

**School of Computer Sciences**

**Universiti Sains Malaysia**

**CPT111 – Principles of Programming**

**Academic Session 2020/2021, Semester 1**

**Assignment 2**

**Date: 17 JANUARY 2021**

|  |  |
| --- | --- |
| **Name** | **Muhammad Armand Bin Muhammad Fazli** |
| **Matric No.** | **153857** |
| **Group** | **B** |

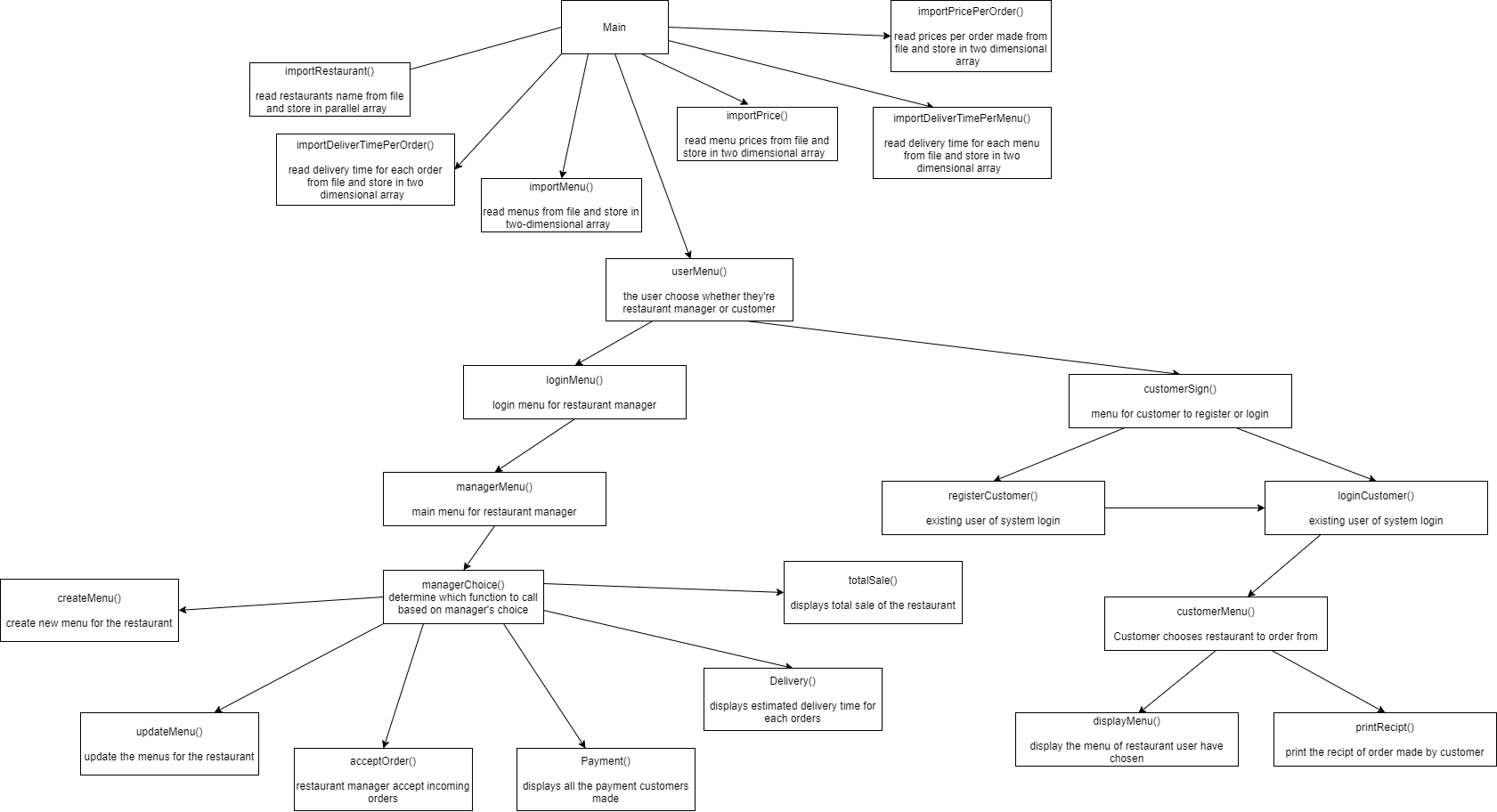
**INTRODUCTION**

This program is developed to allow customers to order food from restaurants to be delivered straight to their door, and allows restaurant managers to manage their restaurant as in , creating a new menu, updating menu, accepting orders, and more to maintain their business.It is a simple Food Ordering System made for convenience for both the customers and restaurant managers to use. A lot of two-dimensional array and parallel array are used to process information imported from multiple files . This is so that the program have continuity so that the program doesn’t lose data on every execution.

**PROBLEM ANALYSIS**

With COVID-19 making social distancing becoming the norm and eating in a crowded restaurant no longer being possible , it is a huge problem for people’s who want to enjoy delicious food from restaurants without having to go out and be in the risk of getting COVID-19. It is also a big problem for restaurants to maintain their business in this tough times we live in. Therefore, a Food Ordering System is needed to be developed to solve this problem where users can order food to be delivered straight to their door. The system also need to be full of features for the Restaurant Managers to use such as , creating a new menu ,updating menu price, calculate total sales, and more.

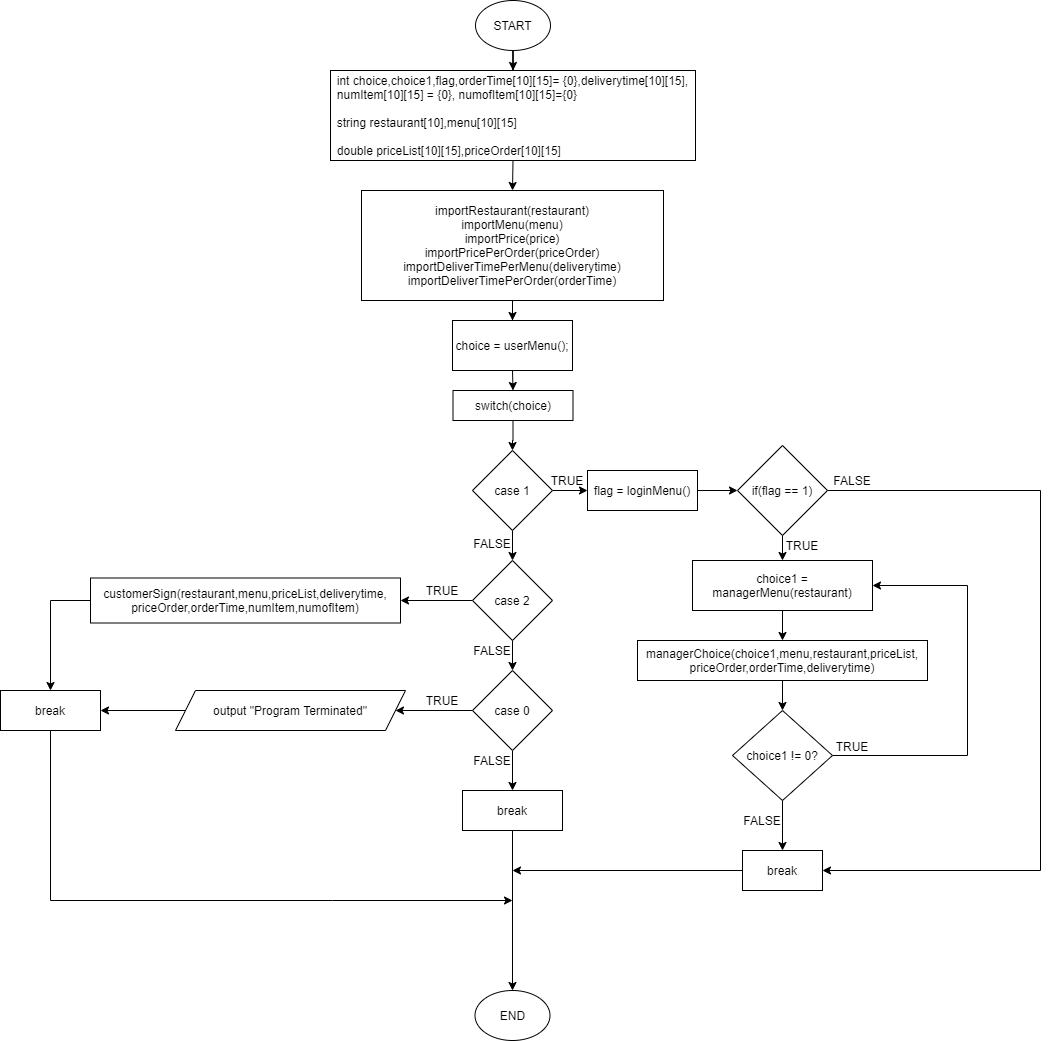
**STRUCTURE CHART**



( For sharper image, refer to Structure.png included in the folder )

**FLOWCHART**

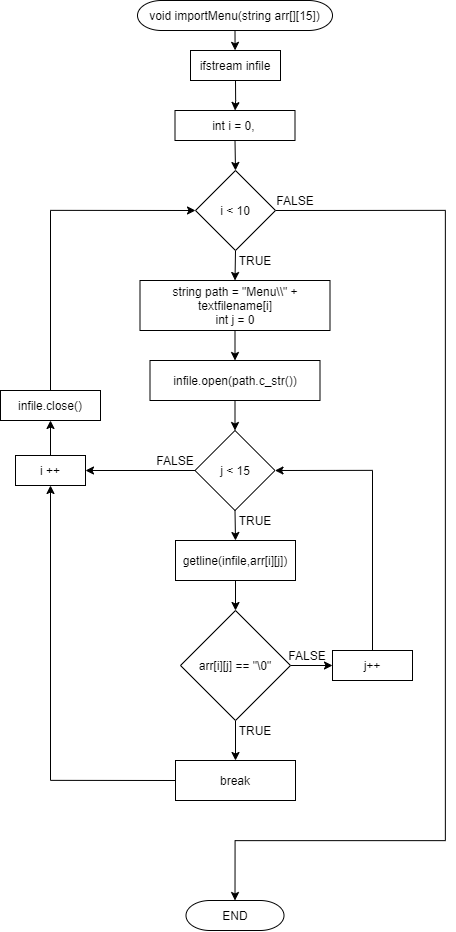
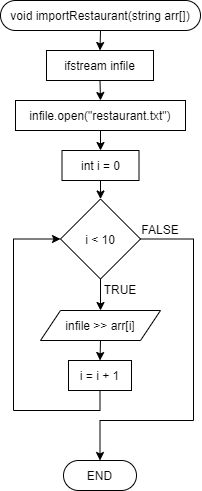
Main Flow Chart



( For clearer image, refer to main.png in the Flowchart folder )

**Importing Datas Into Arrays Flowchart**

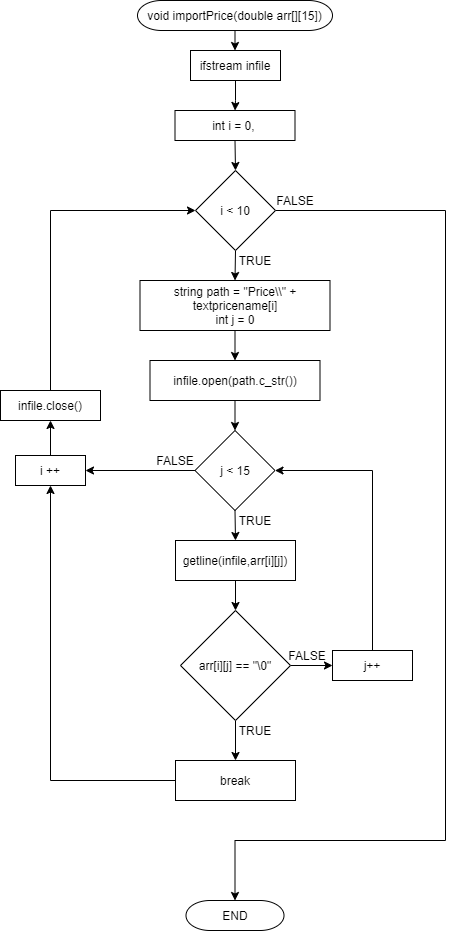
Import Restaurant Import Menus



Import Restaurant is for importing the restaurant names into a parallel array meanwhile, Import Menu is for importing the menus of each restaurant into a two-dimensional array.

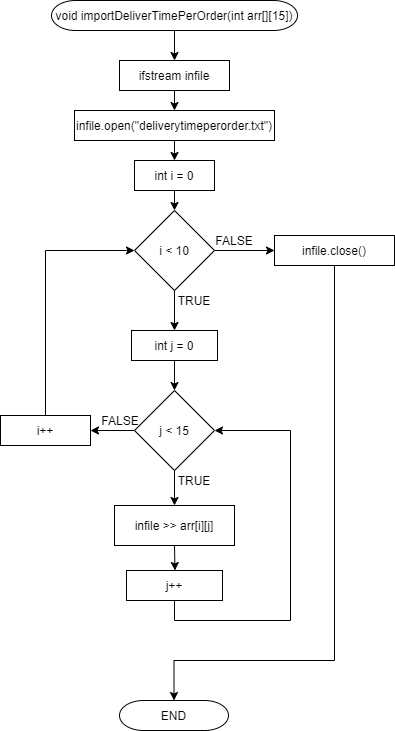
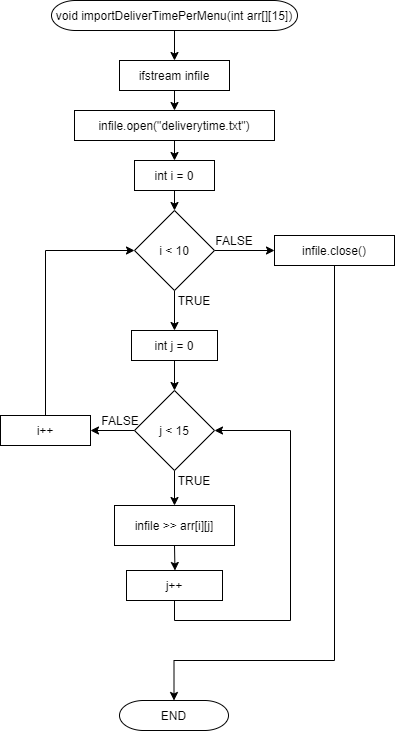
The names of the restaurant is imported from restaurant.txt meanwhile the menus are all imported from its respective file. Since there are 10 restaurants, there are 10 txt files containing the menus of each restaurant respectively.

Import Price



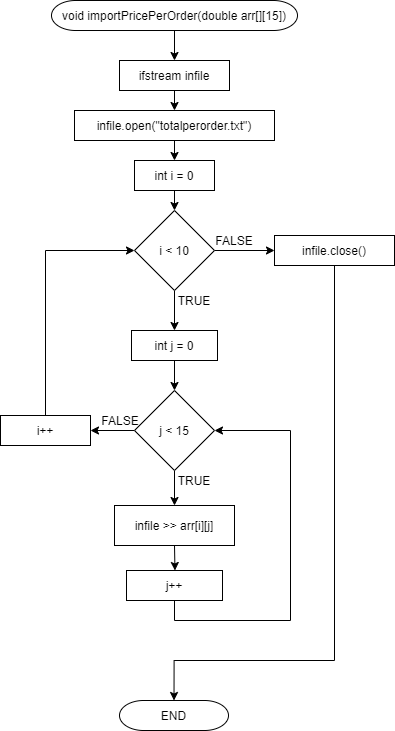
Import all of the prices of each menu of each restaurant into a two dimensional array from files

Import Delivery Time per Menu Import Delivery Time per Order



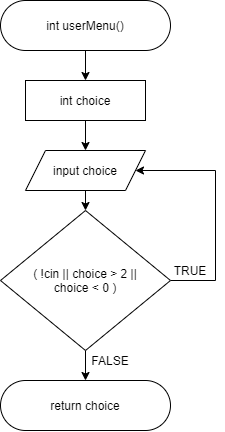
Import Delivery Time per Menu is to import all of the delivery time of each menu for each restaurant as all of the menu has different preparation time ,therefore different estimated delivery time. Import Deliver Time per Order is to import all of the delivery time for each order made to be accessed by restaurant managers later for the function Delivery()

Import Price Per Order



Import all the total price per order which was stored in a file, then stored into an array for it to be accessed later by restaurant managers in the function Payment()

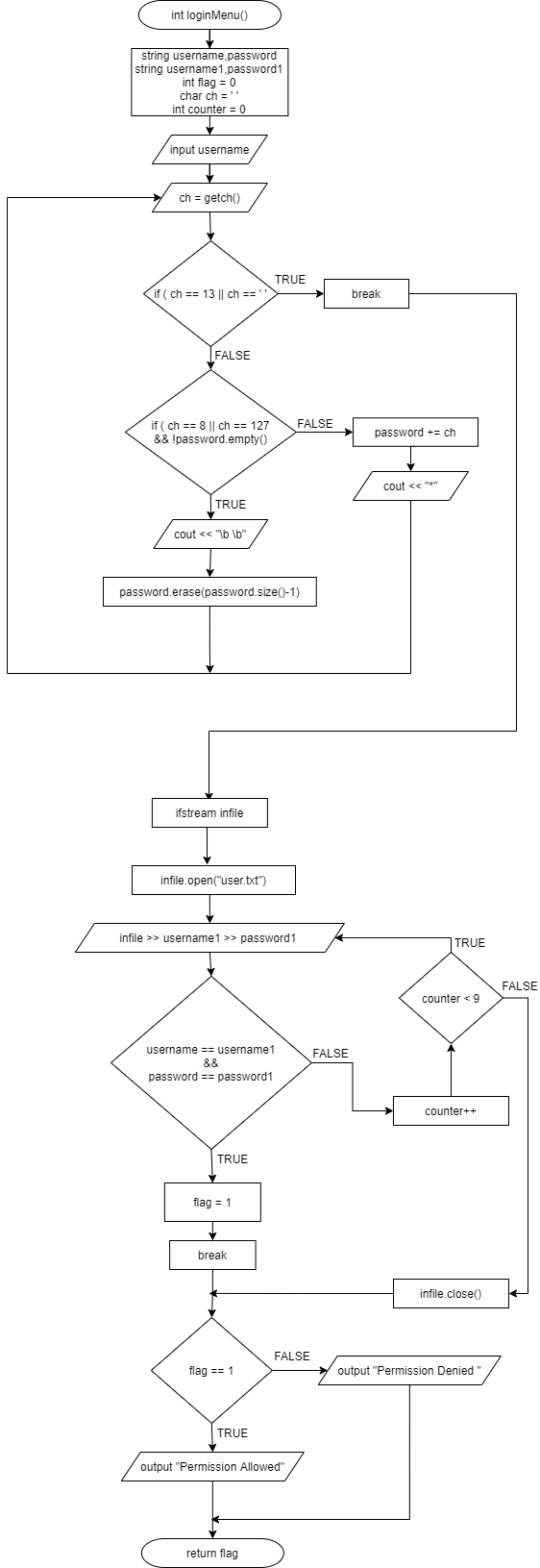
User Menu



User Menu is the first menu shown, to decide whether the user is a Restaurant Manager or a Customer

**Manager’s Function Flowcharts**

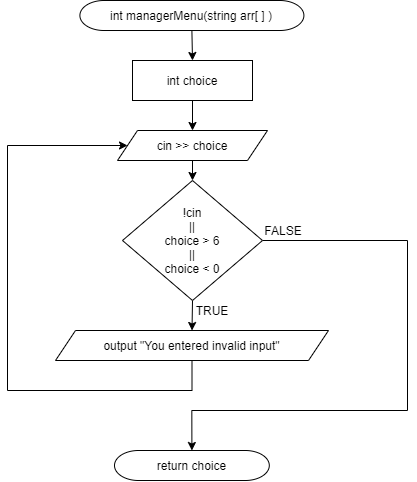
Login Menu



( For sharper image, refer to loginMenu.png included in the Flowchart folder )

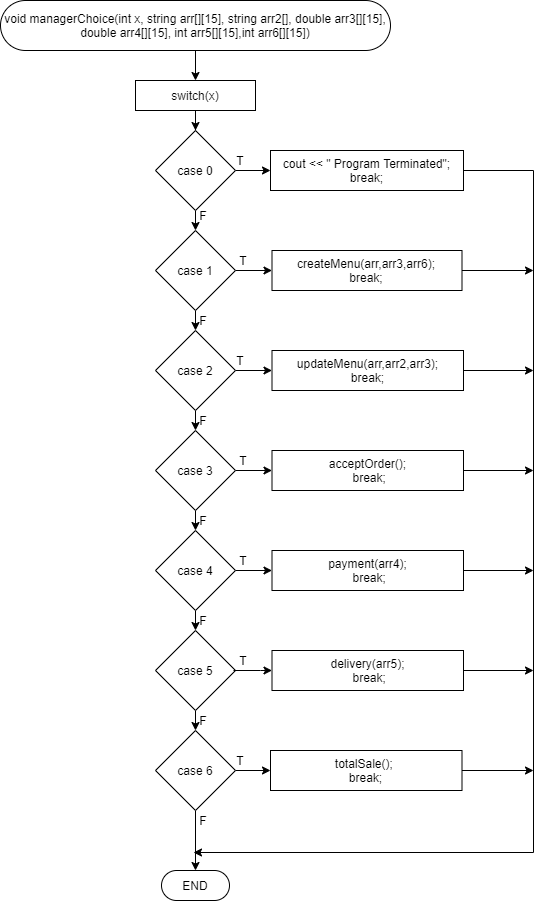
Login Menu is for the Restaurant Manager to input username and password of their account.

Manager Menu



Manager Menu is for the manager to choose on what to do, whether it is to create menu, update menu, accept order, display estimated delivery time, display total sale

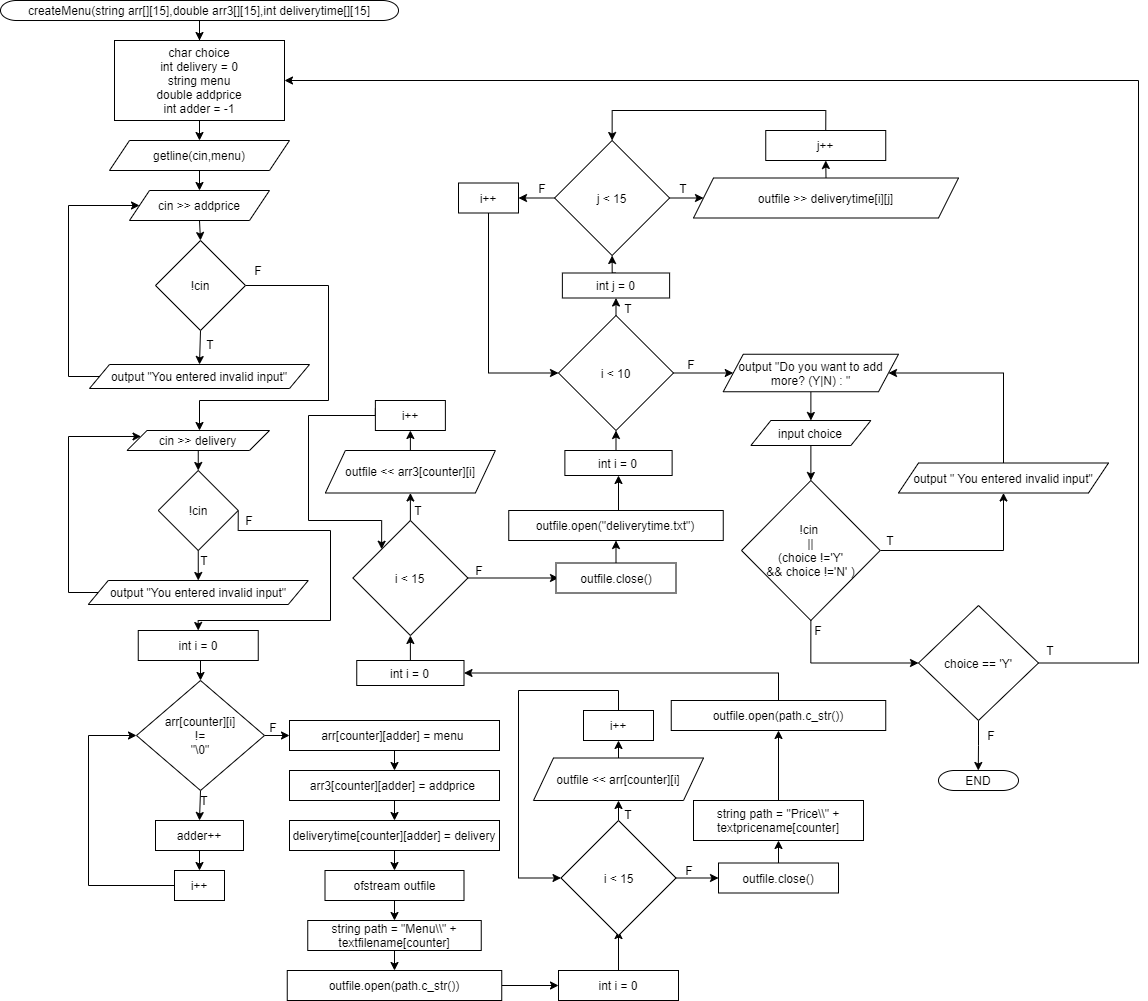
Manager Choice



( For sharper image, refer to managerChoice.png included in the Flowchart folder )

Manager Choice is the function that acts as the main of Manager’s functions. It calls functions accordingly to the choice input by user in the Manager Menu function

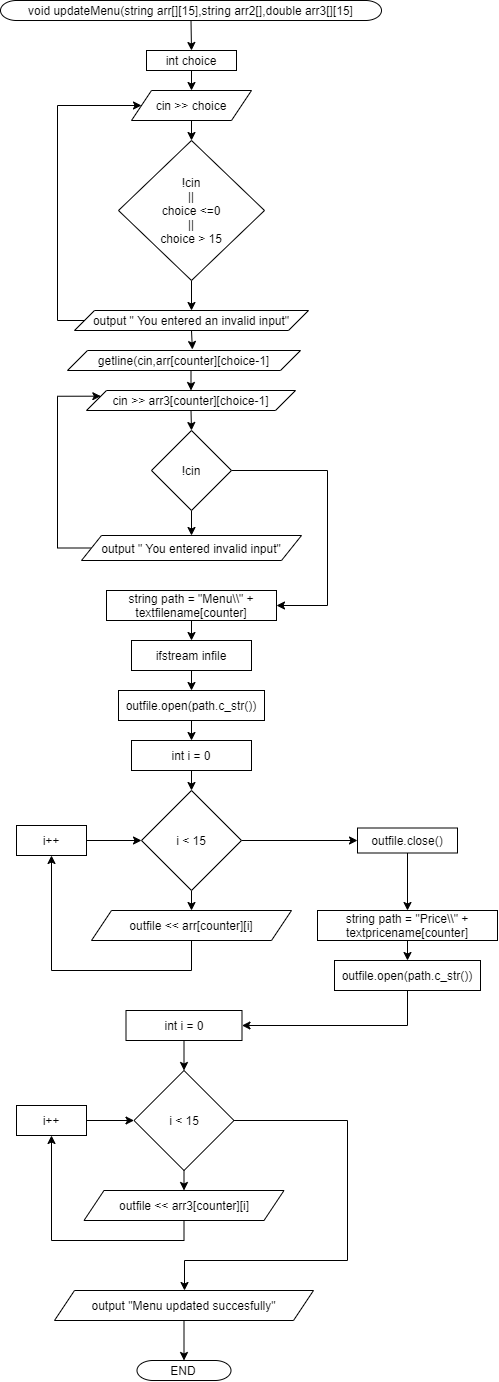
Create Menu



( For sharper image, refer to createMenu.png included in the Flowchart folder )

Create menu is a function that enables restaurant manager to add in new menus, their price, and estimated delivery time ( estimated time to prepare and deliver ) .

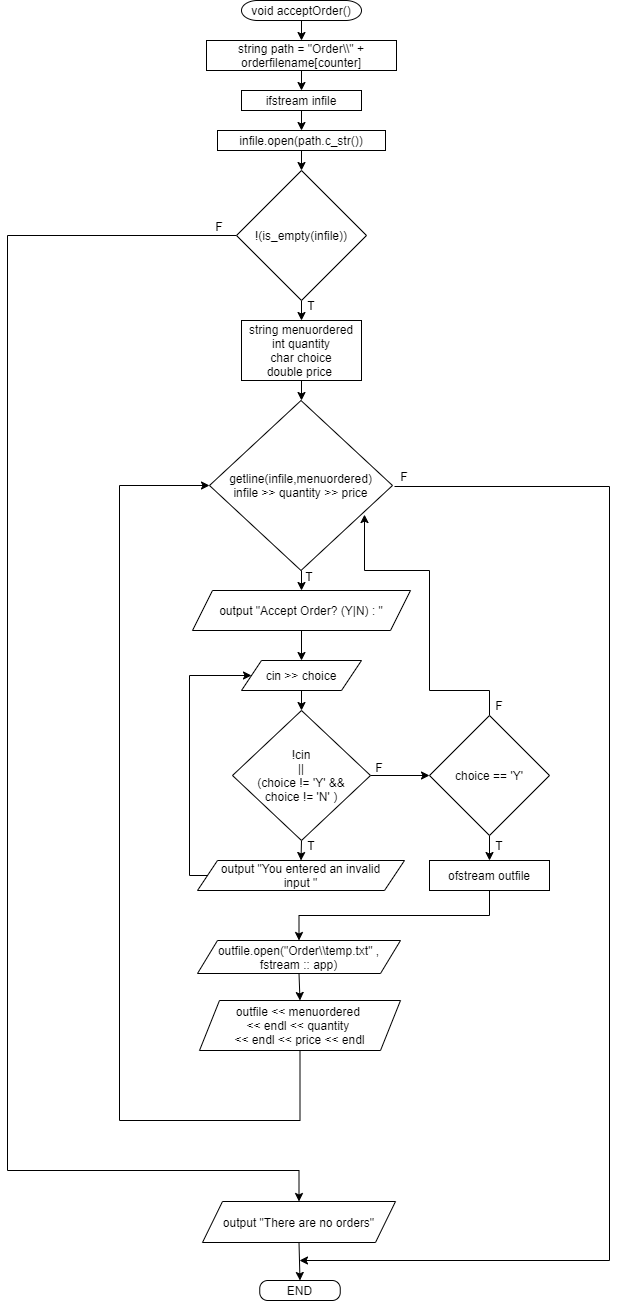
Update Menu



( For Sharper Image, refer to updateMenu.png included in the Flowchart folder )

Update Menu is a function that allows the manager to update old menus with new menus and their new price

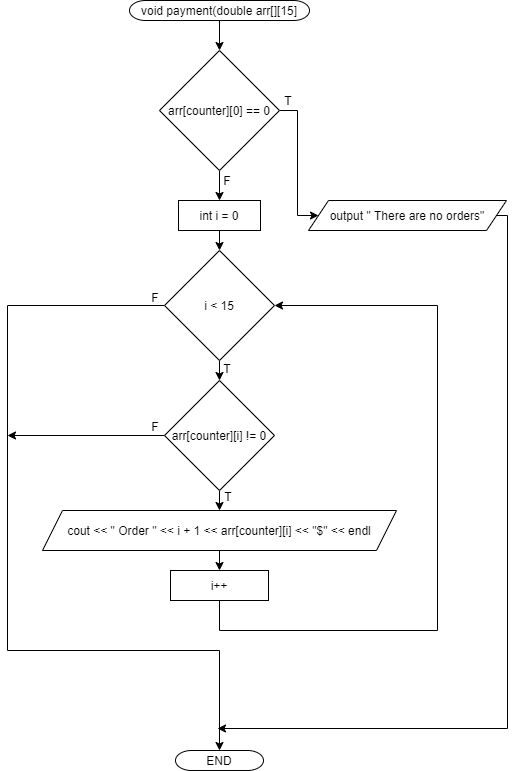
Accept/Reject Order



( For sharper image, refer to acceptOrder.png included in the Flowchart folder )

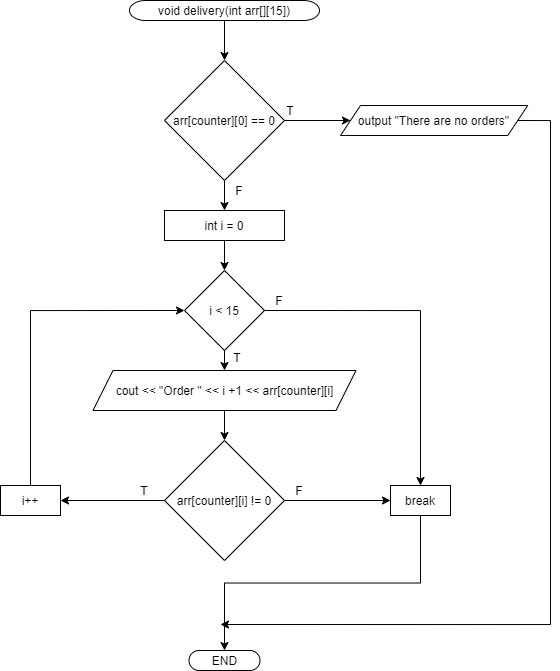
Accept Order function is a function that allows user to accept or reject orders made by users.

Payment



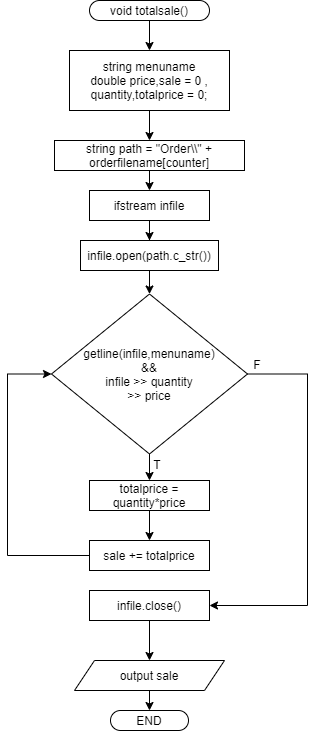
Payment is a function that display to the manager payments made by customers who ordered the foods from their restaurant

Delivery



Delivery is a function to shows the manager what is the estimated delivey time for each Orders made by customers.

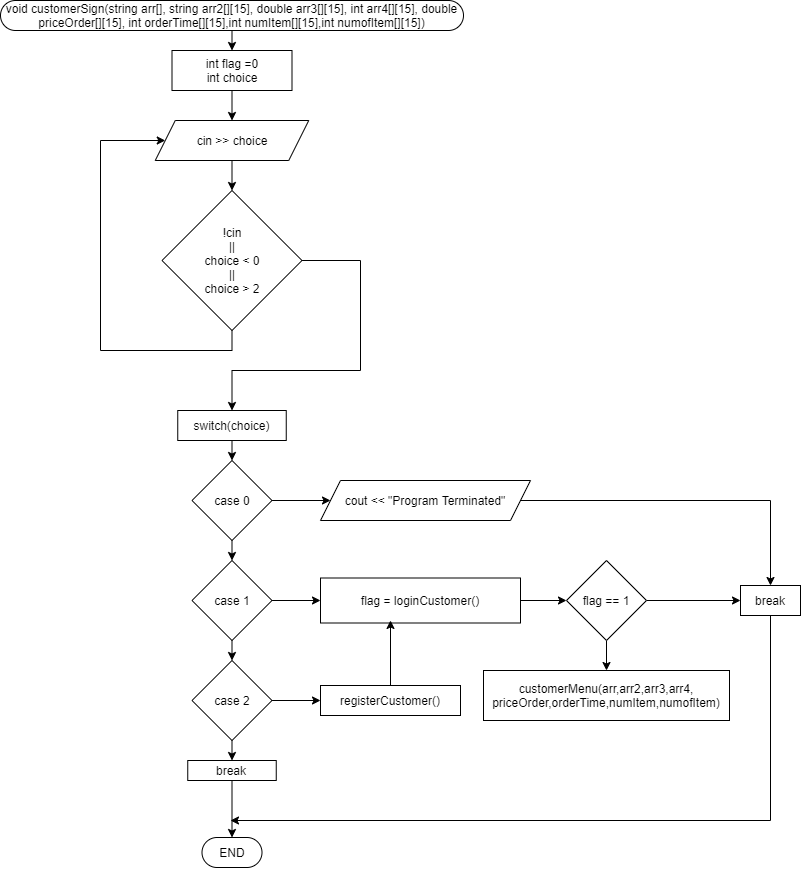
Total Sale



Total Sale is a function to calculate the total sales of the restaurant based on orders made by customers.

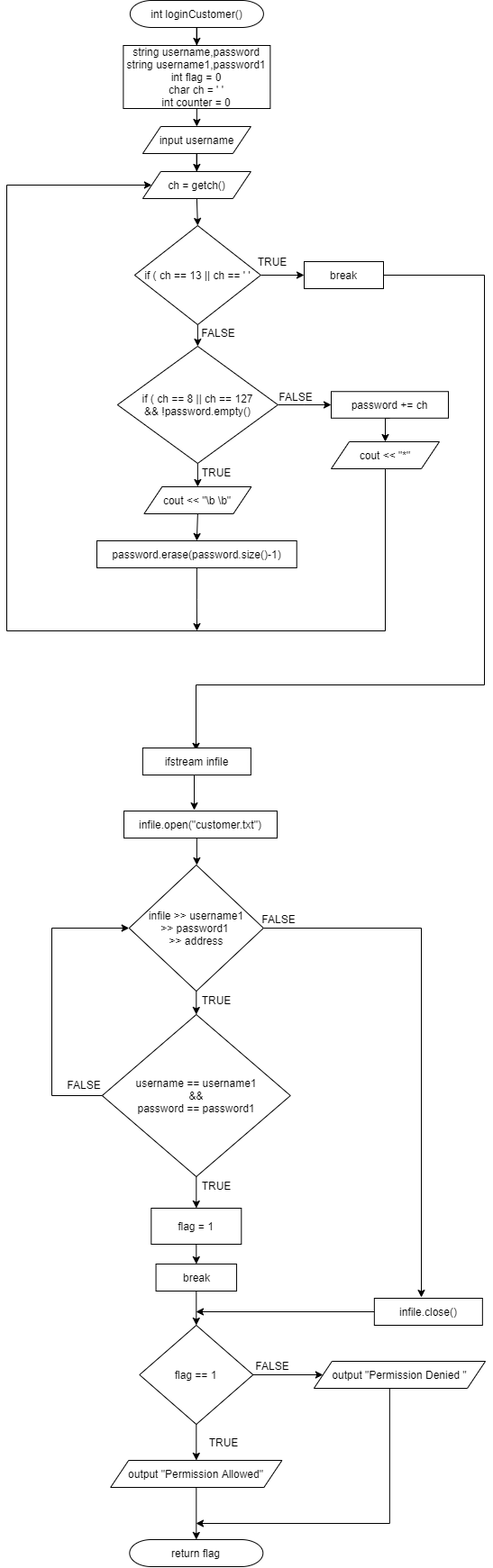
**Customer’s Function Flowchart**

Customer Sign



Customer Sign is a function that allows the user to register or login, which depends on whether they are a new user, or an existing user

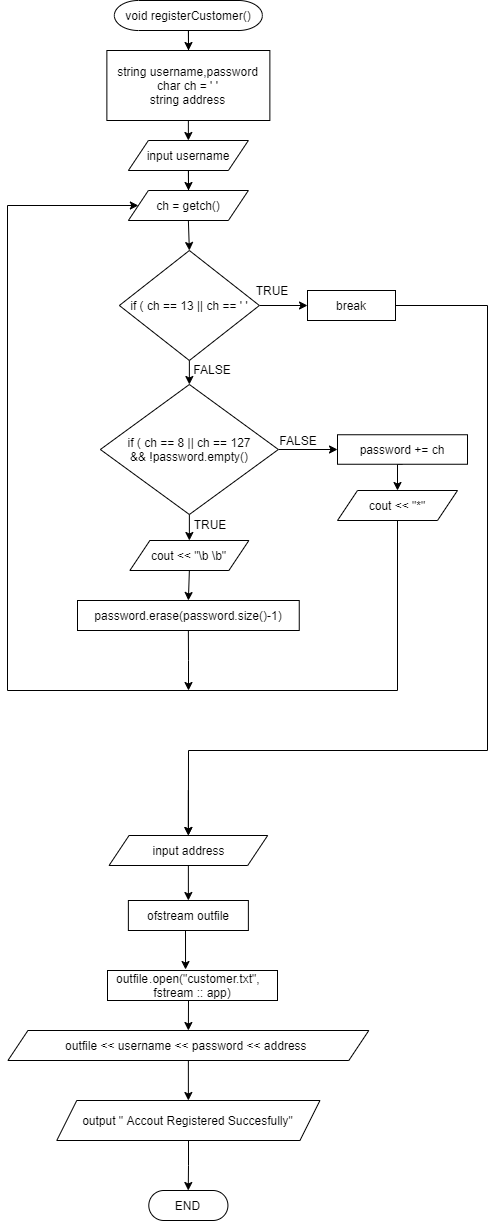
Login Customer



( For sharper image, refer to loginCustomer.png included in the Flowchart Folder )

Login Customer is a function that allows existing users ( customers ) in the system to login.

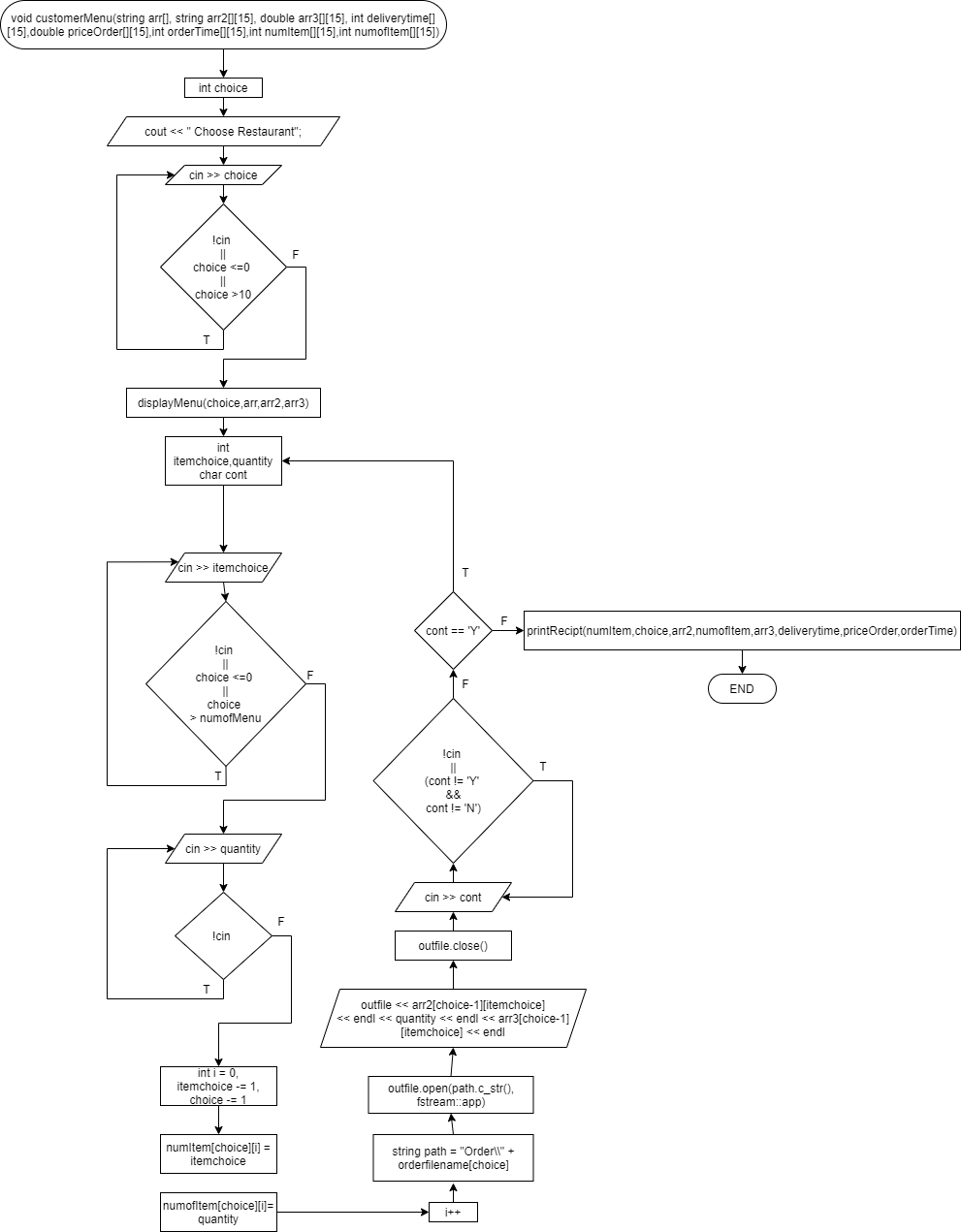
Register Customer



( For sharper image, refer registerCustomer.png which is included in the Flowchart Folder )

Register Customer is a function that allows new user to register an account to use the system.

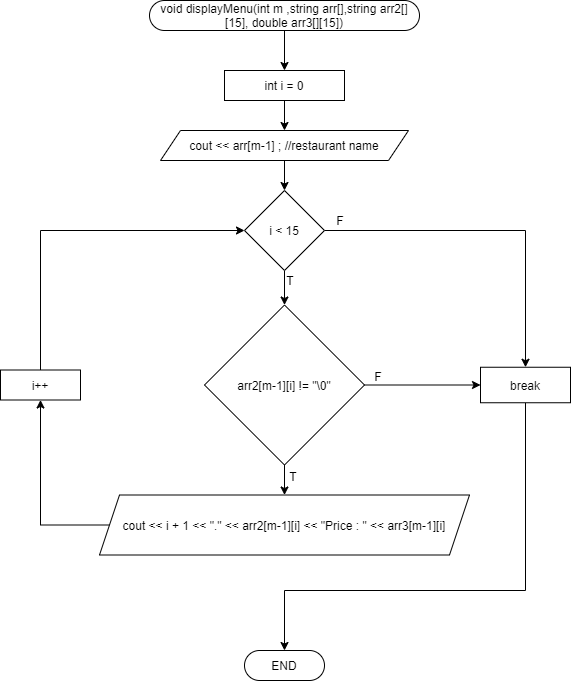
Customer Menu



( For sharper image, refer to customerMenu.png included in the Flowchart folder )

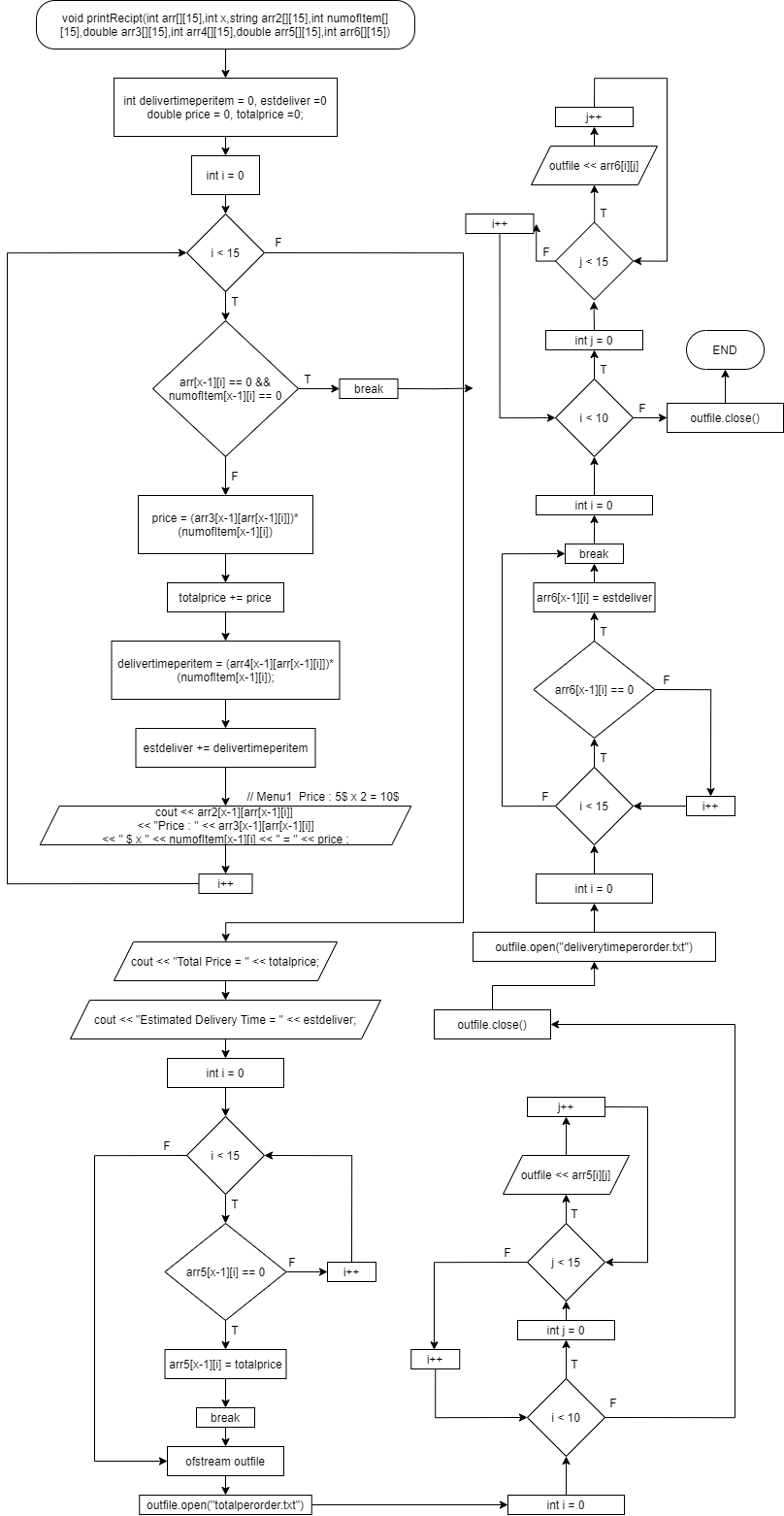
Customer Menu is essentially the main menu for customers. The user decides on which restaurant they want to order from which will then executes displayMenu based on user’s choice. The user then orders food and state the quantity of it. Finally, the printRecipt will execute which will print the Recipt for the user.

Display Menu



Displays the menus of the restaurant user has chosen.

Print Recipt



( For sharper image, refer to printRecipt.png included in the Flowchart folder )

Prints the recipt for user and store the user’s order in a file , to be later accessed by restaurant managers

**PROGRAM LISTING**

1. #include<iostream>
2. #include<fstream>
3. #include<iomanip>
4. #include<string>
5. #include<algorithm>
6. #include<iterator>
7. #include<conio.h>
8. #include<stdio.h>
9. #include<string.h>
10. #include<cmath>
11. #include<vector>
13. // disclaimer : string(int,char) is a function that repeats a char, int times. This is used a lot in the program for the UI , for example, string(100,'-') will create a string that consists of 100 '-'

16. **using** **namespace** std;
17. **int** counter = 0;
18. **int** counter2 = 0;
19. ofstream outfile;
20. ifstream infile;
22. //stores the filename of the restaurant menus so that it can be accessed later by ifstream or ofstream to input and output ( file handling )
23. string textfilename[10] = {"tiffzmenu.txt", "nanathaimenu.txt", "claypotmenu.txt" , "hwapohmenu.txt", "mamamiamenu.txt",
24. "kingmenu.txt", "gloriamenu.txt","chocolatmenu.txt", "mykorimenu.txt","breadhousemenu.txt"};
26. //stores the filename of all the prices for all different menus on each restaurant
27. string textpricename[10] = {"tiffzmenu.txt", "nanaprice.txt", "claypotprice.txt","hwapohprice.txt","mamamiaprice.txt",
28. "kingprice.txt","gloriaprice.txt","chocolatprice.txt","mykoriprice.txt","breadhouseprice.txt"};
30. //stores the filename of all orders made by customers for each different restaurant
31. string orderfilename[10] = {"tiffzorder.txt","nanaorder.txt","claypotorder.txt","hwapohorder.txt","mamamiaorder.txt",
32. "kingorder.txt","gloriaorder.txt","chocolatorder.txt","mykoriorder.txt","breadhouse.txt"};
34. //========================================================[Function Prototype]================================================================//
35. **int** userMenu();
36. **void** clear();
37. **void** importRestaurant(string arr[]);
38. **void** importMenu(string arr[][15]);
39. **void** importPrice(**double** arr[][15]);
40. **int** loginMenu();
41. **int** managerMenu(string arr[]);
42. **void** managerChoice(**int** x, string arr[][15], string arr2[], **double** arr3[][15], **double** arr4[][15], **int** arr5[][15],**int** arr6[][15]);
43. **void** importPricePerOrder(**double** arr[][15]);
44. **void** importDeliverTimePerOrder(**int** arr[][15]);
45. **void** importDeliverTimePerMenu(**int** arr[][15]);
46. **void** customerSign(string arr[], string arr2[][15], **double** arr3[][15], **int** arr4[][15], **double** priceOrder[][15], **int** orderTime[][15],**int** numItem[][15], **int** numofItem[][15]);
48. **bool** is\_empty(std::ifstream& pFile)
49. {
50. **return** pFile.peek() == std::ifstream::traits\_type::eof();
51. }
52. //============================================================================================================================================//
54. //==========================================================[Main]==========================================================================//
55. **int** main(){
56. **int** choice ,flag;
57. string restaurant[10]; // stores the name of restaurant
58. string menu[10][15]; // stores menus of each restaurant
59. **double** pricelist[10][15]; // stores the prices of each menu
60. **double** priceOrder[10][15] = {0}; //an array to store the total payment per order to display to the restaurant managers
61. **int** orderTime[10][15] = {0}; // stores the delivery time of each orders made
62. **int** deliverytime[10][15]; // stores the deliverytime of each menus
63. **int** numItem[10][15] = {0}; //intialize all of the array to 0 so that it can be managed easily [This array contains the item code of the menu chosen by user,this item code can be used to access the array of menu[10][15]
64. **int** numofItem[10][15] = {0}; // this array will store the quantity of order made of a certain item which corresponds to the array above , numItem[x][0] = 2 [Himitsu Pizza] and numofItem[x][0] = 1 [ the quantity of himitsu burger ordered ]
66. importRestaurant(restaurant); //importing and storing the restaurant names from the txt file into the array
67. importMenu(menu); // importing and storing the menus of the restaurant from the txt file into a 2d array
68. importPrice(pricelist); //importing and storing the prices of the menus of each restaurant
69. importPricePerOrder(priceOrder); //importing and storing all of the payment per order for each restaurant made so that it can be accessed by restaurant managers
70. importDeliverTimePerMenu(deliverytime);//importing delivery time for each menu of each restaurant
71. importDeliverTimePerOrder(orderTime); // importing delivery time for each order made


75. choice = userMenu(); // the user choose wheter they're a restaurant manager or a customer
76. clear(); // clear screen function is called
78. //a switch case to decide whether the user is a restaurant manager or customer based on choice picked on the userMenu() function
79. **switch**(choice)
80. {
81. //case 1 is for restaurant managers
82. **case** 1 :    flag = loginMenu(); // flag is used to indicate wheter the restaurant manager managed to Log In with their respective username and password
83. **if**(flag == 1){ // if the restaurant manager entered the correct username and password, the following code inside the if statement will be executed
84. **int** choice1; // declaration of choice to store the return value of managerMenu();
86. // a do-while loop so that the user won't exit the program after doing one operation, this ables the user to only exit the program when choice1 is 0
87. **do**{
88. choice1 = managerMenu(restaurant); //stores the value of user choice in the function managerMenu() so that the do-while loop can run correctly
89. managerChoice(choice1, menu, restaurant, pricelist, priceOrder,orderTime,deliverytime); //managerChoice function is called.It is a function that consists of switch-case to determine the operation the user want to do
90. }**while**(choice1 != 0);
91. }**else** // if the user didn't input the username and password correctly, access will be denied to the user and the program is terminated, therefore, break;
92. **break**;
93. **break**; // break; after all the operations in the case are done
95. //case 2 is for customers
96. **case** 2 :    customerSign(restaurant,menu,pricelist,deliverytime,priceOrder,orderTime,numItem,numofItem); **break**; //customerSign is a function that is called to prompt up a menu to ask wheter the user is an existing user, or a new user, basically a log in and sign up page
98. **case** 0 :    cout << " " << endl;cout << "  Program Terminated "; **break**;
99. }

102. }
104. //============================================================================================================================================//
106. **void** clear(){ //function to clear the console screen
108. system("cls");
109. }
111. //=============================================================[MAIN MENU]================================================================//
112. **int** userMenu(){
114. //userMenu is the first prompt in the console, which is why it is the Main Menu, the console output a menu and the user will choose whether they are a customer or a restaurant manager
115. **int** choice;
116. cout << "                            " << endl;
117. cout << "  --------------------------" << endl;
118. cout << "  |        MAIN MENU       | " << endl;
119. cout << "  --------------------------" << endl;
120. cout << "  | 1.Restaurant Manager   |" << endl;
121. cout << "  | 2.Customer             |" << endl;
122. cout << "  | 0.Quit                 |" << endl;
123. cout << "  --------------------------" << endl;
125. //input validation so that the user can only input 1,2,or 3 as an input, or else, it will go through a do-while loop asking for the input again until the input is entered correctly by user
126. cout << "  Choose : ";
127. **while**(!(cin >> choice) || choice > 2 || choice < 0){
128. cout << "\n  You entered an invalid input. Try again.";
129. cout << "\n  Choose : ";
130. cin.clear();
131. cin.ignore(123,'\n');
132. }

135. **return** choice; // return the choice to main ( for the switch-case )
136. }
137. //==========================================================================================================================================//

140. //==========================================[IMPORTING ALL THE INPUTS FROM TXT FILE]=====================================================//
141. **void** importRestaurant(string arr[]){
142. infile.open("restaurant.txt");
144. **for**(**int** i = 0 ; i<10 ; i++){
145. getline(infile, arr[i]);
146. }
147. infile.close();
148. }
150. **void** importMenu(string arr[][15]){
152. **for**(**int** i = 0 ; i < 10 ; i++){
153. string path = "Menu\\" + textfilename[i];
154. infile.open(path.c\_str());
155. **for**(**int** j = 0; j < 15 ; j++){
157. **if**(getline(infile, arr[i][j]) == "\0"){
158. **break**;
159. }
160. }
161. infile.close();
162. }

165. }
167. **void** importPrice(**double** arr[][15]){
169. **for**(**int** i = 0 ; i < 10 ; i++){
170. string path = "Price\\" + textpricename[i];
171. infile.open(path.c\_str());
172. **for**(**int** j = 0; j < 15 ; j++){
174. **if**((infile >> arr[i][j]) == "\0"){
175. **break**;
176. }
177. }
178. infile.close();
179. }

182. }
184. **void** importPricePerOrder(**double** arr[][15]){
185. infile.open("totalperorder.txt");
186. **for**(**int** i = 0; i < 10 ; i++){
187. **for**(**int** j = 0; j < 15 ; j++){
188. infile >> arr[i][j];
189. }
190. }
191. infile.close();
192. }
194. **void** importDeliverTimePerMenu(**int** arr[][15]){
195. infile.open("deliverytime.txt");
196. **for**(**int** i = 0 ; i < 10 ; i++){
197. **for**(**int** j = 0; j < 15 ; j++){
198. infile >> arr[i][j];
199. }
200. }
201. infile.close();
202. }
204. **void** importDeliverTimePerOrder(**int** arr[][15]){
205. infile.open("delivertimeperorder.txt");
206. **for**(**int** i = 0 ; i < 10 ; i++){
207. **for**(**int** j = 0; j <15 ; j++){
208. infile >> arr[i][j];
209. }
210. }
211. infile.close();
212. }
213. //====================================================================================================================================//


217. //========================================[FUNCTIONS FOR RESTAURANT MANAGER]===========================================================//
218. **int** loginMenu(){ // function for the Restaurant Manager to log in into their respective account, each restaurant have different accounts so that they can manage their own restaurant effectively
219. string username,password; //declaraction of username and password to be inputted by the user
220. string username1,password1; // declaraction of username1 and password1 which will be used to validate wheter the username and password are valid or not
221. **int** flag = 0; // stores a flag as 0
222. **char** ch = ' '; // used for the password inputting
224. //output the login menu
225. cout << "                            " << endl;
226. cout << "  --------------------------" << endl;
227. cout << "  |          LOGIN          | " << endl;
228. cout << "  --------------------------" << endl;
229. cout << "    Username : " ; cin >> username; // user input their username
230. cout << "    Password : " ;
231. //a function to display each character as \* when inputting the password for privacy
233. **do**{
234. ch = getch();
236. **if**(ch == 13 || ch == ' ')
237. {
238. **break**;
239. }
240. **if**((ch==8 || ch==127) && !password.empty()){
241. cout << "\b \b";
242. password.erase(password.size()-1);
243. }**else**{
244. password+=ch;
245. cout << "\*";
247. }
248. }**while**(ch != 13 || ch != ' ');

251. cout << endl << "  --------------------------" ;
253. //ifstream is used to open the user.txt file which stores all the username and passwords for all of the restaurant
254. infile.open("user.txt");
255. //a loop to input username1 and password1 again and again until the end of file
256. **while**(infile >> username1 >> password1){
257. **if**(username == username1 && password == password1){ // checks if the username and password input by user is valid by comparing to the username and password stored in the user.txt
258. flag = 1; // allocate value 1 to flag if the above statement is correct
259. **break**; //break is used so that when the username and password is found, the loop end as there is no longer the need to check until the end of file
260. }**else**
261. counter++; //counter is a global variable that is used to count how many time the loop repeats so that we can use it to access the restaurants and menus later because the username and password are stored chronologically to the restaurant in the user.txt file
262. }
264. infile.close(); // close the user.txt file
266. **if**(flag == 1){ // if the username and password are valid, output the statement below
267. cout << "\n                    ";
268. cout << "\n  Permission Allowed";
269. cout << "\n  Press Enter To Continue...";
270. cin.ignore();
271. cin.get(); // makes the user press Enter for the program to continue
272. clear(); //clear the console
273. }
274. **else**{ // if the username and password are invalid , output the statement below
275. cout << "\n                    ";
276. cout << "\n  Permission Denied";
277. }
279. **return** flag; // return the flag to the main
280. }
282. **int** managerMenu(string arr[]){ // managerMenu is basically the menu displayed after the restaurant manager logged in. The value passed ( string arr[] ) is the array that contains the restaurant name
283. **int** choice;
284. **int** l = arr[counter].length(); // determine the length of the name of the restaurants because the table/menu displayed varies in size for each restaurant, this is to make the allignment of the menu more clean
286. **int** tablel = l\*3; // the size of the table/menu is 3 times the length of the restaurant name
287. cout << "                                       " << endl;
288. cout << "  " << string(tablel+1,'-') <<endl;
289. cout << "  |"<< setw(l) <<""<< arr[counter] << setw((tablel-l)-l) << "|"<<  endl; //setw() is also used for allignment purposes
290. cout << "  " << string(tablel+1, '-') << endl;
291. //display all the choices of operation the restaurant manager can choose
292. cout << "  |" <<  " "<<"1.Create Menu" << setw(tablel-14)<<"|" <<endl;
293. cout << "  |" <<  " "<<"2.Update Menu" << setw(tablel-14)<<"|" <<endl;
294. cout << "  |" <<  " "<<"3.Accept Orders" << setw(tablel-16)<<"|" <<endl;
295. cout << "  |" <<  " "<<"4.Payment" << setw(tablel-10)<<"|" <<endl;
296. cout << "  |" <<  " "<<"5.Delivery" << setw(tablel-11)<<"|" <<endl;
297. cout << "  |" <<  " "<<"6.Total Sale" << setw(tablel-13)<<"|" <<endl;
298. cout << "  |" <<  " "<<"0.Quit" << setw(tablel-7)<<"|" <<endl;
299. cout << "  " << string(tablel+1,'-') << endl;
300. cout << endl;
301. cout << "\n  Choose : ";
302. **while**(!(cin >> choice) || choice > 6 || choice < 0){ // input validation
303. cout << "\n  You entered an invalid input. Try again.";
304. cout << "\n  Choose : ";
305. cin.clear();
306. cin.ignore(123,'\n');
307. }
309. **return** choice; //the choice is returned to main to be sent into another function which is managerChoice() for a switch-case
310. }
312. **void** createMenu(string arr[][15], **double** arr3[][15], **int** deliverytime[][15]){ //create menu is the function to add in a menu into the menu of the restaurant, this is case 1
313. **char** choice;
314. **int** delivery = 0;
315. **do**{ // a do while loop so that the user can enter as many new menu as they want
316. string menu; // declared to store the name of the menu want to be created/added
317. **double** addprice; //declared to store the price of the new added menu
318. **int** adder=0; // adder is declared to determine the index of the new menu and new price in the array of menu[10][15] and pricelist[10][15]
319. adder -= 1; // to make adder value to be -1 because the index will count from 0,
320. outfile.open("Menu\\temp.txt"); // a temp.txt which is a temporary file is created
321. cin.ignore();
322. cout << "                                " << endl;
323. cout << "  Enter menu name to be added : " ;
324. getline(cin,menu); // user input the new menu to be added
325. cout << "  Enter price of menu : ";
327. **while**(!(cin >> addprice)){ // input validation
328. cout << "\n  You entered an invalid input. Try again.";
329. cout << "\n  Enter price of menu : ";
330. cin.clear();
331. cin.ignore(123,'\n');
332. }
333. cout << "  Enter estimated delivery time for this menu : ";
334. **while**(!(cin >> delivery)){ // input validation
335. cout << "\n  You entered an invalid input. Try again.";
336. cin.clear();
337. cin.ignore(123,'\n');
338. }
340. //cin >> addprice; // user input the price for the new menu to be added
342. **for**(**int** i = 0 ; i < 15 ; i++){ // a for loop to store the new values input
343. **if**(arr[counter][i] != "\0"){ // since the array size is [10][15] , but the only data stored are [10][10] so far, the are 5 values that are not initialised for every restaurant. \0 is the value of an empty line, the statement is that if the value is not empty, output the value to the temp.txt
344. outfile << arr[counter][i] << endl;
345. adder++; // used to keep track of the index
346. }
347. **else** **if**(arr[counter][i] == "\0"){ // when an empty line in the array is found, it will be allocated by the new menu, therefore,the new menu will be inserted into the array without deleting or replacing any old values
348. outfile << menu;
349. arr[counter][i] = menu;
350. adder++; // used to keep track of the index
351. **break**; // break when the new menu is output into the temp.txt file
352. }
353. }
354. // arr3 is the array of priceList which stores the prices of the menus of each respective restaurant
355. arr3[counter][adder] = addprice; //since adder was used above the keep track of the index that is supposed to store the new menu,the price can be easily stored in the same index as the new menu as they all correspond as i organised the file accordingly to the menu and price
356. outfile.close();
358. //now that the price is stored in the array, it will be useless if it isn't outputted to a file to be stored
359. outfile.open("Price\\temp.txt"); // open a new file named temp.txt in the folder Price
360. **for**(**int** i = 0 ; i < 15 ; i++){
361. outfile << arr3[counter][i] << endl; //output every pricelist into the file
362. }
363. outfile.close();
364. //for the PRICE
365. string def4 = "Price\\";
366. string pathnameprice = def4 + textpricename[counter]; // this method is used because each prices are stored in different files for each restaurant, thefore this method is used for flexibility of the program to output to the correct file respective to the restaurant
367. remove(pathnameprice.c\_str()); // remove the original pricelist file
368. rename("Price\\temp.txt", pathnameprice.c\_str()); // rename the temp.txt file to the name of the original pricelist file
370. //for delivery time of the new menu
371. **for**(**int** i = 0 ; i < 15 ; i++){
372. **if**(deliverytime[counter][i]==0){
373. deliverytime[counter][i] = delivery;
374. **break**;
375. }
376. }
377. outfile.open("deliverytime.txt");
378. **for**(**int** i = 0 ; i < 10 ; i++){
379. **for**(**int** j = 0 ; j < 15 ; j++){
380. outfile << deliverytime[i][j] << endl;
381. }
382. }
383. outfile.close();
385. cout << "  Menu added succesfully." << endl;
386. cout << "  Do you want to add more menu? (Y|N): ";
387. // user input Y or N to determine whether they want to add more menus or end their order
388. **while**(!(cin >> choice) || (choice != 'Y'  && choice != 'N')){ // input validation
389. cout << "\n  You entered an invalid input. Try again.";
390. cout << "\n  Do you want to add more menu? (Y|N) : ";
391. cin.clear();
392. cin.ignore(123,'\n');
393. }
394. //for the MENU
395. string def = "Menu\\";
396. string pathname = def + textfilename[counter];
397. remove(pathname.c\_str());
398. rename("Menu\\temp.txt", pathname.c\_str());
400. }**while**(choice == 'Y');
402. cout << "\n  Press any key to go back to main menu ...";
403. cin.ignore();
404. cin.get();
405. clear();
406. }
408. **void** updateMenu(string arr[][15], string arr2[], **double** arr3[][15]){ //a function so that the restaurant manager can update the menu, and their price
410. **int** choice;
411. //============for menu size allignment optimization/flexibility=============
412. **int** l = arr2[counter].length();
413. **int** tablel = l\*3;
414. //=========================================================================
415. cout << "                                       " << endl;
416. cout << "  " << string(tablel+1,'-') <<endl;
417. cout << "  |"<< setw(l) <<""<< arr2[counter] << setw((tablel-l)-l) << "|"<<  endl;
418. cout << "  " << string(tablel+1, '-') << endl;
419. **int** j = 0;
420. **for**(**int** i = 0 ; i<15 ; i++){ // a for loop to print the menus of the restaurant and the prices of the menus respectively
421. **int** character = arr[counter][i].length();
423. **if**(arr[counter][i] != "\0"){
424. j = j + 1; // used to count how many menus are inside the array so that it can be used for input validation later
425. **if**(i+1 < 10){ // check if the the number of the menu is less than 10 or not because the allignment are different because (1-9) are 1 digit meanwhile (10 and above ) are two digits
426. **if**(arr3[counter][i] < 10){ // check if the price of the menu is less that 10 or not because the allignment are different
427. cout << "  |" << " " << i+1 << "  - " << arr[counter][i] <<right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[counter][i] << "$"<< setw(tablel-(6+character+(tablel-character-15)+5)) <<"|" << endl;
428. }**else**{
429. cout << "  |" << " " << i+1 << "  - " << arr[counter][i] <<right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[counter][i] <<"$"<< setw(tablel-(6+character+(tablel-character-15)+6)) <<"|" << endl;
430. }
431. }**else**{
432. **if**(arr3[counter][i] < 10){
433. cout << "  |" << " " << i+1 << " - " << arr[counter][i] << right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[counter][i] <<"$" << setw(tablel-(5+character+(tablel-character-15)+6)) << "|" << endl;
434. }**else**{
435. cout << "  |" << " " << i+1 << " - " << arr[counter][i] <<right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[counter][i] <<"$"<< setw(tablel-(6+character+(tablel-character-15)+6)) <<"|" << endl;
436. }
437. }
439. }
440. }
441. cout << "  " << string(tablel+1,'-') << endl;
442. cout << endl;
443. cout << "  Choose the menu number you want to update : ";
444. **while**(!(cin >> choice) || (choice <= 0 || choice > j)){ // input validation
445. cout << "\n  You have entered an invalid input. Try again.";
446. cout << "\n  Choose the menu number you want to update : ";
447. cin.clear();
448. cin.ignore(123,'\n');
450. }


454. cin.ignore();
455. cout << "  New name of the menu : ";
456. getline(cin, arr[counter][choice-1]); // user input the new name of the menu to be updated, it replace the old menu in the menu array, choice-1 is used as index because the user choose the menu from [1- ?] , An array is counted from [0-?]
457. cout << "  Price of the updated menu : ";
458. // user input the new price for the menu to be updated , it replace the old price in the priceList array
459. **while**(!(cin >> arr3[counter][choice-1])){ // input validation
460. cout << "\n  You entered an invalid input. Try again.";
461. cout << "\n  Price of the updated menu : ";
462. cin.clear();
463. cin.ignore(123,'\n');
464. }
465. cout << "  Menu Updated Succesfuly";
466. cout << "\n  Press any key to return to main menu ...";
467. cin.ignore();
468. cin.get();

471. outfile.open("Menu\\temp.txt"); // open/create a temp.txt which is a temporary file in the menu folder
473. **for**(**int** i = 0 ; i < 15 ; i++){
474. **if**(arr[counter][i] == "\0"){ // if the array at index i is empty, break the loop
475. **break**;
476. }**else**{ // else if the array at index i is not empty, output it into the temp.txt file, this temp.txt file stores the latest value ( updated value )
477. outfile << arr[counter][i] << endl;
478. }
479. }
480. outfile.close(); // file is closed
482. string def1 = "Menu\\";
483. string pathname1 = def1 + textfilename[counter];
484. remove(pathname1.c\_str()); // remove the original menu file
485. rename("Menu\\temp.txt", pathname1.c\_str()); // rename the temp.txt file to the name of the original menu file
487. outfile.open("Price\\temp.txt"); // create/open a temp.txt file which is a temporary file in the Price folder
489. **for**(**int** i = 0 ; i < 15 ; i++){
490. outfile << arr3[counter][i] << endl; // output all of the latest price to the temp.txt file
491. }
492. outfile.close();
493. string def2 = "Price\\";
494. string pathname2 = def2 + textpricename[counter];
495. remove(pathname2.c\_str()); // the original price file is removed
496. rename("Price\\temp.txt" , pathname2.c\_str()); // the temp.txt file is renamed to the original price file name
497. clear(); // clear console
499. }
501. **void** acceptOrder(){
502. string path = "Order\\" + orderfilename[counter]; // store the string of the path of the order file, each restaurant have its own respective order file, which is why this method is used
503. infile.open(path.c\_str()); // open the order file of the restaurant
504. outfile.open("Order\\temp.txt", fstream::app); // create/open a temp file in the folder Order
506. **if**(!(is\_empty(infile))){
507. string menuordered; // used to store the name of the menu ordered by customers
508. **int** quantity; // store the quantity of the menu ordered
509. **char** choice;
510. **double** price; // store the price of the menu
511. cout << "  Orders" << setw(20) << " "<<"Quantity" << endl;
512. //a loop to read the file opened until the end of file
513. **while**(getline(infile,menuordered) && infile >> quantity >> price){
514. cout << " " << endl;
515. cout << "  " << menuordered << setw(30-(menuordered.length())) << " " << quantity << endl; // output the name of the menu ordered ( which is stored in the file order ) and the quantity ordered
516. cout << "  Accept Order? (Y|N) : " ;  // asks the user wheter to accept the order or not

519. **while**(!(cin >> choice) || (choice != 'Y'  && choice != 'N')){ // input validation
520. cout << "\n  You entered an invalid input. Try again.";
521. cout << "\n  Accept Order? (Y|N) : ";
522. cin.clear();
523. cin.ignore(123,'\n');

526. }
527. **if**(choice=='Y'){ // if the user input Y , ( accept the order ) , the order will be outputted into the temp.txt file, else, if the user pick N (Decline the order ), the order will not be outputted
528. outfile << menuordered << endl << quantity << endl << price << endl;
529. }
531. infile.ignore();
532. }
534. infile.close();
535. outfile.close();
537. remove(path.c\_str()); // remove the original order file menu
538. rename("Order\\temp.txt", path.c\_str()); // rename the temp.txt to the original order file name so that it stores the latest values because accepted orders are stored , meanwhile declined orders are esentially removed
539. clear();
540. }**else**{
541. cout << " " << endl;
542. cout << "  There are no orders.";
543. cout << "\n  Press Enter to go back to main menu ...";
544. cin.clear();
545. cin.ignore();
546. **if**(cin.get()){
547. clear();
548. }
549. }
550. }
552. **void** payment(**double** arr[][15]){ // a function to show the restaurant managers the payment/price per order made ( each customer has different price per order made )
553. **if**(arr[counter][0] == 0){
554. cout << " " << endl;
555. cout << "  There are no orders." << endl;
556. }**else**{
557. cout << " " << endl;
558. **for**(**int** i = 0 ; i < 15 ; i++){
559. **if**(arr[counter][i] != 0){ // priceOrder is a global array that is used to store the total price per order which is calculated in a Customer's function , the statement below will run aslong as the value inside the array is not empty
560. cout << "  Order " << i+1 << " : " << fixed << setprecision(2) << arr[counter][i] << "$" << endl;
561. }
562. }
563. }
565. cout <<"  Press Enter to go back to the Main Menu ...";
566. cin.ignore();
567. cin.get();
568. clear();
570. }
572. **void** delivery(**int** arr[][15]){
573. **if**(arr[counter][0] == 0){
574. cout << " " << endl;
575. cout << "  There are no orders." << endl;
576. }**else**{
577. cout << " " << endl;
578. cout << "  Estimated Delivery Time : " << endl;
579. **for**(**int** i = 0; i < 15 ; i++){
580. **if**(arr[counter][i] != 0){
581. cout << "  Order " << i+1 << " : " << fixed << setprecision(2) << arr[counter][i] << " minutes" << endl;
582. }
583. }
584. }
585. cout << "  Press Enter to go Back to Main Menu ..." ;
586. cin.ignore();
587. cin.get();
588. clear();
589. }
591. **void** totalSale(){ // a function to calculate the total sale of the restaurant
592. string menuname;
593. **double** price;
594. **double** sale = 0; // initialise the sale as 0
595. **double** quan; // shortform of quantity to store the quantity of the food ordered
596. **double** totalprice = 0; // initialise the total price as 0, total price is used to store the value of price\*quan which is The food price x The Quantity Ordered
598. string path = "Order\\" + orderfilename[counter];
599. infile.open(path.c\_str());

602. //loop to read until the end of file
603. **while**(getline(infile,menuname) && infile >> quan >> price){
604. infile.ignore();
605. totalprice = quan\*price; // calculate the total price of an item ordered ( because quantities and price varies for each item ordered )
606. sale += totalprice; // sale is the total of the totalprice of each item
608. }
610. infile.close();
611. cout << "                    " << endl;
612. cout << fixed << setprecision(2) << "  Total Sale : " << sale << "$";
613. cout << "\n  Press Enter to return to Menu ...";
614. cin.ignore();
615. cin.get();
616. clear();

619. }
621. **void** managerChoice(**int** x, string arr[][15], string arr2[], **double** arr3[][15], **double** arr4[][15], **int** arr5[][15],**int** arr6[][15]){ //managerChoice is esentially the main() function for the Restaurant Manager because it calls out all the functions above ( functions for restaurant manager )
622. clear();
624. **switch**(x)
625. {
626. **case** 0 : cout << endl << "  Program Terminated."; **break**; // if the passed value x which is choice made at managerMenu() is 0, Program Terminated
627. **case** 1 : createMenu(arr,arr3,arr6);**break**; // if the user chooses to Create Menu, createMenu function is called
628. **case** 2 : updateMenu(arr,arr2,arr3); **break**; // if the user chooses to update menu, updateMenu function is called
629. **case** 3 : acceptOrder(); **break**; // if the user chooses to accept Order, acceptOrder function is called
630. **case** 4 : payment(arr4); **break**; // if the user chooses the payment option, payment function is called
631. **case** 5 : delivery(arr5); **break**; // if the user chooses delivery option, delivery function is called
632. **case** 6 : totalSale(); **break**; // if the user chooses Total Sale, totalSale function is called
634. }

637. }
638. //======================================================================================================================================//


642. //=================================================[CUSTOMER FUNCTIONS]================================================================//
644. **void** displayMenu(**int** m ,string arr[],string arr2[][15], **double** arr3[][15]){ // a function that displays the menu of restaurant
645. **int** l = arr[m-1].length(); // menu flexibility purposes
646. **int** tablel = l\*3;
647. cout << "                                       " << endl;
648. cout << "  " << string(tablel+1,'-') <<endl;
649. cout << "  |"<< setw(l) <<""<< arr[m-1] << setw((tablel-l)-l) << "|"<<  endl; // cout the restaurant name at the top of the menu
650. cout << "  " << string(tablel+1, '-') << endl;
651. **for**(**int** i = 0 ; i<15 ; i++){
652. **int** character = arr2[m-1][i].length(); // the length of each menu names are initialised for aliggnment menu flexibility
654. **if**(arr2[m-1][i] != "\0"){
655. counter2 = i+1;
656. **if**(i+1 < 10){
657. **if**(arr3[m-1][i] < 10){
658. cout << "  |" << " " << i+1 << "  - " << arr2[m-1][i] <<right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[m-1][i] << "$"<< setw(tablel-(6+character+(tablel-character-15)+5)) <<"|" << endl;
659. }**else**{
660. cout << "  |" << " " << i+1 << "  - " << arr2[m-1][i] <<right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[m-1][i] <<"$"<< setw(tablel-(6+character+(tablel-character-15)+6)) <<"|" << endl;
661. }
662. }**else**{
663. **if**(arr3[m-1][i] < 10){
664. cout << "  |" << " " << i+1 << " - " << arr2[m-1][i] << right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[m-1][i] <<"$" << setw(tablel-(5+character+(tablel-character-15)+6)) << "|" << endl;
665. }**else**{
666. cout << "  |" << " " << i+1 << " - " << arr2[m-1][i] <<right << setw(tablel-character-15) << " " << fixed << setprecision(2) << arr3[m-1][i] <<"$"<< setw(tablel-(6+character+(tablel-character-15)+6)) <<"|" << endl;
667. }
668. }
670. }
671. }

674. cout << "  " << string(tablel+1,'-') << endl;
675. cout << endl;
676. }
678. **int** loginCustomer(){ //function for the customer to login to the system ( the same login menu as restaurant manager loginmenu
679. clear();
680. string username,password;
681. string address;
682. string username1,password1;
683. **int** flag = 0;
684. **char** ch;
685. cout << "                            " << endl;
686. cout << "  -----------------------------------------------" << endl;
687. cout << "  |                    LOGIN                    |" << endl;
688. cout << "  -----------------------------------------------" << endl;
689. cout << "    Username : " ; cin >> username;
690. cout << "    Password : " ;
692. **do**{
693. ch = getch();
695. **if**(ch == 13 || ch == ' ')
696. {
697. **break**;
698. }
699. **if**((ch==8 || ch==127) && !password.empty()){
700. cout << "\b \b";
701. password.erase(password.size()-1);
702. }**else**{
703. password+=ch;
704. cout << "\*";
706. }
708. }**while**(ch != 13 || ch != ' ');
710. cout << endl << "  -----------------------------------------------" ;
712. infile.open("customer.txt");
713. **while**(infile >> username1 >> password1 && getline(infile,address)){
714. **if**(username == username1 && password == password1){
715. flag = 1;
716. **break**;
717. }
718. }
719. **if**(flag == 1){
720. cout << "\n                    ";
721. cout << "\n  Permission Allowed";
722. cout << "\n  Press Enter To Continue...";
723. cin.ignore();
724. cin.get();
726. clear();
727. }
728. **else**{
729. cout << "\n                    ";
730. cout << "\n  Permission Denied";
731. }
733. **return** flag;
734. }
736. **void** registerCustomer(){ // function to register a new user to the system
737. string username,password;
738. string address;
739. clear();
740. **int** flag = 0;
741. **char** ch;
742. cout << "                            " << endl;
743. cout << "  -----------------------------------------------" << endl;
744. cout << "  |                   REGISTER                  |" << endl;
745. cout << "  -----------------------------------------------" << endl;
746. cout << "    Username  : " ; cin >> username; //user input their username
747. cin.clear();
748. cin.ignore(123,'\n');
749. cout << "    Password  : " ;
750. //user input password ( the program will display it every character input as \*  for privacy )=============
751. **do**{
752. ch = getch();
754. **if**(ch == 13 || ch == ' ')
755. {
756. **break**;
757. }
758. **if**((ch==8 || ch==127) && !password.empty()){
759. cout << "\b \b";
760. password.erase(password.size()-1);
761. }**else**{
762. password+=ch;
763. cout << "\*";
765. }
767. }**while**(ch != 13 || ch != ' ');
769. //=======================================================================================================
771. cout << "\n    Address   : ";
772. getline(cin,address); //user input their delivery address
773. cout << endl << "  -----------------------------------------------" ;
775. outfile.open("customer.txt" , fstream::app); // customer.txt is opened to store the login details of all the user, which is why fstream::app is used so that it will not overwrite everything inside
776. outfile << username << endl << password; // output the username and password to the file
777. outfile <<  endl << address << endl; // output the address to the file
778. outfile.close();

781. outfile.close();
782. cout << "\n  Account Registered Succesfully!" << endl;
783. cout << "\n  Press Enter to go to Login Menu ..." ;
784. cin.get();
785. clear();
787. }
788. //DELIVERYTIME         //PRICEORDER         //ORDERTIME
789. **void** printRecipt(**int** arr[][15],**int** x,string arr2[][15],**int** numofItem[][15],**double** arr3[][15],**int** arr4[][15],**double** arr5[][15],**int** arr6[][15]){
790. clear();
791. **int** delivertimeperitem = 0,estdeliver = 0;
792. **double** price=0,totalprice=0;
794. **int** l = 20;
795. **int** tablel = l\*3;
796. cout << "                                       " << endl;
797. cout << "  " << string(tablel+2,'-') <<endl;
798. cout << "  |"<< setw(l+6) <<""<< "RECIPTS" << setw((tablel-l)-12) << "|"<<  endl;
799. cout << "  " << string(tablel+2, '-') << endl;
801. **for**(**int** i = 0 ; i<15 ; i++){
802. **int** character = arr2[x-1][arr[x-1][i]].length(); // for alignment flexibility use
803. **if**(arr[x-1][i] == 0 && numofItem[x-1][i] == 0){
804. **break**;
805. }**else**{

808. price = (arr3[x-1][arr[x-1][i]])\*(numofItem[x-1][i]); //THE PRICE IS PRICE OF ITEM X THE QUANTITY OF ITEM,
809. totalprice += price; // the total price of all the items are the total of the price of the items
810. delivertimeperitem = (arr4[x-1][arr[x-1][i]])\*(numofItem[x-1][i]);
811. estdeliver += delivertimeperitem;
812. //alignmment flexibility/problem which is why if and else are used a lot to make sure the menu allign properly with different values of price, etc
813. **if**(arr3[x-1][arr[x-1][i]] < 10){
814. **if**(price < 10)
815. cout << "  |" << " " << i+1 << "  - " << arr2[x-1][arr[x-1][i]] <<right << setw(tablel-character-27) << " " << fixed << setprecision(2) << arr3[x-1][arr[x-1][i]] << "$ x " << numofItem[x-1][i] << " = "<< price << "$" << setw(tablel-(6+character+(tablel-character-15)+4)) <<"|" << endl;
816. **else**
817. cout << "  |" << " " << i+1 << "  - " << arr2[x-1][arr[x-1][i]] <<right << setw(tablel-character-27) << " " << fixed << setprecision(2) << arr3[x-1][arr[x-1][i]] << "$ x " << numofItem[x-1][i] << " = "<< price << "$" << setw(tablel-(6+character+(tablel-character-15)+5)) <<"|" << endl;
818. }**else**{
819. cout << "  |" << " " << i+1 << "  - " << arr2[x-1][arr[x-1][i]] <<right << setw(tablel-character-28) << " " << fixed << setprecision(2) << arr3[x-1][arr[x-1][i]] << "$ x " << numofItem[x-1][i] << " = "<< price << "$" << setw(tablel-(6+character+(tablel-character-15)+5)) <<"|" << endl;
820. }
821. }
823. }
824. cout << "  " << string(tablel+2,'-') << endl;
825. cout << "  Total Price = " << totalprice << endl; //print out the total price of the orders
826. cout << "  Estimated Delivery Time = " << estdeliver << " minutes";
828. **for**(**int** i =0 ; i < 15 ; i++){
829. **if**(arr5[x-1][i] == 0){ //priceOrder( arr5 ) is an that stores the payment per order price,since the price per order is only inputted one at a time in a single run, it only allocate the array if the value is 0 ( empty )
830. arr5[x-1][i] = totalprice; //totalprice is stored in the array priceOrder
831. **break**;
832. }
833. }
834. outfile.open("totalperorder.txt");
835. **for**(**int** i = 0 ; i < 10 ; i++){
836. **for**(**int** j = 0 ; j < 15 ; j++){
837. outfile << arr5[i][j] << endl; // output the pricerperOrder to a file to be accesed later
838. }
839. }
840. outfile.close();
842. **for**(**int** i = 0 ; i< 15 ; i++){
843. **if**(arr6[x-1][i] == 0){
844. arr6[x-1][i] = estdeliver; // estimated delivery time is stored in the arr6
845. **break**;
846. }
847. }
849. outfile.open("delivertimeperorder.txt");
850. **for**(**int** i = 0; i<10 ; i++){
851. **for**(**int** j = 0 ; j<15 ; j++){
852. outfile << arr6[i][j] << endl; // estimated delivery time is output into a file for restaurant manager to access later
853. }
854. }
855. outfile.close();
856. cout << endl;

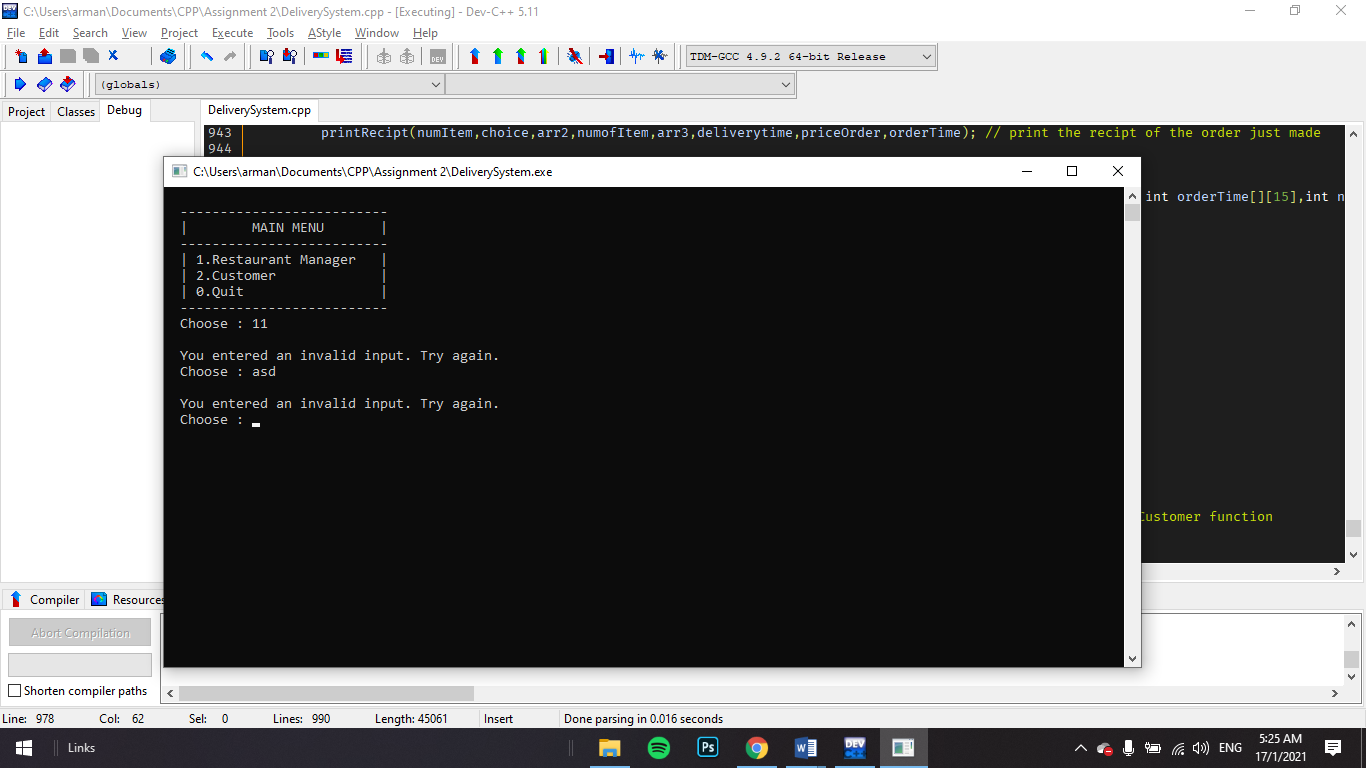
859. }
861. **void** customerMenu(string arr[], string arr2[][15], **double** arr3[][15], **int** deliverytime[][15],**double** priceOrder[][15],**int** orderTime[][15],**int** numItem[][15],**int** numofItem[][15]){
863. **int** choice = 0;
864. **int** j = 1;
865. //======table size optimization/for flexibility of menu==========//
866. **int** l = 20;
867. **int** tablel = l\*3;
868. //================================================================//
869. cout << "                                       " << endl;
870. cout << "  " << string(tablel+2,'-') <<endl;
871. cout << "  |"<< setw(l+2) <<""<< "RESTAURANTS" << setw((tablel-l)-12) << "|"<<  endl;
872. cout << "  " << string(tablel+2, '-') << endl;
873. **for**(**int** i = 0 ; i<10 ; i++){
874. **int** character = arr[i].length();
875. **if**(arr[i] != "\0"){ //if the value of array at the index is not empty, execute below statement , the array passed is restaurant[] which contains the restaurant names
876. //if and else is used for allignment purpose because the number 1-9 are 1 digit meanwhile 10 and above are two digit which will have different allignment settings
877. **if**(i+1 < 10){
878. cout << "  |" << " " << i+1 << "  - " << arr[i]  << setw(tablel-(character+6)) << " " <<"|" << endl;
879. }**else**{
880. cout << "  |" << " " << i+1 << " - " << arr[i]  << setw(tablel-(character+6)) << " " << "|" << endl;
881. }
883. }**else** **if**(arr[i] == "\0"){ //this ensures that the loop stop when the value inside the array is empty, we only want to print out all of the restaurant names, not print out empty lines
884. **break**;
885. }
886. }
887. cout << "  " << string(tablel+2,'-') << endl;
888. cout << endl;
889. cout << "  Choose restaurant : ";
890. **while**(!(cin >> choice) || choice <= 0 || choice > 10){
891. cout << "\n  You have entered an invalid input. Try again.";
892. cout << "\n  Choose restaurant : ";
893. cin.clear();
894. cin.ignore(123,'\n');
896. }
897. // user choose the restaurant they want to order from
899. clear();
900. displayMenu(choice,arr,arr2,arr3); // displayMenu function is called to display the Menu of the restaurant user picked
902. **int** itemchoice;
903. **int** quantity;
904. **char** cont; //shortform of continue
905. string path = "Order\\" + orderfilename[choice-1]; // order file of the respective restaurant is opened to store the orders of the customer
906. outfile.open(path.c\_str(), fstream::app); //fstream:: app is so that the file is not overwritten everytime a user orders, instead, it will append in the file
907. **int** i = 0; //initialize i as 0 to input in the array of numItem and numofItem in the do-while loop
908. **do**{
909. cout << "  Choose an item : ";
910. // the user chooses the item displayed by displayMenu function
911. **while**(!(cin >> itemchoice) || (itemchoice <= 0 || itemchoice > counter2 ) ){
912. cout << "\n  You have entered an invalid input. Try again.";
913. cout << "\n  Choose an item : ";
914. cin.clear();
915. cin.ignore(123,'\n');
916. }
917. itemchoice = itemchoice - 1;
918. numItem[choice-1][i] = itemchoice; //choice-1 is used because the number displayed in the menu are 1-10 while an array are counted from 0-9, the choice of item the user picked are stored in the array to be accesed later in another function
919. cout << "  Quantity? : ";
920. //user input the quantity of the item want to be ordered
921. **while**(!(cin >> quantity)){
922. cout << "\n  You have entered an invalid input. Try again.";
923. cout << "\n  Quantity? : ";
924. cin.clear();
925. cin.ignore(123,'\n');
926. }
927. numofItem[choice-1][i] = quantity; // stored in the array numofItem to be accessed later
928. i++; // i is incremented
929. outfile << arr2[choice-1][itemchoice] << endl << quantity << endl << arr3[choice-1][itemchoice] << endl; // output all of the choices made into the order file
930. cout << "  Continue(Y|N) : "; // asks the user if they want to add more orders or stop
931. // user input Y or N
932. **while**(!(cin >> cont) || (cont != 'Y' && cont != 'N')){
933. cout << "\n  You have entered an invalid input. Try again.";
934. cout << "\n  Continue(Y|N) : ";
935. cin.clear();
936. cin.ignore(123,'\n');
937. }
939. }**while**(cont == 'Y'); // continue the loop aslong as user picks Y

942. outfile.close(); //close file
943. printRecipt(numItem,choice,arr2,numofItem,arr3,deliverytime,priceOrder,orderTime); // print the recipt of the order just made
945. }
947. **void** customerSign(string arr[], string arr2[][15], **double** arr3[][15], **int** arr4[][15], **double** priceOrder[][15], **int** orderTime[][15],**int** numItem[][15],**int** numofItem[][15]){ // esentially the first menu of customer and also kind of act like the main for the customer functions
948. **int** flag=0;
949. **int** choice;
950. cout << "                            " << endl;
951. cout << "  --------------------------" << endl;
952. cout << "  |      CUSTOMER MENU     | " << endl;
953. cout << "  --------------------------" << endl;
954. cout << "  | 1.Existing User        |" << endl;
955. cout << "  | 2.New User             |" << endl;
956. cout << "  | 0.Quit                 |" << endl;
957. cout << "  --------------------------" << endl;
959. cout << "\n  Choose : ";
960. **while**(!(cin >> choice) || (choice < 0 || choice > 2)){
961. cout << "\n  You entered an invalid input, Try again.";
962. cout << "\n  Choose : ";
963. cin.clear();
964. cin.ignore(123,'\n');
965. }
967. //a swtich case, if the user an existing user (case 1) the console will prompt up login menu in the LoginCustomer function
968. **switch**(choice)
969. {
970. **case** 1 : flag = loginCustomer(); // login menu for customer
971. **if**(flag == 1){ //if the username and password input by user is correct, the Menus / Restaurant options will be prompted
972. customerMenu(arr,arr2,arr3,arr4,priceOrder,orderTime,numItem,numofItem); // prompt Restaurant options the user can choose
973. }
974. **else**
975. **break**;
976. **break**;
977. **case** 2 : registerCustomer(); // if the user is a new user, the console will prompt up a register menu first, then store the login detail in customer.txt file so that the user can login later
978. flag = loginCustomer(); // login after account is registered
979. **if**(flag == 1){ // if the username and password input by user is valid, the menus/restaurant option will be prompted
980. customerMenu(arr,arr2,arr3,arr4,priceOrder,orderTime,numItem,numofItem);
981. }
982. **else**{
983. **break**;
984. }
985. **break**;
986. **case** 0 : clear();cout << " " << endl ; cout << "  Program Terminated" ; **break**;
987. }
988. }
990. //================================================================================================================================//

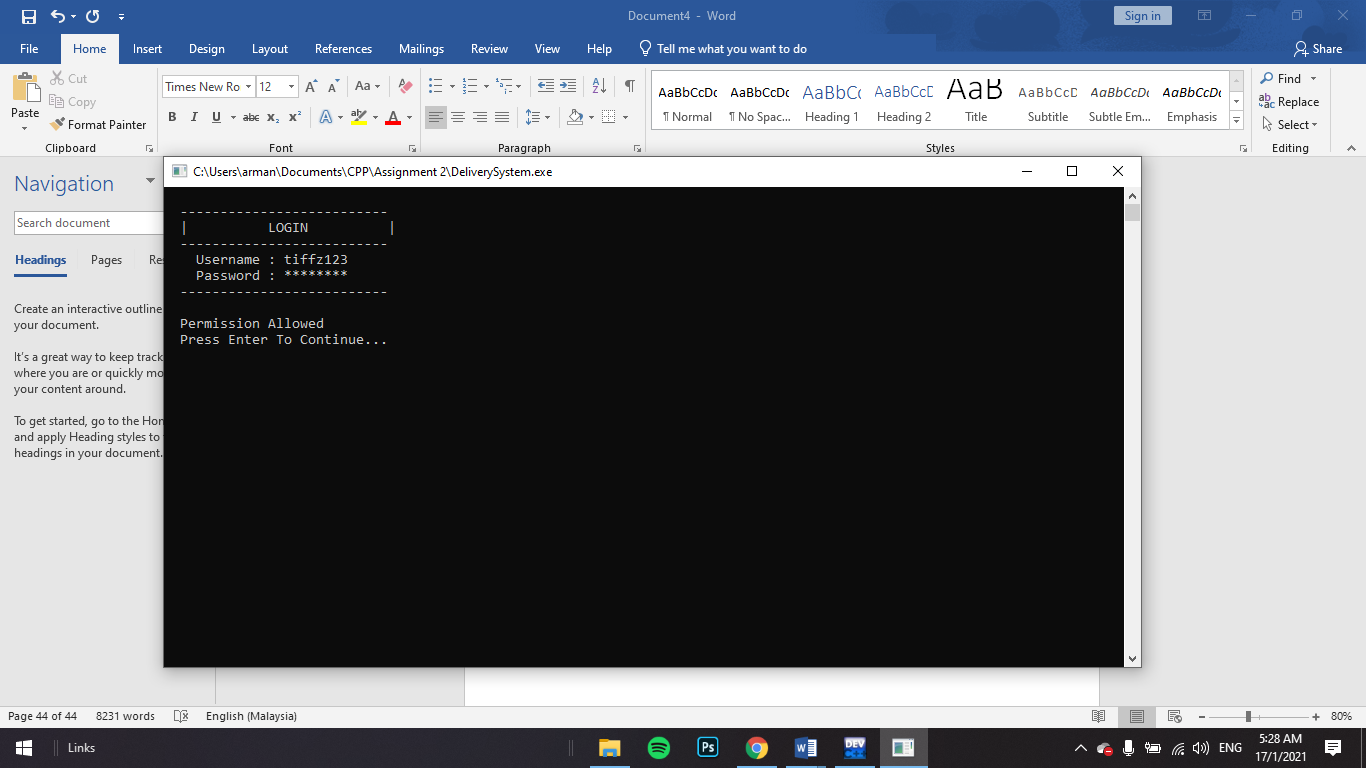
**Input Output of The Program**

Main Page

1. **User choose Restaurant Manager or Customer**
2. Input Validation

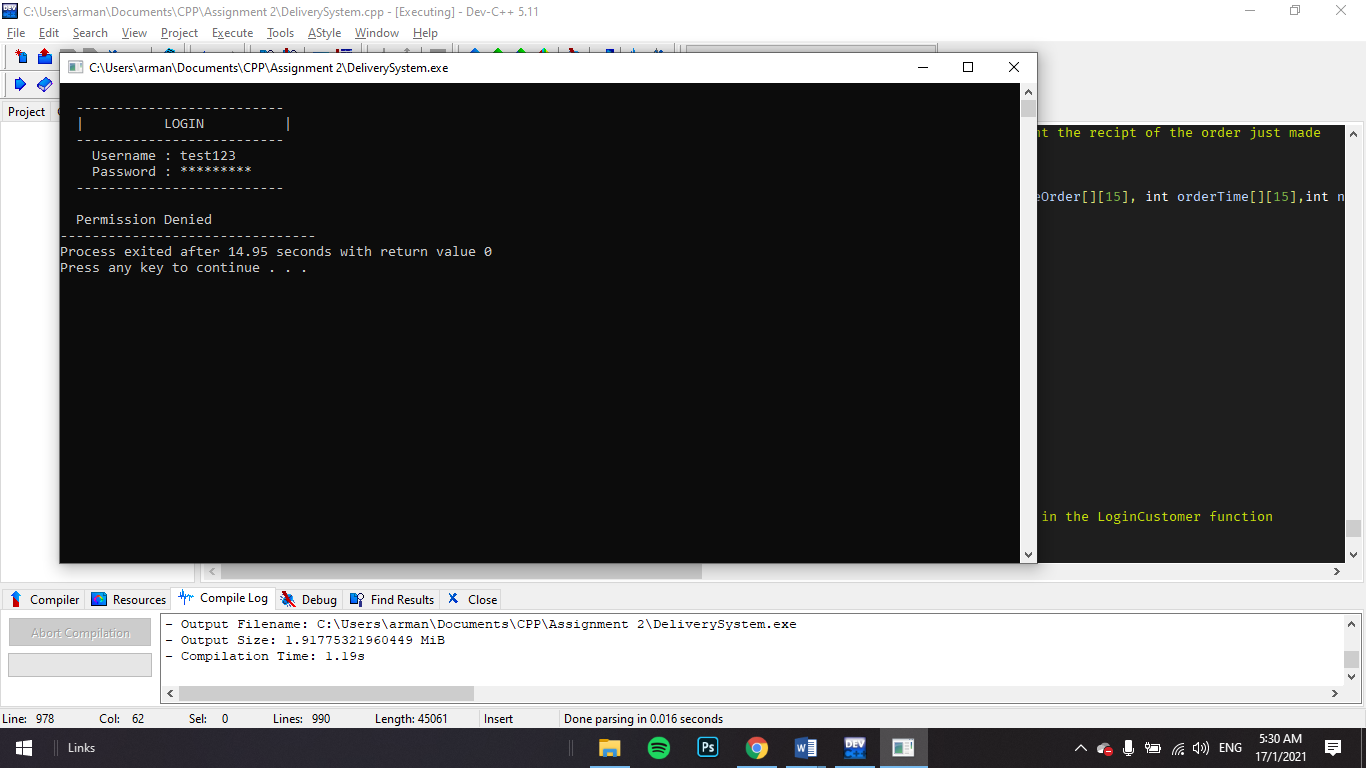


1. **User choose Restaurant Manager**
   1. Correct login details

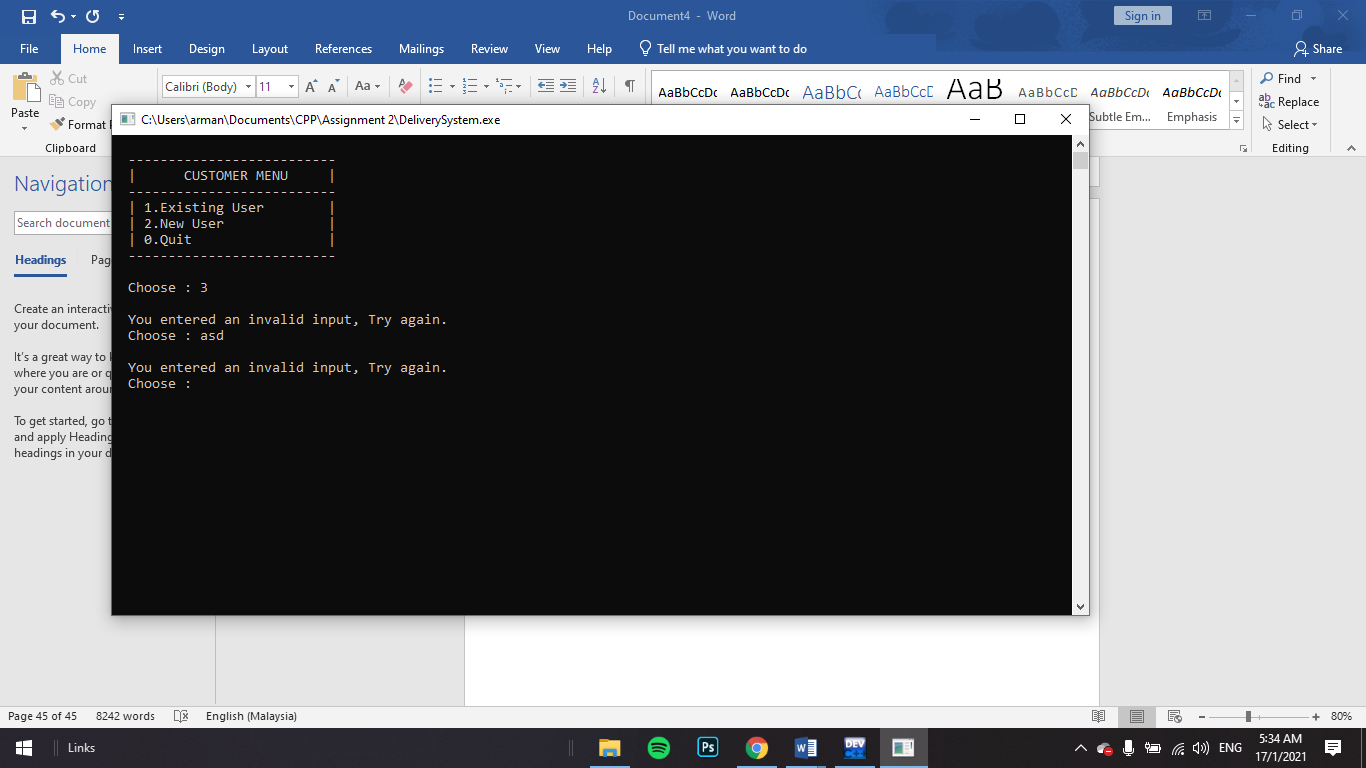


( For Login Details of Restaurant, refer to Logindetails.pdf file included )

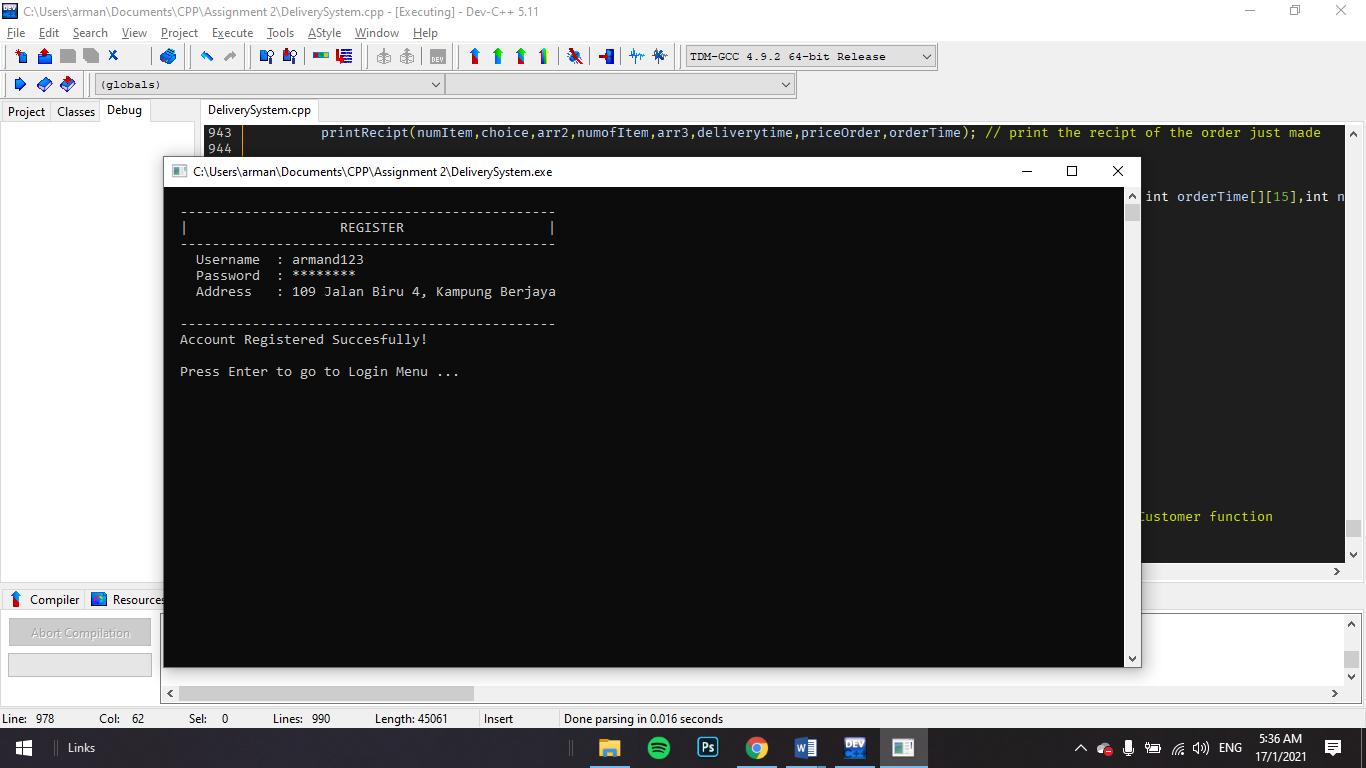
* 1. Incorrect login details



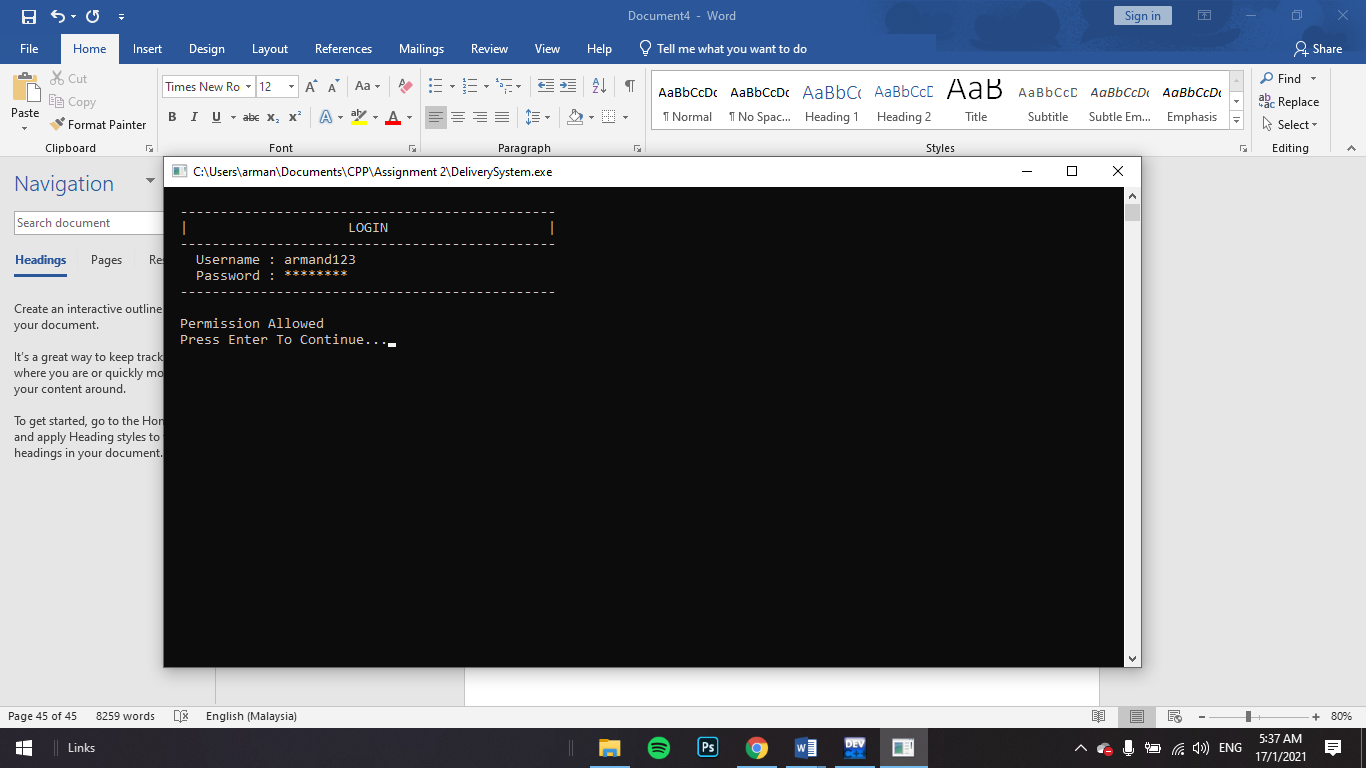
1. **User Choose Customer**
   1. Input validation



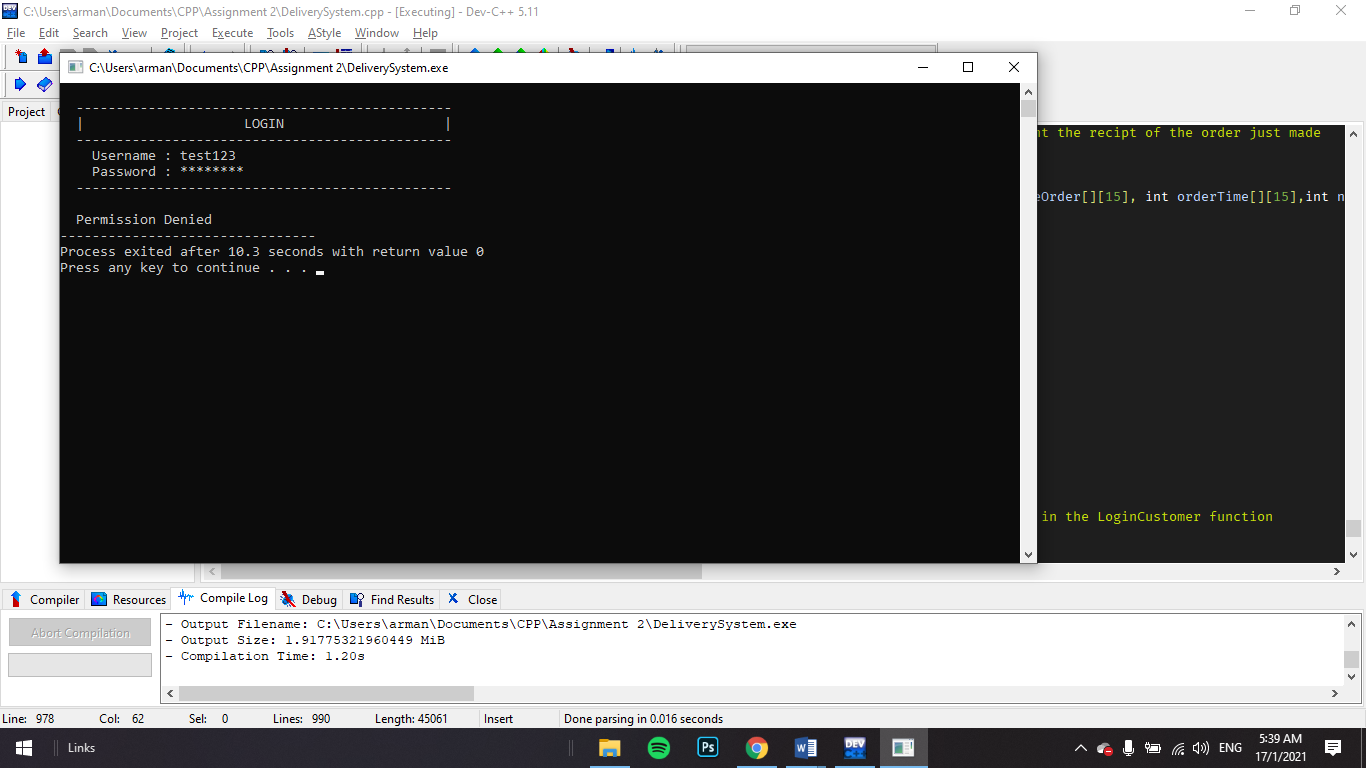
1. **User Choose New User**
   1. User register



1. **User Choose Existing User**
   1. Correct login details

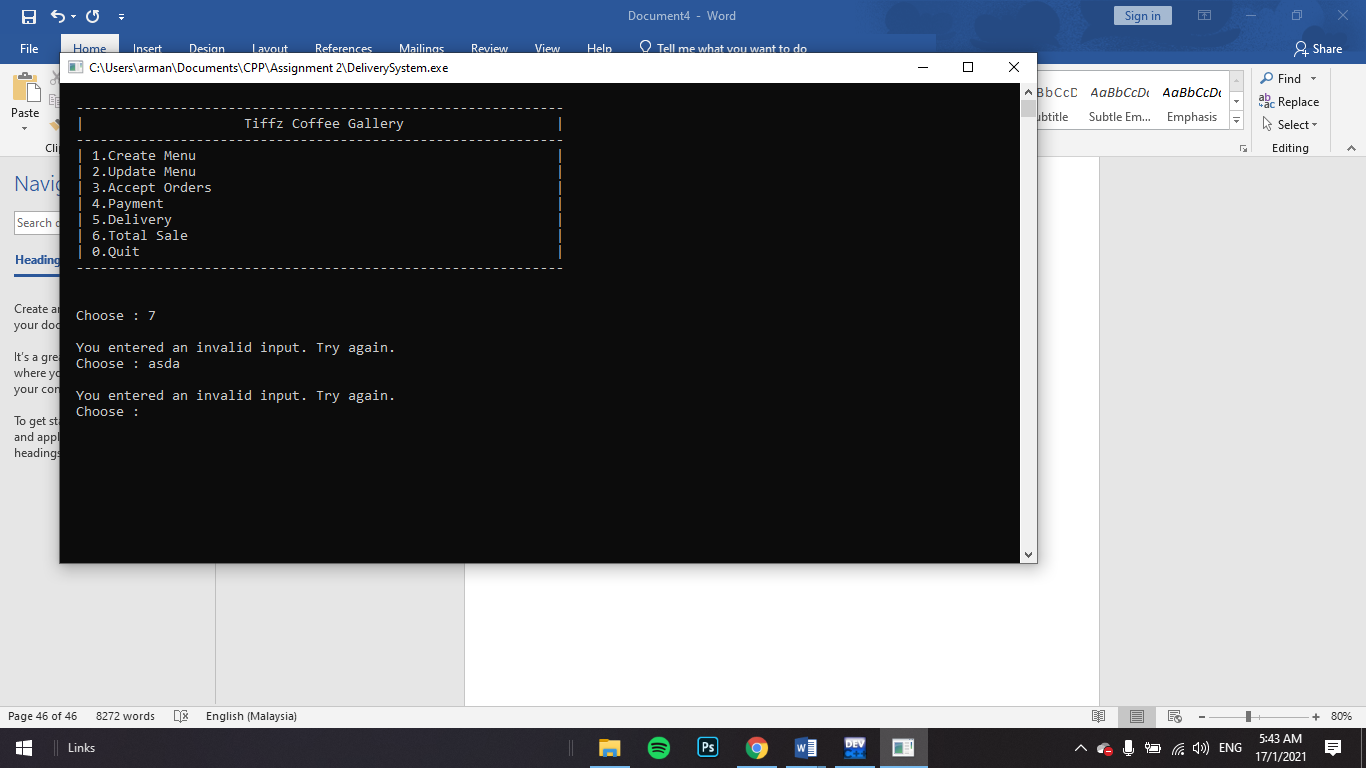


* 1. Incorrect login details

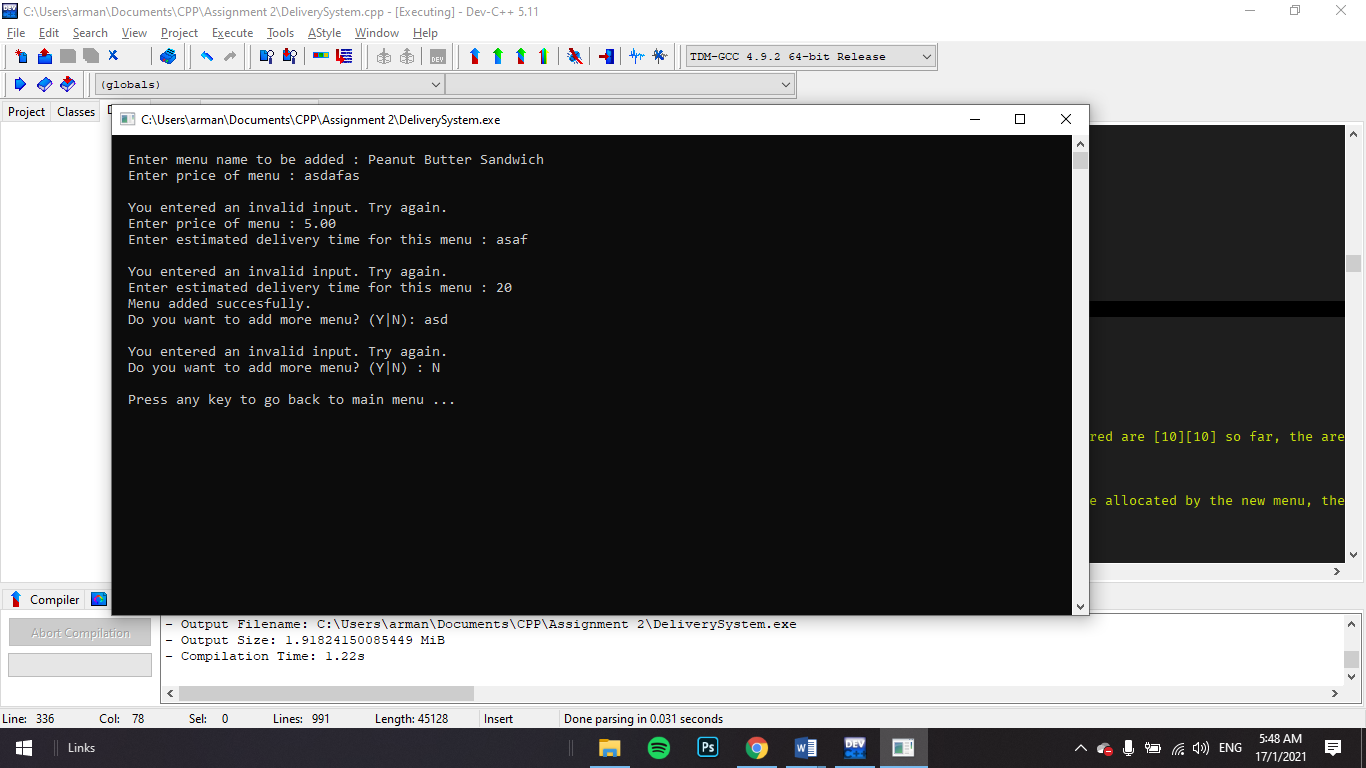


**Manager’s Input Output**

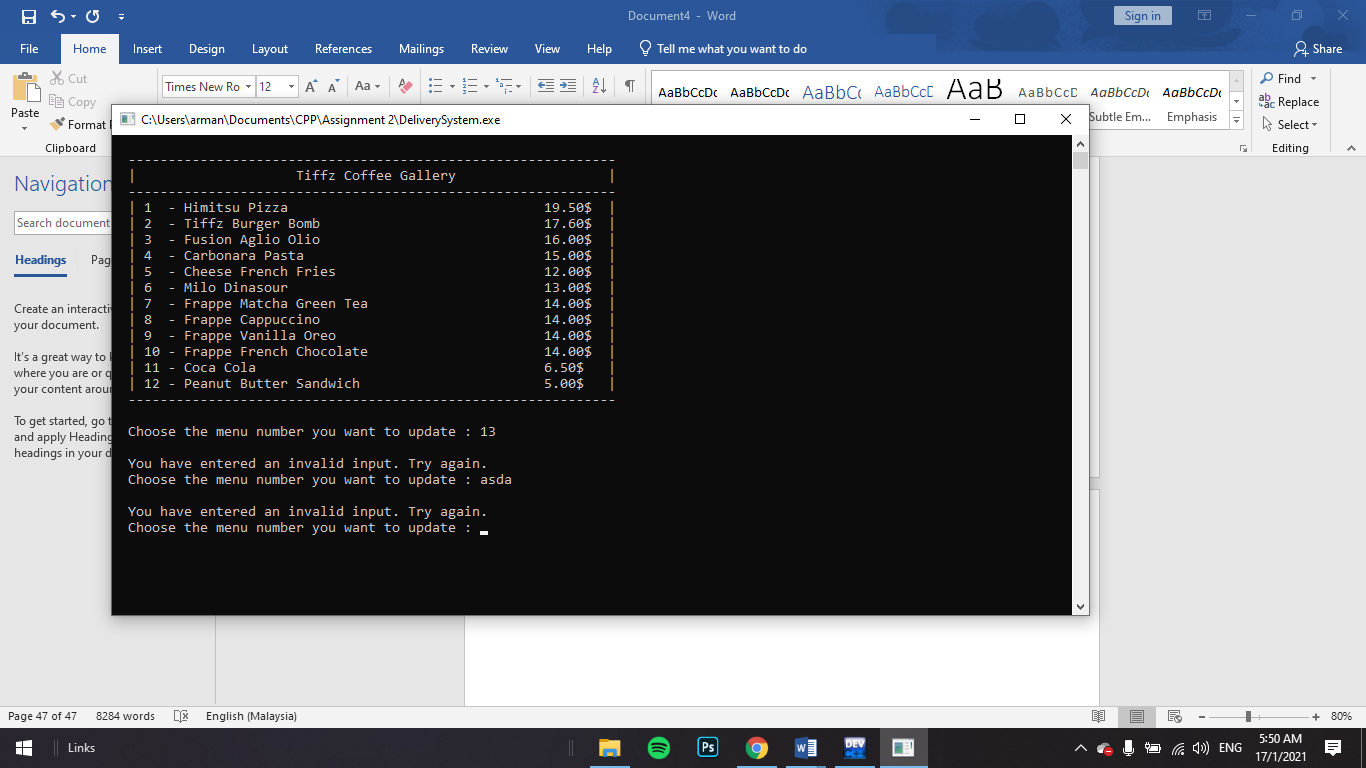
1. **Manager’s Menu**
2. Input Validation



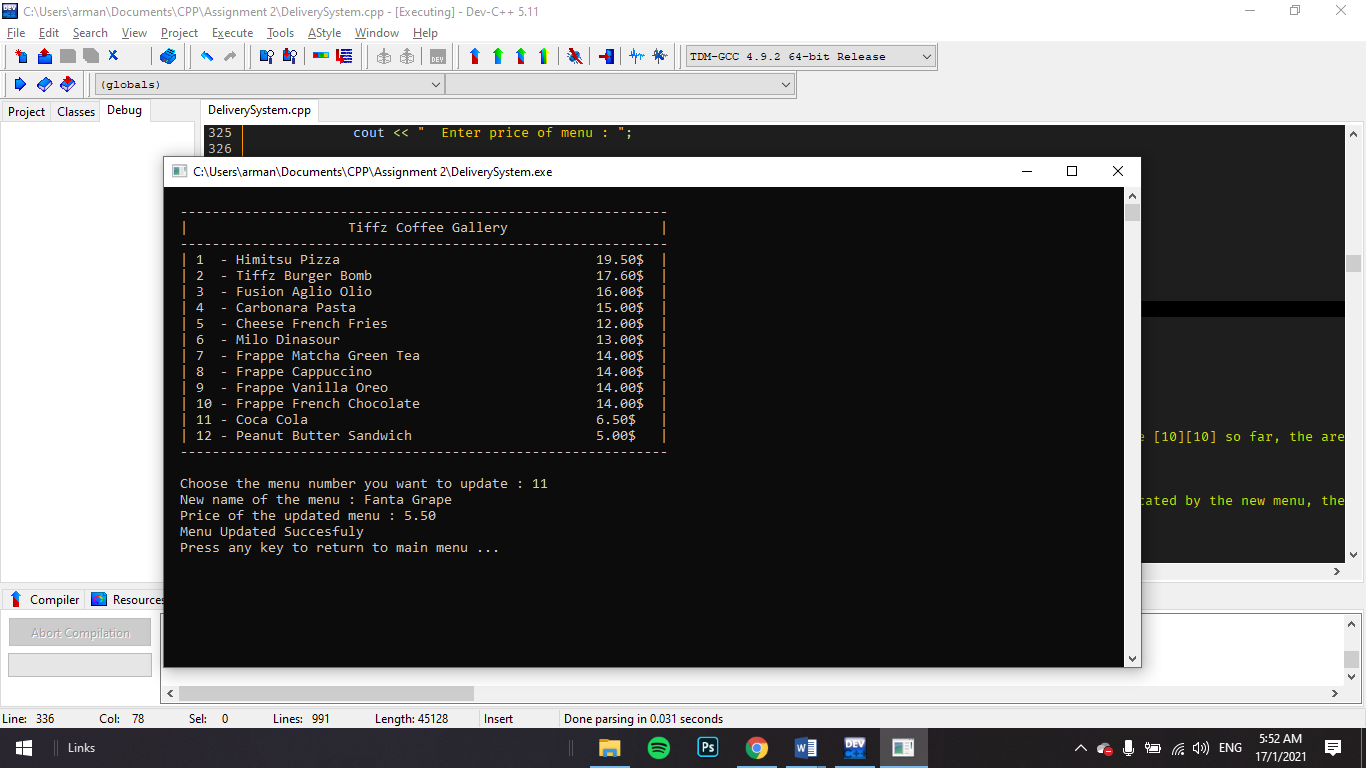
1. **User Choose Create Menu**
2. Input Validation

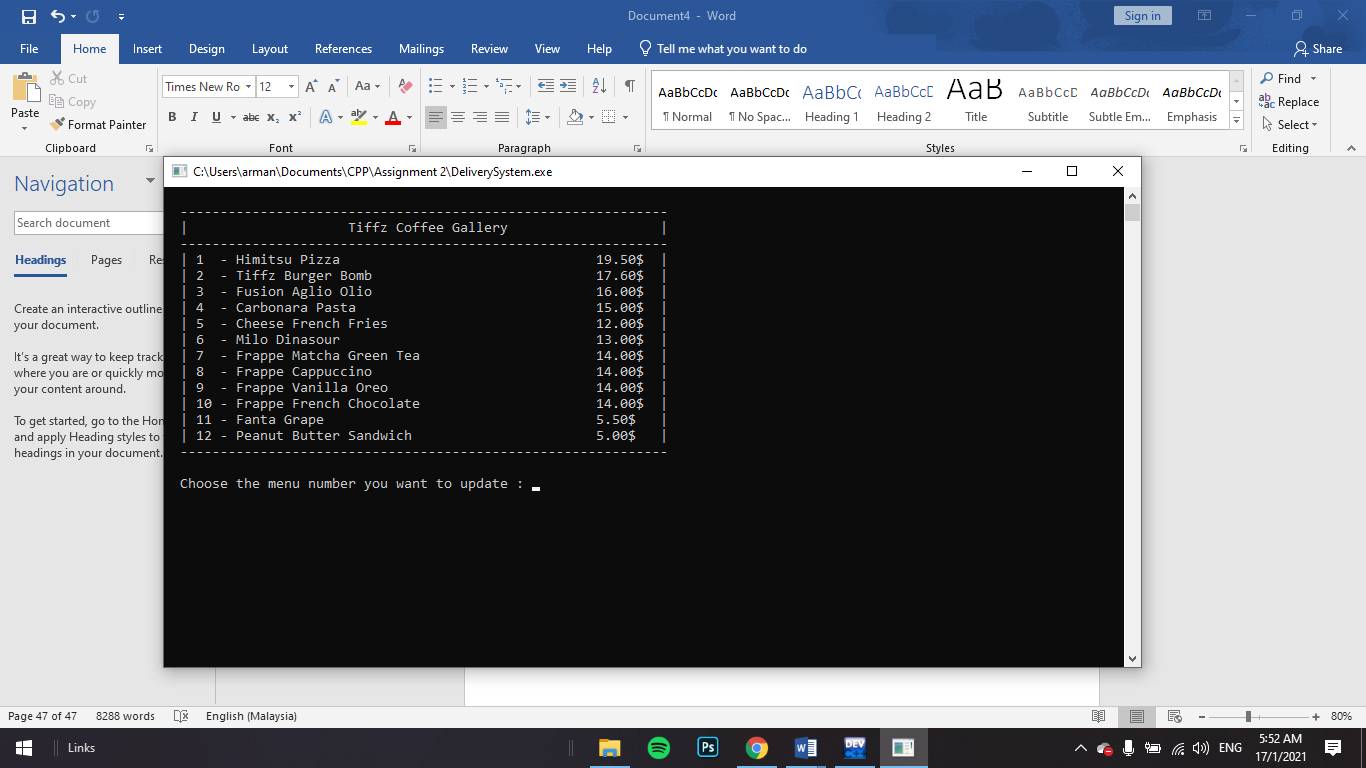


1. **User Choose Update Menu**
2. Input Validation

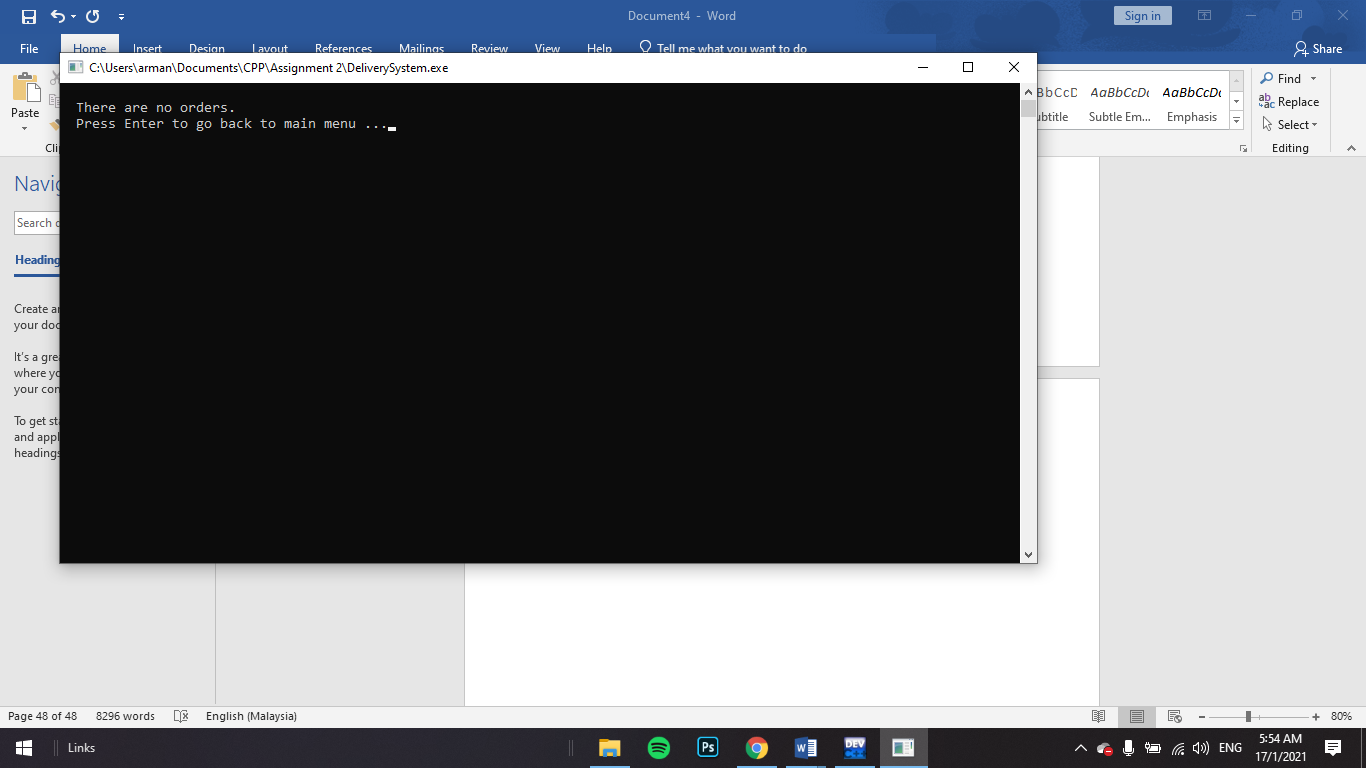


1. User Update Menu

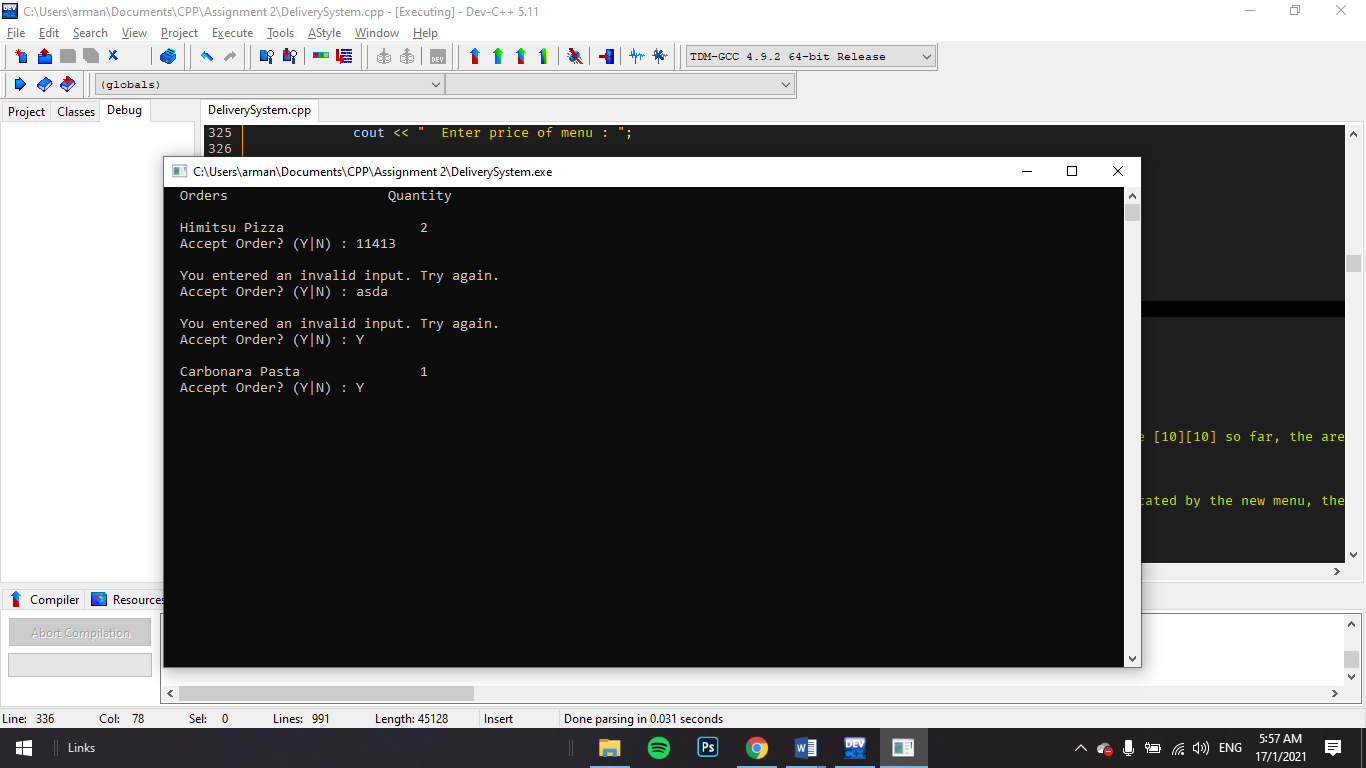




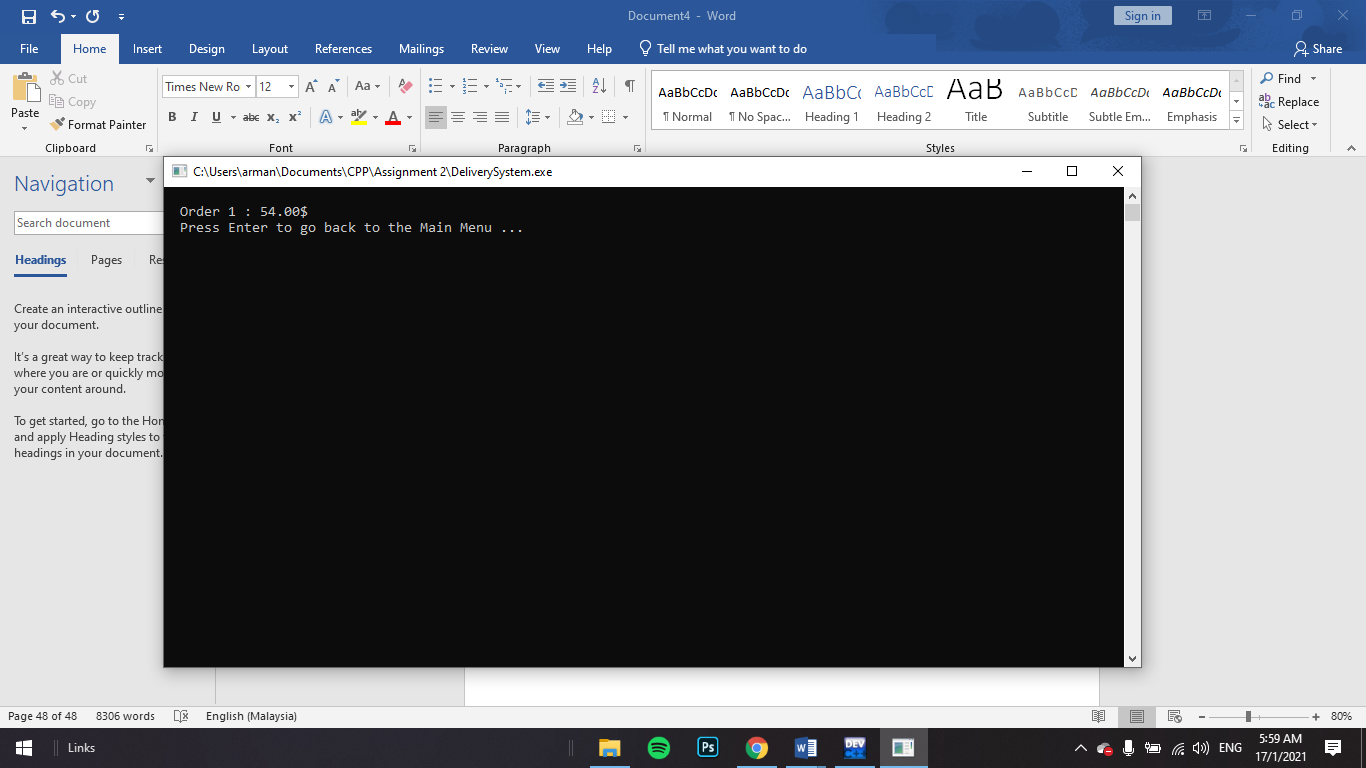
1. **User Choose Accept Order**
2. No orders from customers



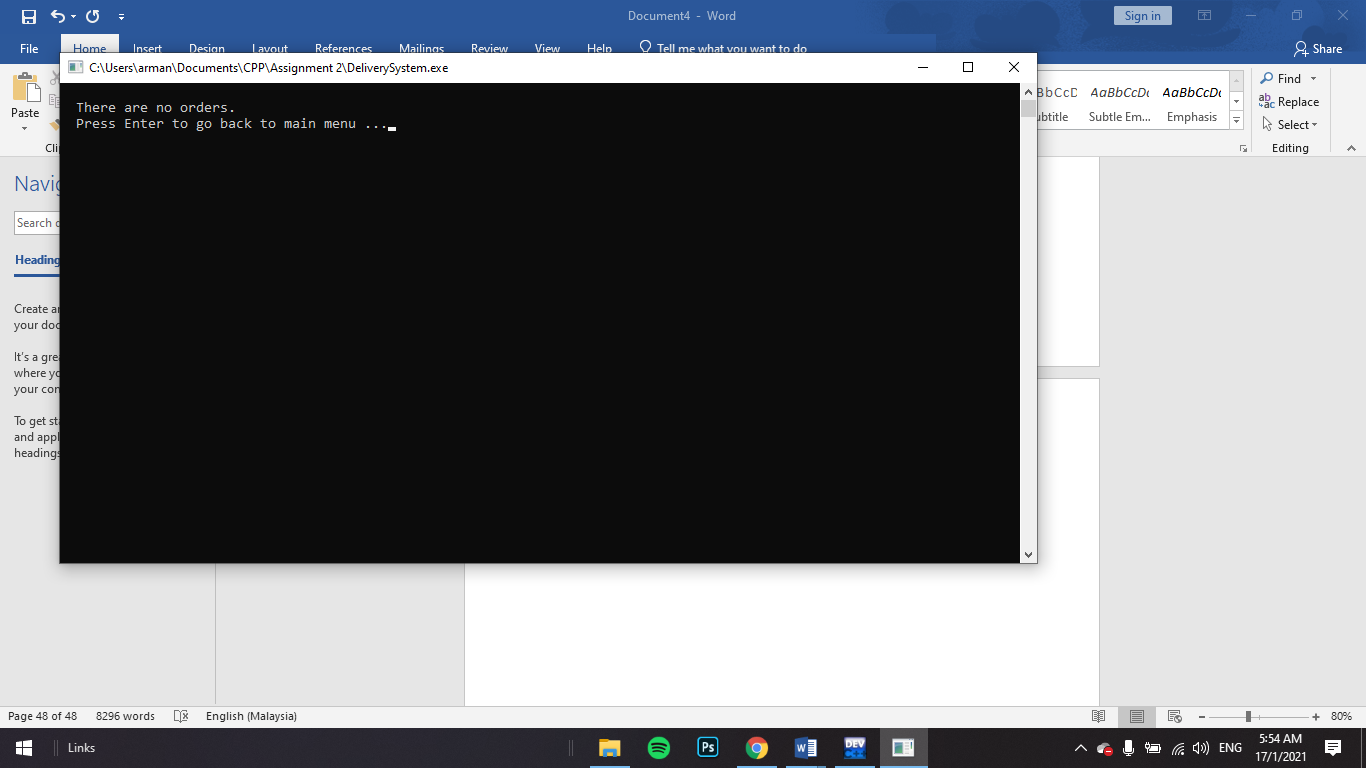
1. Input Validation



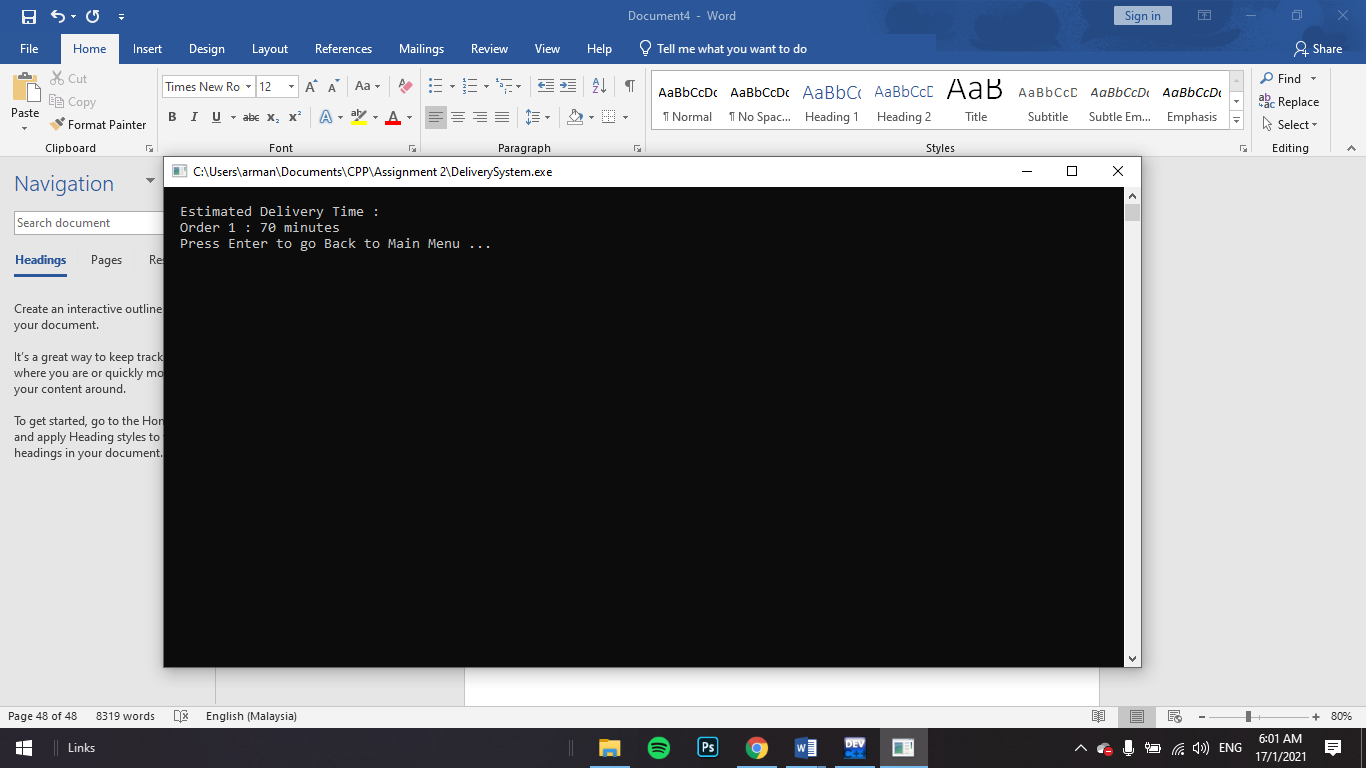
1. **User Choose Payment**
2. Payment made by customers



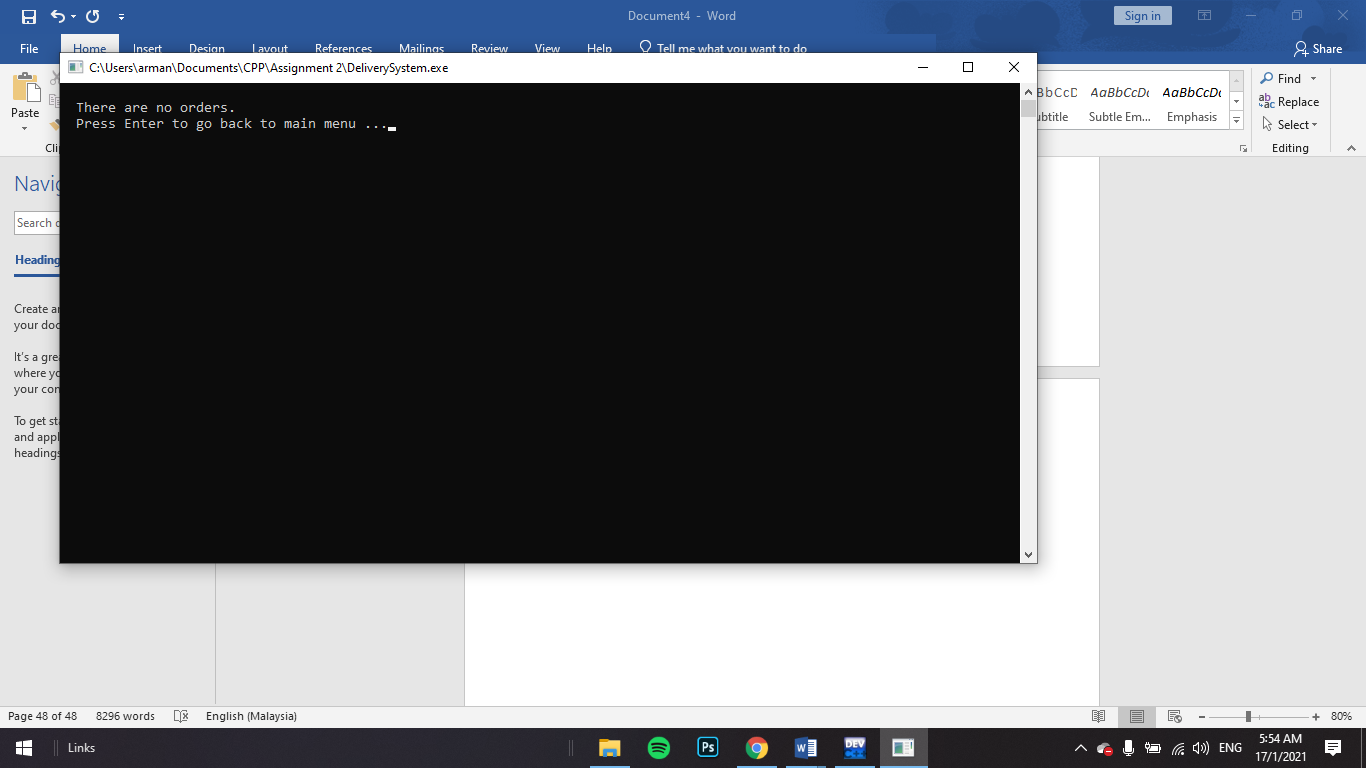
1. No orders by customers



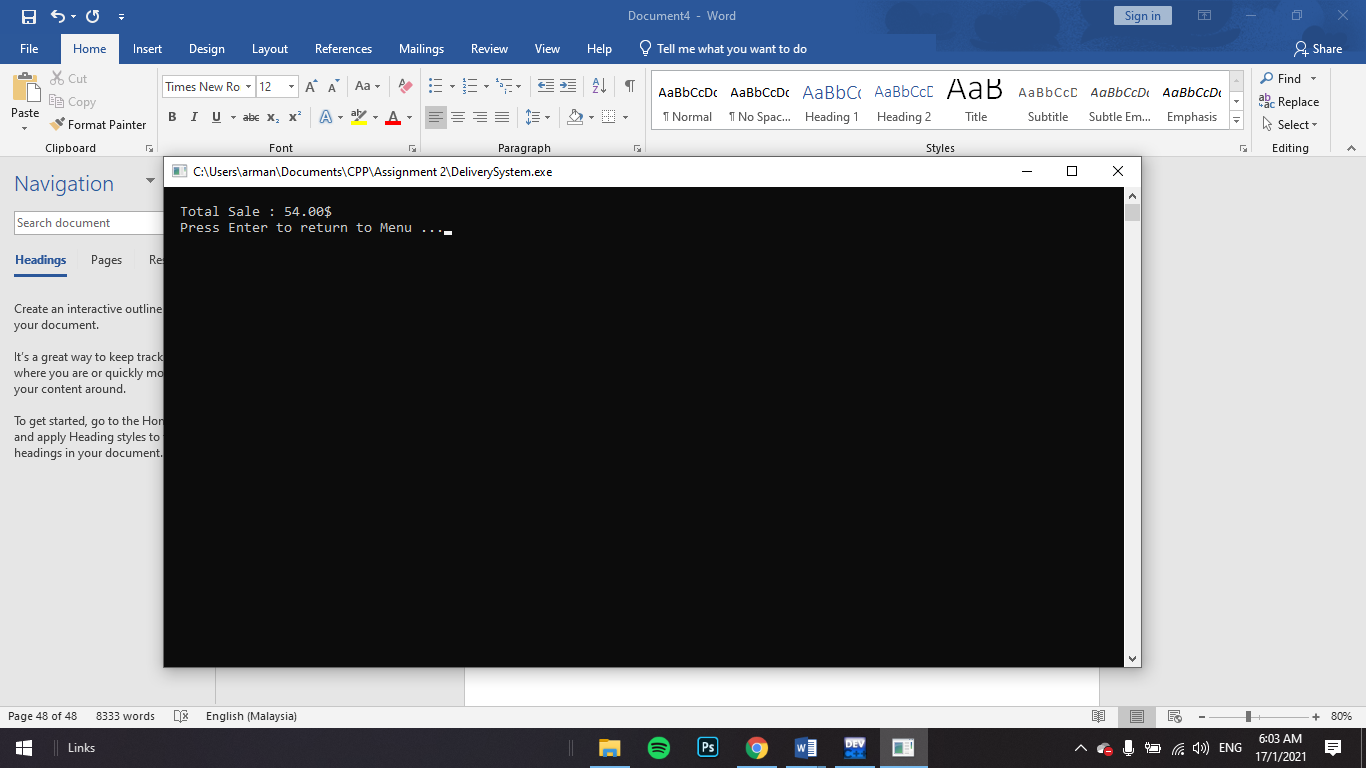
1. **User Choose Delivery**
2. Estimated delivery time for orders



1. No orders by customers

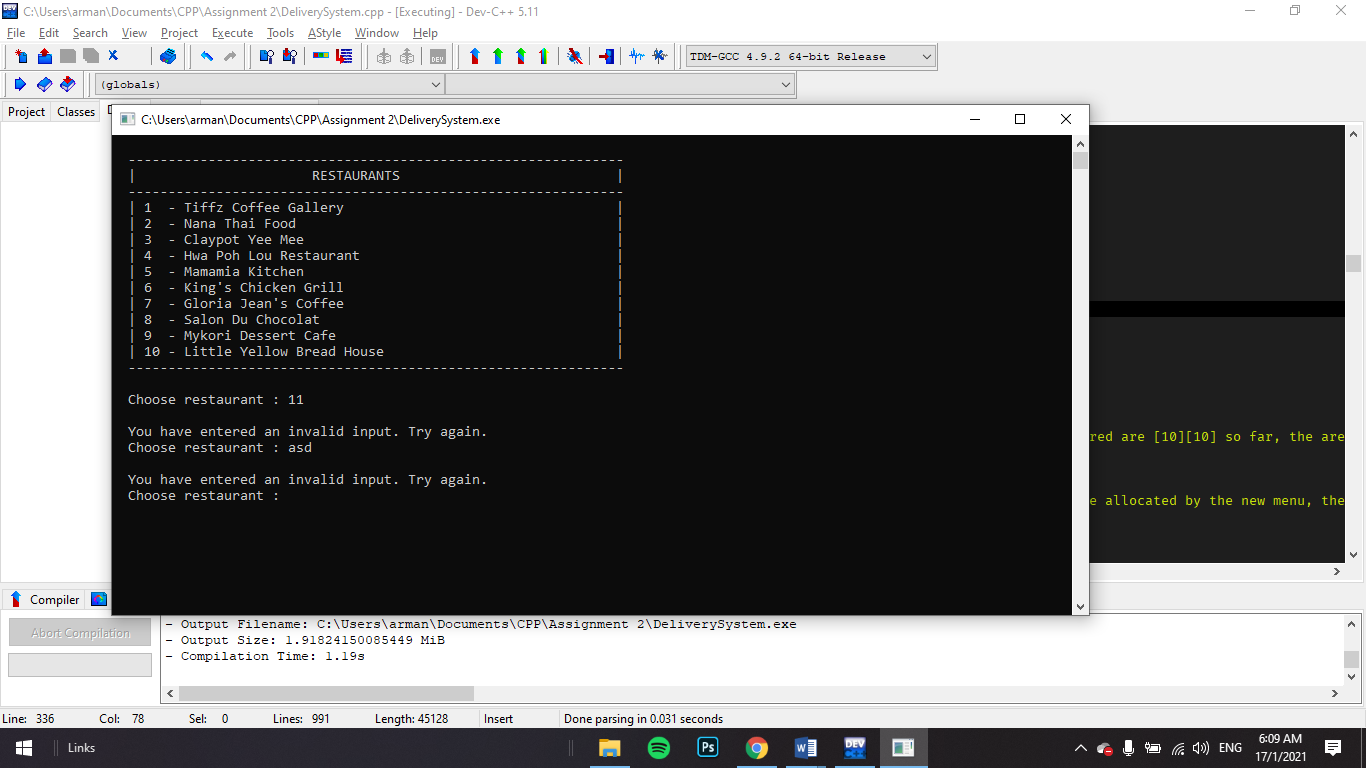


1. **User Choose Total Sale**
2. Total Sale of the restaurant

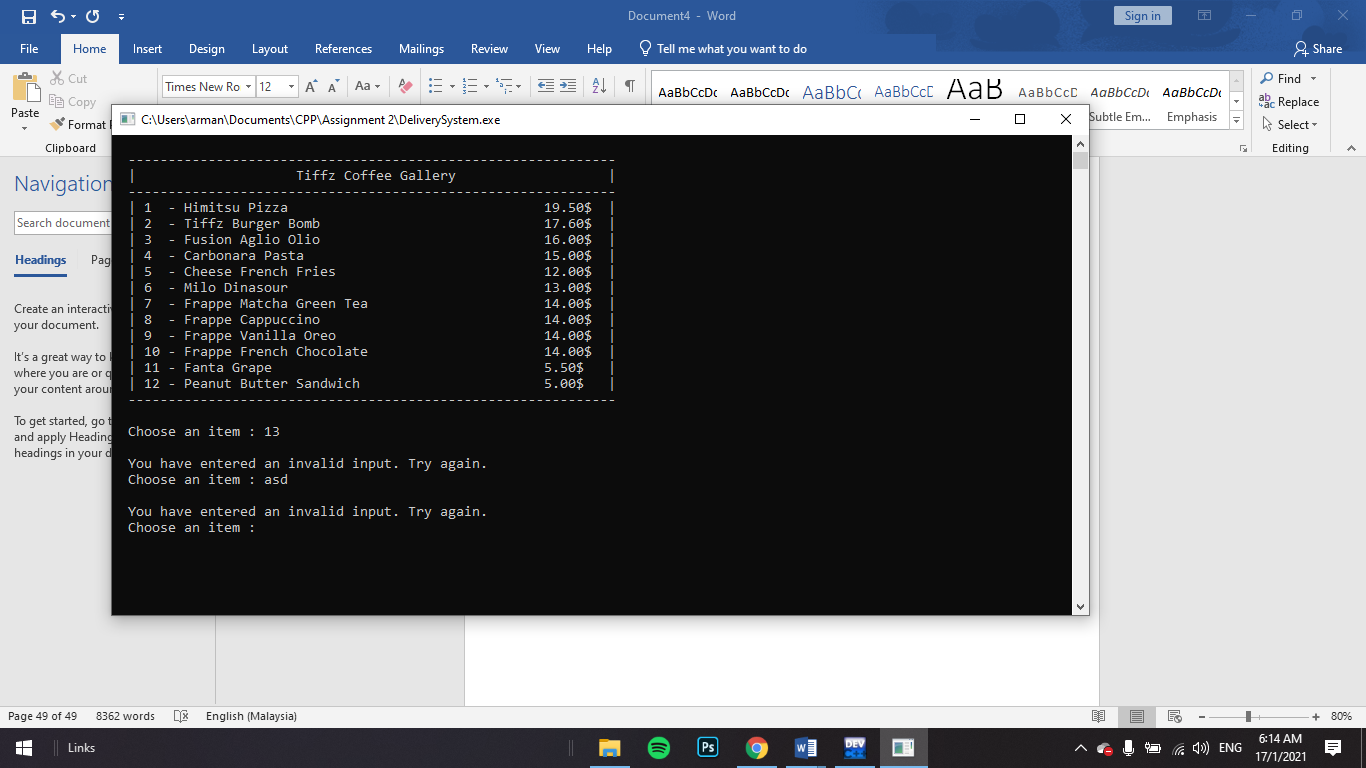


**Customer’s Input Output**

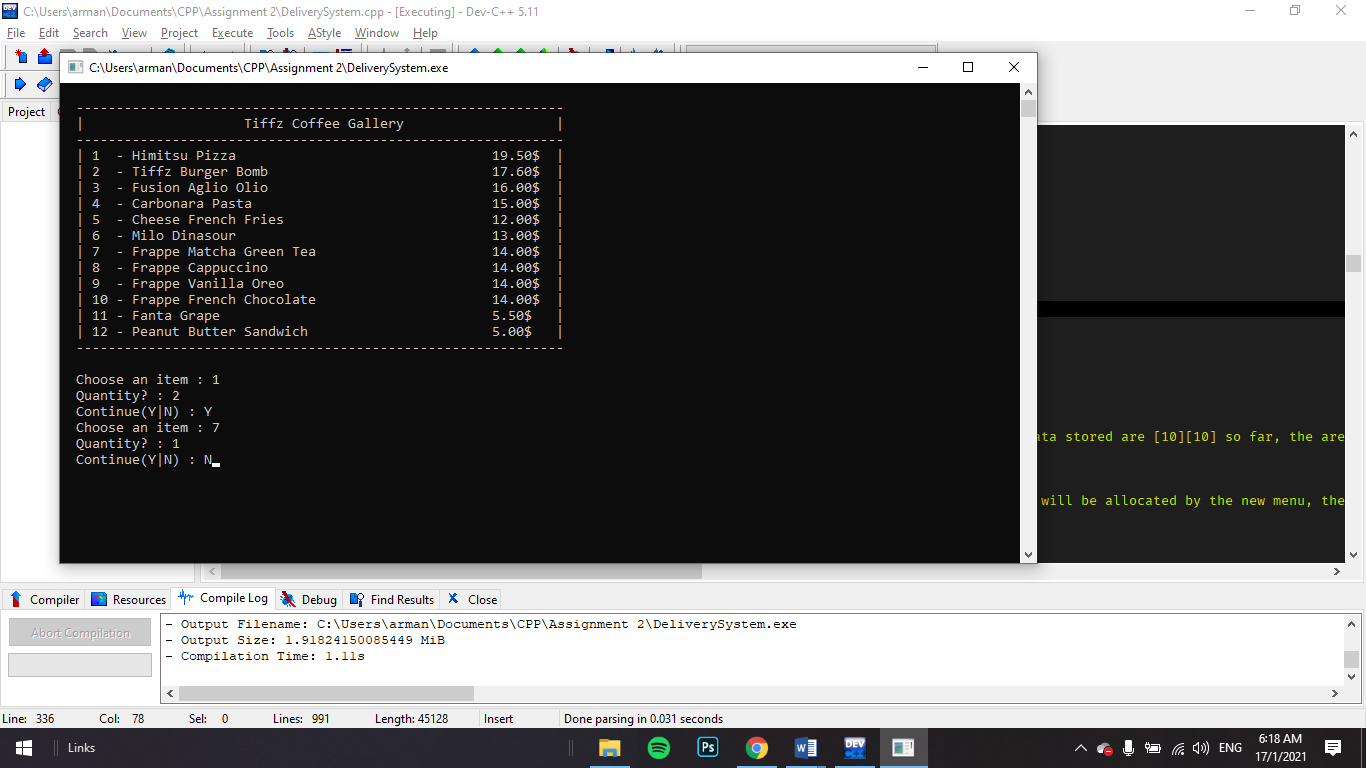
1. **User Choose Restaurant**
   1. Total Sale of the restaurant



1. **User Choose Menu**
   1. Input Validation



* 1. User orders food



* 1. Recipt is printed

