry Mee|11

003|Penang Hokkien Prawn Mee|11

004|Penang Loh Mee|10

005|Penang Asam Laksa|10

006|Penang Kuey Teow Soup|9.5

007|Penang Loh Bak|18

008|Fish Ball Soup|8

009|Fried Fish Cake|10

010|Fried Shrimp Cake|8

secret_revieew.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

** for Poor Service

Bad Attitutude and behaviour

snowy.txt

001|Bingsu(3 topping)|12.9

002|Bingsu(5 topping)|14.9

003|Snow Fungus Longan|4.9

004|Longan Jelly|4.9

005|Jasmine Tea|3

006|Arno White Taro Ball|10.9

007|Explosion Glutinous Rice Cake (Yogurt)|9.9

008|Glutinous Rice Cake (Dipping Black Sugar)|9.9

009|Explosion Glutinous Rice Cake (Brown sugar)|9.9

010|Grass Jelly + Taro Balls|9.9

snowy_review.txt

Lee Guo Jian, from Bandar Sungai Long, Selangor

**** for Exceptional Service

Staffs are friendly and kind

Taiwan_dami.txt

001|Braised Beef Soup Noodle|20.9

002|Pork Dumpling Soup Noodle|11.9

003|Taiwanese Sun Bei Chicken Rice|26.9

004|Ma Po To Fu Rice|20.9

005|Sweet & Sour Chicken Bento|17.9

006|SAN PIN Bento|19.9

007|Fried Chicken Chop with Black Pepper Sauce Bento|17.9

008|Braised Pork "PAI KU" Rice|20.9

009|Jasmine Green Tea|3

010|Lemon AiYu Jelly|1.9

Taiwan_review.txt

Chong Xin, from Bandar Sungai Long, Selangor

*** for Nice Service

Still can improve the cleanliness of the environment

Temp_review.txt

Wei_duo.txt

001|Hang Zhou Xiao Lang Bao (6/pcs)|7

002|Steamed Dumplings (8/pcs)|8

003|Tasty Wonton (8/pcs)|6

004|Hot and Sour Rice Noodle|8

005|Fried Rice with Egg|8

006|Fried Noodle with Egg|10

007|Pork Bone Soup Noodle|12

008|Griddle-Cooked Fat Intestines|26.8

Wei_duo_review.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

** for Poor services and environment

Staffs are disrespect to customers, and the environment is not dirty

Zaba_long.txt

001|Kerabu Rice|9.9

002|Japanese Curry Chicken Rice|18.5

003|Japan Chicken Rice|13.9

004|Potato Luncheon Meat And Scrambled Egg Rice Bowl|12.5

005|Double Mushroom And Scrambled Egg Rice Bowl|13.9

006|Curry Chicken Nasi Lemak|17.9

007|Crunchy Chicken Chop Noodles|15.5

008|Fried Chicken (2 Pcs)|15.5

009|French Fries|11.5

010|Nestum+Kopi+Milo|6.6

Zaba_review.txt

Ali, from Bandar Sungai Long, Selangor

***** for Exceptional Service

Excellent Service

Zus_coffee.txt

001|White Peach Oolong Lemonade 1L|24.9

002|Matcha latte 1L|33.9

003|Spanish latte|11.2

004|Ceo latte|9.9

005|Zus signature Curry Puff|3.9

006|Chicken slice & Cheese Bagel|10.9

007|Chocolate roll|8.5

008|Pearl white sugar|6.9

009|Tom yum chicken puff|3.9

010|Big brekkie croissant|10.9

Zus_review.txt

Carolyn Chong, from Bandar Sungai Long, Selangor

*** for Nice taste of foods

Services have to improve

5.0 Contribution of contribution

Name	Contribution
Tee Le Xuan	Review system code
Nicoleete Tu Tze Ying	Flowchart
Evelyn Chin Shien Lin	Tracking system code
Hooi Guan Weng	Structure chart and report