



SCHOOL OF COMPUTING

Subject Code	:	DCS1101
Subject Description	:	PROGRAMMING FUNDAMENTALS
Programme	:	DIPLOMA IN COMPUTER SCIENCE
Section	:	F1
Lecturer	:	Ms. Yugeswari
Coursework Description	:	Project(30%)
Type	:	Group (3 students)

Handout : Week 6 (19/2/22)

Submission: Week 7 (29/2/24)

Students' declaration:

We declare that:

- *We understand what is meant by plagiarism.*
- *This assignment is all our own work and we have acknowledged any use of the published or unpublished works of other people.*
- *We hold a copy of this assignment which we can produce if the original is lost or damaged*

Name	Matriculation number	Signature
1. Chai Yee Jin	J22038243	
2. Lee Zhao Shean	J22037224	
3. Quan Voon Joe	J22037904	
4. Cheah Zheng Ting	J22038494	

Assessment Criteria :		Total Marks	Marks
PART A			
1.	Report	20	
2.	Program	70	
3.	Presentation	10	
	TOTAL	100	
		Penalty	
Final Mark (30%)			
Lecturer's Comments :			



Vending Machine

A vending is an automated machine which is intended to provide the users with a diverse range of products: snacks, beverages, pizzas, cupcakes, newspapers, tickets, etc. It dispenses a product to the users based on the amount of money inserted and selection of the product.



The program should do the following task :

1. Show the customer a list of products sold by the Vending Machine.
(Minimum of 10 items in the Menu)
2. Let the customer make the selection.
3. Show the customer the item selected and total charges.
4. Accept money from customer.
5. Release the item and receipt.

Input – Selected item, quantity

Output – Customer receipt showing quantity of each item and the total payment.

Additional Requirements:

Execute the files by using any C++ software. Understand the flow of the program. Next, manipulate or add appropriate functions in the existing program to:

1. Let the customer choose more than one item at a time (iterate till the customer request to end choosing the item).
2. Prompt user to enter more money as long as the user has not entered enough money to buy the drinks.
3. Produce receipt to the customer which shows the drinks they choose, the cost, and balance of their money in `receipt.txt`.
4. Error Handler for invalid user input or item selection.

- ➔ Each group should choose a type or category of food e,g : Coffee, Juice, Chocolate, Stationaries, Cookies / Biscuits etc.
- ➔ Design solution for the task assigned – PAC, IPO, Algorithm and Flowchart
- ➔ You are required to add Decision , Looping , Functions, Array and Text file in one full program . You may include additional Structure / Functionality as per program Need/ Required.
- ➔ Explain your codes in the report.



Report - 20 Marks

- Format-Times New Roman, Size 12, 1.5 spacing
 - 1. Cover page
 - 2. Project description.(3 marks)
 - 3. Task for each group member (3 marks)
 - 4. Project explanation for each structure and functions .(5marks)
 - 5. The whole program (font Courier New)(3marks)
 - 6. Screenshots of the output(3marks)
 - 7. Summary – what you have learnt, experience gained, any challenges faced, etc.(3marks)

Presentation- 10 Marks (2marks for each component)

1. Content / Organisation- e.g Relevant, Clear
2. Teamwork- Participation from all members
3. Presentation Style- e.g DressCode, Voice, Pace
4. Time management
5. Use media- captivating presentation

Program- 70 Marks

Rubric

Design- PAC, IPO, Algorithm and Flowchart	15 marks
Display menu and item selection	5 marks
Show selected item and Total charges	10 marks
Accept payment	5 marks
Print receipt(text file) showing all the relevant details : Date, Item, Quantity, Total Charges, Balance.	10 marks
Program Structure- Decision Making, Looping, Function and Array	20 marks
Programming Styles- Comments, Indentation and etc.	5 marks
Total	70 marks



TABLE OF CONTENTS

1. INTRODUCTION-----	5
2. PROJECT DESCRIPTION -----	6
3. PROJECT DESIGN -----	7
4. PAC, IPO, ALGORITHM, FLOWCHART-----	11
5. CODE DESIGN -----	30
6. SCREENSHOT OF OUTPUT -----	36
7. SUMMARY -----	38



INTRODUCTION

On February 19, 2024, at 9.30 PM, our team held the first meeting online after our group member finish their all dinner. Our team consisted of four members, as listed above, with the collective goal of strategizing for the successful execution of our upcoming project. This report outlines the key discussions and decisions made during this meeting.

Meeting Agenda:

During the meeting, our team leader, Quan Voon Joe, introduced three critical questions that would guide our deliberations:

- 1. Which platform should we use for the project development?**
- 2. How should we approach the project's design?**
- 3. Who will be responsible for specific project segments?**

The meeting concluded with a well-structured plan for platform selection, project design, and assignment of responsibilities. The team is now equipped to move forward with a clear vision and defined roles, ensuring smooth and successful project execution.

On February 23, 2024, at 11.30 PM, Quan Voon Joe (our leader) initiated the first Coding 1.0 and then we continue start to do our Flowchart, Algorithm, IPO, and PAC. After we all discuss and keep continue redo and redo make sure nothing mistaken.

On February 28, 2024, at 11.10pm, as our last checking everything in processing don't have any polish in the coding, report, and presentation slide.



PROJECT DESCRIPTION

In this project, we will be constructing a program for a Coffee vending Machine.

Our Coffee Vending Machine consists of a structured programmed coding to function as an automated machine which allows the user to purchase one or more types of coffee conveniently.

We have discussed and distributed our task before we proceed with this project to prevent facing communication issues. These are the task for the team members:

Task	TEAM MEMBERS
Coding	Quan Voon Joe, Lee Zhao Shean
IPO chart, PAC chart, Algorithm, flowchart	Chai Yee Jin, Lee Zhao Shean
Project Report	Chai Yee Jin, Cheah Zheng Ting
Presentation	Cheah Zheng Ting, Quan Voon Joe

After distributing the class, we can start coding immediately. Once the coding is finished, we will check the code for any errors and try to improvise by modifying it so that the user can understand the vending machine easily. We should make sure that the code has no issue so that the user will be satisfied with the vending machine.



PROJECT DESIGN

```
//Define a struct for drinks
```

```
// Define a struct for drinks
struct Drink {
    string code;
    string name;
    double price;
    int quantity;
};
```

Figure 2.1

Figure 2.1 , **struct Drink {** is a user defined data type called a struct to let you combine variables of different type. ‘**String code;**’ declares a member variable name ‘**code**’ inside the ‘**Drink**’ struct. It is a type of ‘**string**’. ‘**string name**’ declare a variable named ‘**name**’ of type ‘**string**’. ‘**double price**’ declare a variable named ‘**price**’ of type ‘**double**’. ‘**int quantity**’ declares a variable named ‘**quantity**’ of type ‘**int**’.

```
//Display coffee menu
```

```
// Function: Display coffee menu
void display_coffee_menu(const Drink coffee[], int size) {
    cout << "Drinks Menu:" << endl;
    for (int i = 0; i < size; ++i) {
        cout << coffee[i].code << " = " << coffee[i].name << " (RM" << fixed << setprecision(2) << coffee[i].price << ")" << endl;
    }
}
```

Figure 2.2

Figure 2.2 shows the code about display coffee menu. “coffee[i].code” meant display the code of the coffee, “coffee[i].name” meant display the name of coffee and “coffee[i].price” meant display the price of each coffee.

```
//Get more items
```

```
// Function: Get more items
bool add_on_items() {
    char choice;

    while (true) {
        cout << "Anything else you would like to add on? (Y/N): ";
        cin >> choice;

        if (choice == 'Y' || choice == 'y') {
            return true;
        } else if (choice == 'N' || choice == 'n') {
            return false;
        } else {
            cout << "Invalid input, please enter (Y/N) to add on items or not..." << endl;
            cin.clear(); //Clear the error state flags
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer
        }
    }
}
```

Figure 2.3

Figure 2.3 show the code about the user selection. If user input “Y” or “y”, the program will loop back to display coffee menu and let user continue select other coffee. If user input “N” or “n”, the looping will stop and run it to calculate payment function.



```
//Get quantity
// Function: Get quantity
int get_quantity() {
    int quantity;

    do {
        cout << "Enter quantity: ";
        cin >> quantity;

        if (quantity <= 0) {
            cout << "Invalid quantity, please a valid positive quantity..." << endl;
            cin.clear(); //Clear the error state flags
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear the input buffer
        }
    } while (quantity <= 0);
    return quantity;
}
```

Figure 2.4

Figure 2.4 shows the code about user input quantity. It similar like Figure 2.3 program but the program change it from while loop function to do while function.

```
//Print receipt
// Function: Print Receipt
void print_receipt(const Drink coffee[], int size, double totalAmount, double payment, double change) {
    ofstream receiptFile("receipt.txt"); // Open receipt from txt file

    // Set time-zone to Malaysia
    putenv("TZ=Asia/Kuala_Lumpur");

    // Get current date and time
    time_t currentTime = time(nullptr);
    tm* localTime = localtime(&currentTime);
    char dateTime[100];
    strftime(dateTime, sizeof(dateTime), "%c", localTime);

    receiptFile << "Receipt: " << dateTime << endl; // Include date and time
    receiptFile << "-----" << endl;
    receiptFile << "Drinks Ordered:" << endl;

    for (int i = 0; i < size; ++i) {
        if (coffee[i].quantity > 0) {
            receiptFile << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed << setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl;
        }
    }

    receiptFile << "\nTotal Amount: RM" << fixed << setprecision(2) << totalAmount << endl;
    receiptFile << "Payment: RM" << fixed << setprecision(2) << payment << endl;
    receiptFile << "Change: RM" << fixed << setprecision(2) << change << endl;

    cout << "Thank you for your purchase! Here is your change: RM" << fixed << setprecision(2) << change << endl;
    cout << "Do enjoy your coffee, have a nice day!" << endl;

    receiptFile.close(); // Close the file
}
```

Figure 2.5

Figure 2.5 shows the code about print receipt. The code will open a “receipt.txt” file and it will print the current local time and date first, then it will start display the coffee that user choose, display the number of quantity that user input and display the price of each coffee. After display the detail of user input, it will start display the totalAmount, payment and change that the program already calculate.



```
//Calculate payment
// Function: calculate payment
void handle_payment(double totalAmount, const Drink coffee[], int size) {
    double payment, imbalance = totalAmount;

    while (imbalance > 0) {

        for (int i = 0; i < size; ++i) {
            if (coffee[i].quantity > 0) {
                cout << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed << setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl;
            }
        }

        cout << "Enter payment amount RM: ";
        cin >> payment;

        if (cin.fail() || payment <= 0) {
            cout << "Invalid payment, please enter a valid positive amount." << endl;
            cin.clear(); // Clear the error state flags
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer
        } else {
            imbalance -= payment;
            if (imbalance > 0) {
                cout << "Insufficient payment! You still have to pay RM" << fixed << setprecision(2) << imbalance << " in order to purchase your coffee." << endl;
            }
        }
    }

    // calculate change
    double change = -imbalance;

    // Generate receipt from 'Function: print receipt'
    print_receipt(coffee, size, totalAmount, totalAmount - imbalance, change);
}
```

Figure 2.6

Figure 2.6 shows the code about calculate payment. The program will start calculate the final totalAmount that user have been choose. After get the final totalAmount, it will start calculate the change, if user input value is less than final totalAmount, it will let user input the rest of the amount, if user input more money than final totalAmount, it will start calculate the change and find a change to user.

```
//Find a drink by code
```

```
// Function: Find a drink by code
void input_drink_code(Drink coffee[], int size) {
    double totalAmount = 0.0;

    do {
        string userCode;
        bool validCode = false;

        do {
            cout << "Enter your coffee by following the menu code:" << endl;
            cin >> userCode;

            for (int i = 0; i < size; ++i) {
                if (coffee[i].code == userCode) {
                    cout << "You selected " << coffee[i].name << " (RM" << fixed << setprecision(2) << coffee[i].price << ")" << endl;
                    validCode = true;
                    // Ask quantity using 'Function: Get quantity'
                    int quantity = get_quantity();
                    // Calculate total amount
                    totalAmount += coffee[i].price * quantity;
                    // Update quantity in the coffee array
                    coffee[i].quantity += quantity;
                }
            }
        } while (!validCode);
        // Ask add more items using 'Function: Get more items'
    } while (add_on_items());

    cout << "Total amount is RM: " << fixed << setprecision(2) << totalAmount << endl;

    handle_payment(totalAmount, coffee, size);
}
```

Figure 2.7

Figure 2.7 shows the code about find a drink by code. First, it will output the coffee menu first, next, it will call the user input the code, the code input by user will record it and list the coffee name and price, at the same time, it also will call user input the quantity. Next step is system will asking user did user need select another coffee or not, if yes it will looping back again if no it will continue move to next step.



```
//Int main
int main() {
    Drink coffee[] = {
        {"H1", "Hot-Light Coffee", 5.0},
        {"H2", "Hot-Medium Coffee", 5.0},
        {"H3", "Hot-Dark Coffee", 5.0},
        {"H4", "Hot-Latte", 8.0},
        {"H5", "Hot-Mocha", 10.0},
    };
    {"C1", "Ice-Light Coffee", 7.0},
    {"C2", "Ice-Medium Coffee", 7.0},
    {"C3", "Ice-Dark Coffee", 7.0},
    {"C4", "Ice-Latte", 10.0},
    {"C5", "Ice-Mocha", 12.0}
};

    display_coffee_menu(coffee, sizeof(coffee) / sizeof(coffee[0]));
    input_drink_code(coffee, sizeof(coffee) / sizeof(coffee[0]));

    return 0;
}
```

Figure 2.8

Figure 2.8 shows the code about types of the coffee. The program will output the list of the coffee, code and price for user select.



PAC

//Display coffee menu

Given Data	Required Result
code, name, price, quantity	Display coffee menu
Process Required	Solution Alternative
<pre>Output drink menu for (int i = 0; i < size; ++i) { cout << coffee[i].code << " = " << coffee[i].name << " (RM" << fixed << setprecision(2) << coffee[i].price << ")" << endl;</pre>	<pre>Define code as string Define name as string Define price as double Define quantity as integer</pre>

//Get more items

Given Data	Required Result
choice	Display true or false
Process Required	Solution Alternative
<pre>Output choice if (choice == 'Y' choice == 'y') { return true; } else if (choice == 'N' choice == 'n') { return false; } else { cout << "Invalid input, please enter (Y/N) to add on items or not..." << endl; cin.clear(); //Clear the error state flags cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer }</pre>	<pre>Define choice as char</pre>

//Get quantity

Given Data	Required Result
quantity	Display the value that user input
Process Required	Solution Alternative
<pre>do { cout << "Enter quantity: "; cin >> quantity; if (quantity <= 0) { cout << "Invalid quantity, please a valid positive quantity..." << endl; cin.clear(); //Clear the error state flags cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer } }</pre>	<pre>Define quantity as integer</pre>



//Print receipt

<u>Given Data</u>	<u>Required Result</u>
size, totalAmount, payment, change, currentTime	Display receipt and localTime
<pre>// Set time-zone to Malaysia putenv("TZ=Asia/Kuala_Lumpur"); // Get current date and time time_t currentTime = time(nullptr); tm* localTime = localtime(&currentTime); char dateTime[100]; strftime(dateTime, sizeof(dateTime), "%c", localTime); receiptFile << "Receipt: " << dateTime << endl; // Include date and time receiptFile << "----- -----" << endl; receiptFile << "Drinks Ordered:" << endl; for (int i = 0; i < size; ++i) { if (coffee[i].quantity > 0) { receiptFile << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed << setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl; } } receiptFile << "\nTotal Amount: RM" << fixed << setprecision(2) << totalAmount << endl; receiptFile << "Payment: RM" << fixed << setprecision(2) << payment << endl; receiptFile << "Change: RM" << fixed << setprecision(2) << change << endl; cout << "Thank you for your purchase! Here is your change: RM" << fixed << setprecision(2) << change << endl;</pre>	<pre>Define size as integer Define totalAmount as double Define payment as double Define change as double Define currentTime as double</pre>



//Calculate payment

Given Data	Required Result
totalAmount, size, payment, imbalance, quantity	Display change
Process Required <pre>while (imbalance > 0) { for (int i = 0; i < size; ++i) { if (coffee[i].quantity > 0) { cout << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed << setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl; } } cout << "Enter payment amount RM: "; cin >> payment; if (cin.fail() payment <= 0) { cout << "Invalid payment, please enter a valid positive amount." << endl; cin.clear(); // Clear the error state flags cin.ignore(numeric_limits<streamsize>::max(), \n); // Clear the input buffer } else { imbalance -= payment; if (imbalance > 0) { cout << "Insufficient payment! You still have to pay RM" << fixed << setprecision(2) << imbalance << " in order to purchase your coffee." << endl; } } }</pre>	Solution Alternative Define totalAmount as double Define size as integer Define payment as double Define imbalance as double Define quantity as integer



//Find a drink by code

Given Data	Required Result
size, totalAmount, quantity	Display the drink that user select
Process Required	Solution Alternative
<pre>do { cout << "Enter your coffee by following the menu code:" << endl; cin >> userCode; for (int i = 0; i < size; ++i) { if (coffee[i].code == userCode) { cout << "You selected " << coffee[i].name << " (RM" << fixed << setprecision(2) << coffee[i].price << ")" << endl; validCode = true; // Ask quantity using 'Function: Get quantity' int quantity = get_quantity(); // Calculate total amount totalAmount += coffee[i].price * quantity; // Update quantity in the coffee array coffee[i].quantity += quantity; } }</pre>	<p>Define size as integer Define totalAmount as double Define quantity as integer</p>



//Int Main

Given Data	Required Result
main	Display coffee menu
Process Required	Solution Alternative
<pre>Drink coffee[] = { {"H1", "Hot-Light Coffee", 5.0}, {"H2", "Hot-Medium Coffee", 5.0}, {"H3", "Hot-Dark Coffee", 5.0}, {"H4", "Hot-Latte", 8.0}, {"H5", "Hot-Mocha", 10.0}, {"C1", "Ice-Light Coffee", 7.0}, {"C2", "Ice-Medium Coffee", 7.0}, {"C3", "Ice-Dark Coffee", 7.0}, {"C4", "Ice-Latte", 10.0}, {"C5", "Ice-Mocha", 12.0} }; display_coffee_menu(coffee, sizeof(coffee) / sizeof(coffee[0])); input_drink_code(coffee, sizeof(coffee) / sizeof(coffee[0]));</pre>	Define main as integer



IPO

//Display coffee menu

Input	Processing	Output
code, name, price, quantity	<ol style="list-style-type: none"> 1. Read code 2. Read name 3. Read Price 4. Read quantity 5. Display menu 6. for (int i = 0; i < size; ++i) { cout << coffee[i].code << " = " << coffee[i].name << " (RM" << fixed << setprecision(2) << coffee[i].price << ")" << endl; 7. End 	coffee menu

//Get more items

Input	Processing	Output
choice	<ol style="list-style-type: none"> 1. Read choice 2. Process user input <pre>if (choice == 'Y' choice == 'y') { return true; } else if (choice == 'N' choice == 'n') { return false; } else { cout << "Invalid input, please enter (Y/N) to add on items or not..." << endl; cin.clear(); //Clear the error state flags cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer }</pre> 3. Print selection 4. End 	True, False

//Get quantity

Input	Processing	Output
quantity	<ol style="list-style-type: none"> 1. Read quantity 2. Process user input <pre>do { cout << "Enter quantity: "; cin >> quantity; if (quantity <= 0) { cout << "Invalid quantity, please a valid positive quantity..." << endl; cin.clear(); //Clear the error state flags cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer } }</pre> 	value



	3. Print value 4. End	
--	--------------------------	--

//Print receipt

Input	Processing	Output
size, totalAmount, payment, change, currentTime	<pre>1. Read size 2. Read totalAmount 3. Read payment 4. Read change 5. Read currentTime 6. Display receipt // Set time-zone to Malaysia putenv("TZ=Asia/Kuala_Lumpur"); // Get current date and time time_t currentTime = time(nullptr); tm* localTime = localtime(&currentTime); char dateTime[100]; strftime(dateTime, sizeof(dateTime), "%c", localTime); receiptFile << "Receipt: " << dateTime << endl; // Include date and time receiptFile << "----- ---" << endl; receiptFile << "Drinks Ordered:" << endl; for (int i = 0; i < size; ++i) { if (coffee[i].quantity > 0) { receiptFile << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed << setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl; } } receiptFile << "\nTotal Amount: RM" << fixed << setprecision(2) << totalAmount << endl; receiptFile << "Payment: RM" << fixed << setprecision(2) << payment << endl; receiptFile << "Change: RM" << fixed << setprecision(2) << change << endl; cout << "Thank you for your purchase! Here is your change: RM" << fixed << setprecision(2)</pre>	Receipt, localTime



	<< change << endl; 7. End	
--	------------------------------	--

//Calculate payment

Input	Processing	Output
totalAmount, size, payment, imbalance, quantity	<pre>1. Read totalAmount 2. Read size 3. Read payment 4. Read imbalance 5. Read quantity 6. Calculate change for (int i = 0; i < size; ++i) { if (coffee[i].quantity > 0) { cout << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed << setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl; } } cout << "Enter payment amount RM: "; cin >> payment; if (cin.fail() payment <= 0) { cout << "Invalid payment, please enter a valid positive amount." << endl; cin.clear(); // Clear the error state flags cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer } else { imbalance -= payment; if (imbalance > 0) { cout << "Insufficient payment! You still have to pay RM" << fixed << setprecision(2) << imbalance << " in order to purchase your coffee." << endl; } } 7. End</pre>	change



//Find a drink by code

Input	Processing	Output
size, totalAmount, quantity	<pre> 1. Read size 2. Read totalAmount 3. Read quantity 4. Display selection do { cout << "Enter your coffee by following the menu code:" << endl; cin >> userCode; for (int i = 0; i < size; ++i) { if (coffee[i].code == userCode) { cout << "You selected " << coffee[i].name << " (RM" << fixed << setprecision(2) << coffee[i].price << ")" << endl; validCode = true; // Ask quantity using 'Function: Get quantity' int quantity = get_quantity(); // Calculate total amount totalAmount += coffee[i].price * quantity; // Update quantity in the coffee array coffee[i].quantity += quantity; } } } 5. End </pre>	selection

//Int Main

Input	Processing	Output
main	<pre> 1. Read main 2. Display coffee menu Drink coffee[] = { {"H1", "Hot-Light Coffee", 5.0}, {"H2", "Hot-Medium Coffee", 5.0}, {"H3", "Hot-Dark Coffee", 5.0}, {"H4", "Hot-Latte", 8.0}, {"H5", "Hot-Mocha", 10.0}, {"C1", "Ice-Light Coffee", 7.0}, {"C2", "Ice-Medium Coffee", 7.0}, {"C3", "Ice-Dark Coffee", 7.0}, {"C4", "Ice-Latte", 10.0}, {"C5", "Ice-Mocha", 12.0} }; </pre>	coffee menu



SCHOOL OF COMPUTING

	<pre> display_coffee_menu(coffee, sizeof(coffee) / sizeof(coffee[0])); input_drink_code(coffee, sizeof(coffee) / sizeof(coffee[0])); 3. End</pre>	
--	--	--



ALGORITHM

1. Start
2. Define a struct for Drinks
 - string code
 - string name
 - double price
 - int quantity
3. Int main function
4. Initialize an array of Drinks objects with menu details
5. Display the coffee menu in display_coffee_menu function
6. Input drink code
7. Input payment in input_drink_code function
8. Called display_coffee_menu function
9. Display heading “Drink menu: ”
10. For int I = 0, I < size statement is true, ++I go to step 11
11. Display coffee[I].code, coffee[I].name and coffee[I].price
12. End display_coffee_menu function
13. Called add_on_items function
14. Set char choice
15. While statement is true, go to step 16
16. Read choice
17. If choice == ‘Y’ || choice == ‘y’ statement is true, go to step 20 otherwise go to step 18
18. Else if choice == ‘N’ || choice == ‘n’ statement true, go to step 21 otherwise go to step 19
19. Print “Invalid input, please enter(Y/N) to add on items or not.”, go to step 16
20. Return true
21. Return false
22. End add_on_items function
23. Called get_quantity function
24. Set int quantity
25. Do, read quantity
26. If quantity <= 0 statement is true, go to step 27 otherwise go to step 28
27. Print “Invalid quantity, please enter a valid positive quantity”, go to step 25
28. While quantity <= 0 statement is true, go to step 27, otherwise go to step 29



29. Return quantity
30. End get_quantity function
31. Called handle_payment function
32. Set double totalAmount, const Drink coffee[], int size, double payment, imbalance = totalAmount
33. For int I=0; I<size statement is true, go to step 34
34. If coffee[I].quantity>0 statement is true, print coffee[I].quantity, coffee[I].name
35. while imbalance > 0 statement is true, go to step 36
36. Read payment
37. If payment <= 0 statement is true, go to step 36 otherwise go to step 39
38. Print “Invalid payment, please enter a valid positive amount”, go to step 35
39. Imbalance -= payment
40. If imbalance >= 0 statement is true, go to step 41 otherwise go to step 35
41. Print imbalance
42. Change -= imbalance
43. print_receipt(coffee, size, totalAmount, totalAmount - imbalance, change)
44. return
45. End handle_payment function
46. Called input_drink_code function
47. Set Drink coffee[], int size, double totalAmount = 0.0
48. Do, set string userCode, bool validCode = false
49. Do, read userCode
50. For (nt I=0; I < size; ++i)
51. If coffee[i].code == userCode statement is true, go to step 52
52. Print coffee[i].name, coffee[i].price
53. validCode = true int quantity = get_quantity() totalAmount += coffee[i].price * quantity coffee[i].quantity += quantity
54. if !validCode statement is true, go to step 55 otherwise go to step 56
55. print “Invalid code, please enter the code by following the menu”, go to step 49
56. while (!validCode)
57. while (add_on_items())
58. print totalAmount
59. handle_payment(totalAmount, coffee, size);
60. Return
61. End input_drink_code function



Print Receipt

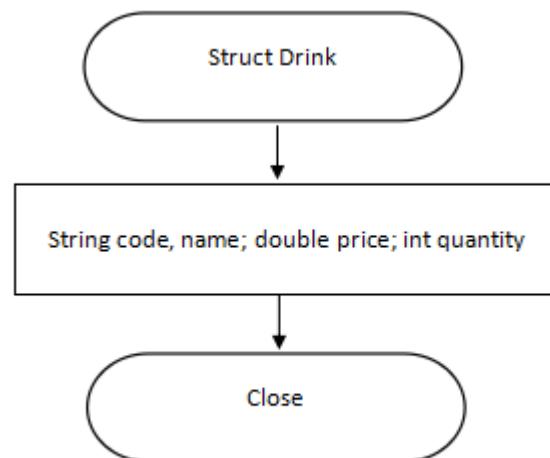
1. Start
2. Input array drink objects, size of the array, total amount, payment and change
3. Open a “receipt.txt” file
4. Display the receipt including (date, time, ordered drinks with quantity and price, total amount, payment, and change)
5. End



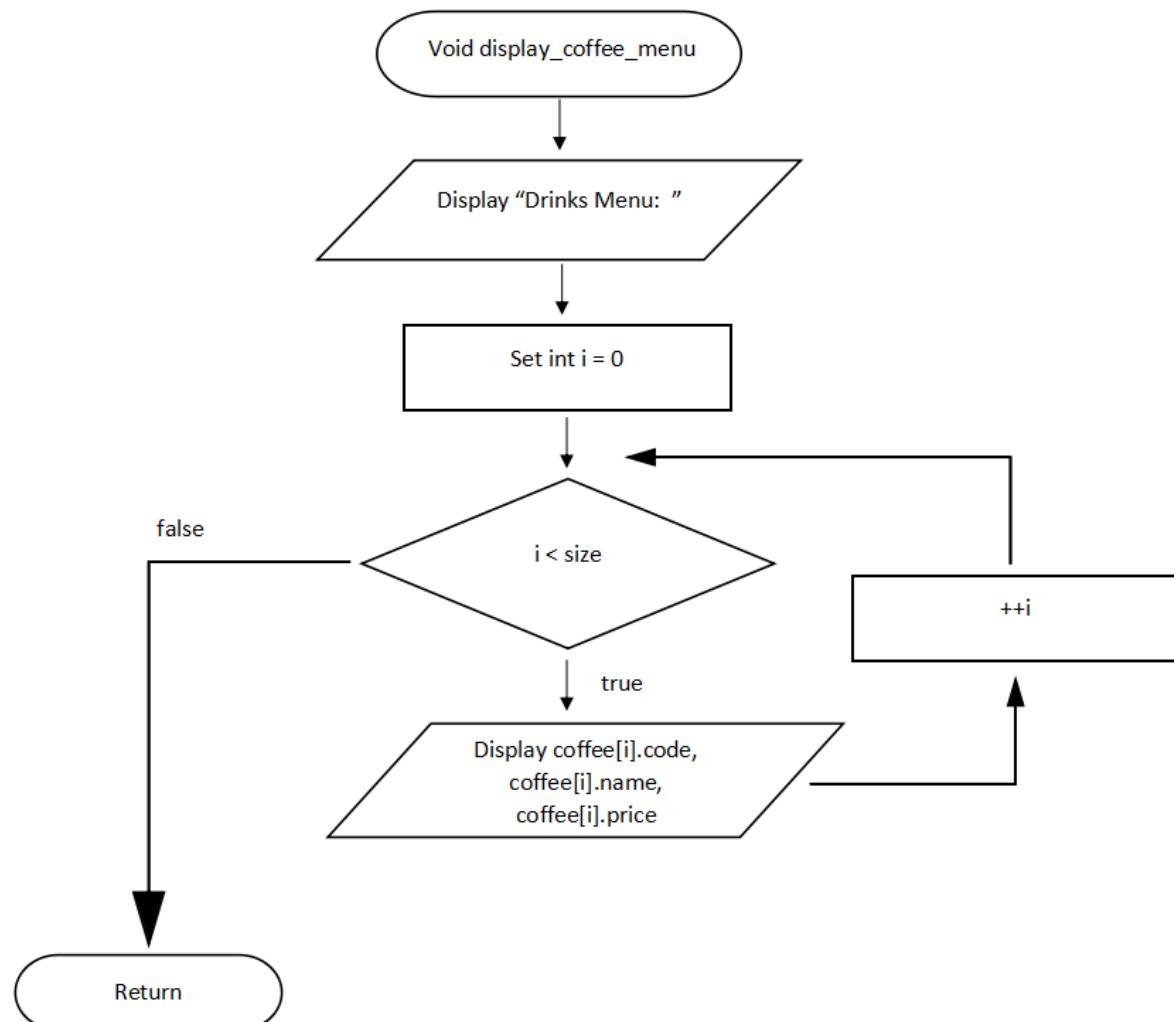
SCHOOL OF COMPUTING

FLOWCHART

Define a struct for drinks



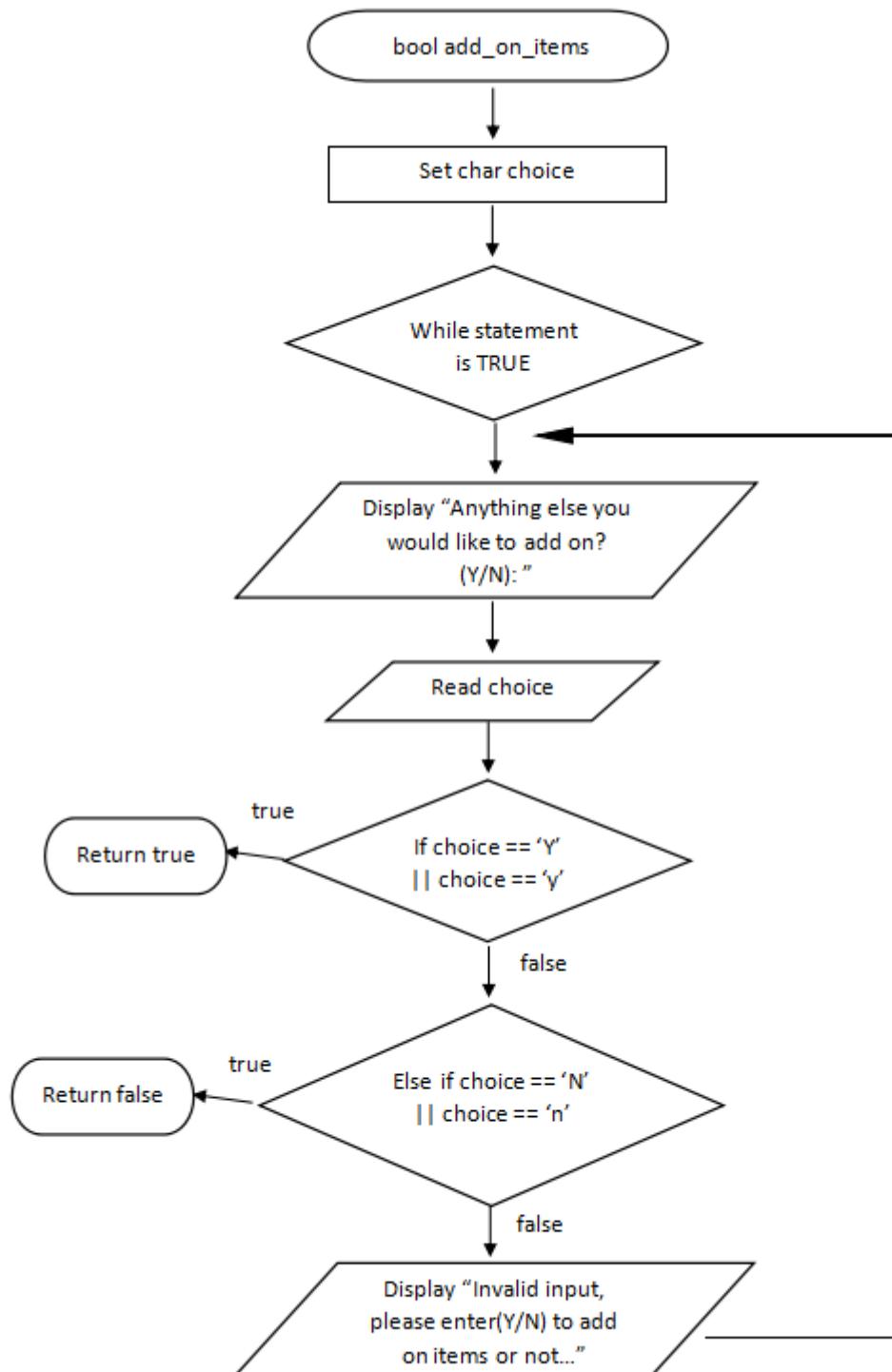
Display coffee menu





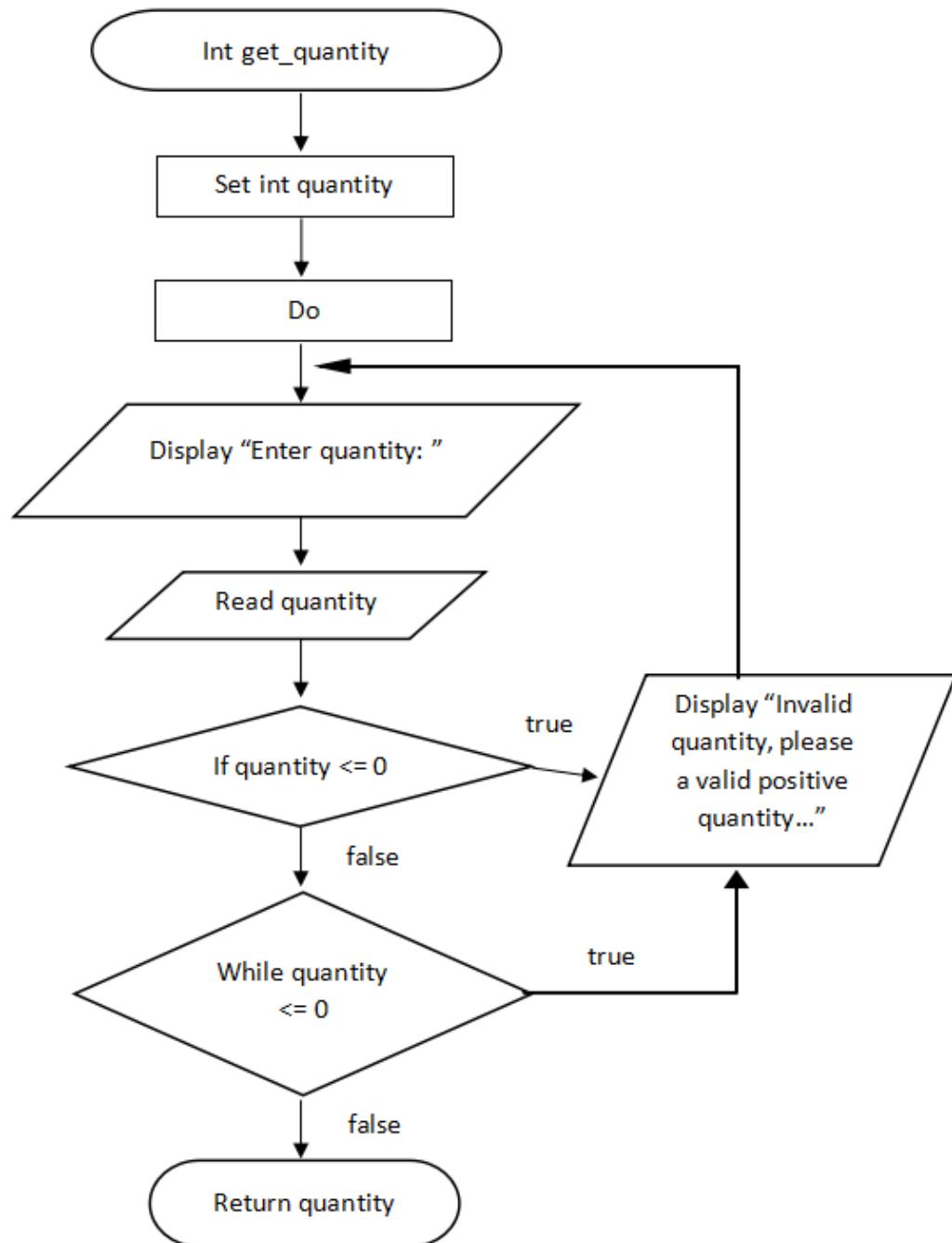
SCHOOL OF COMPUTING

Get more items



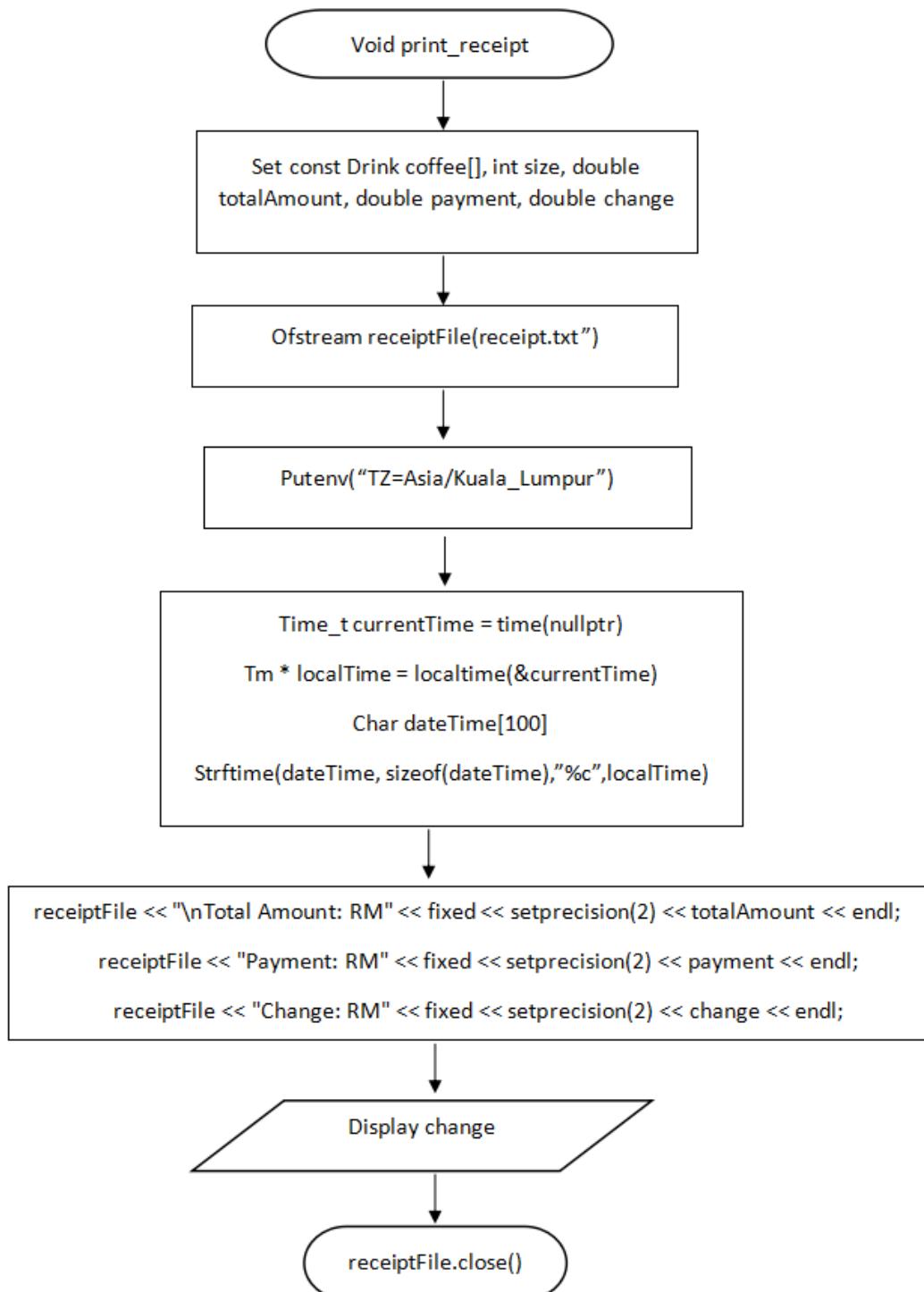


Get quantity



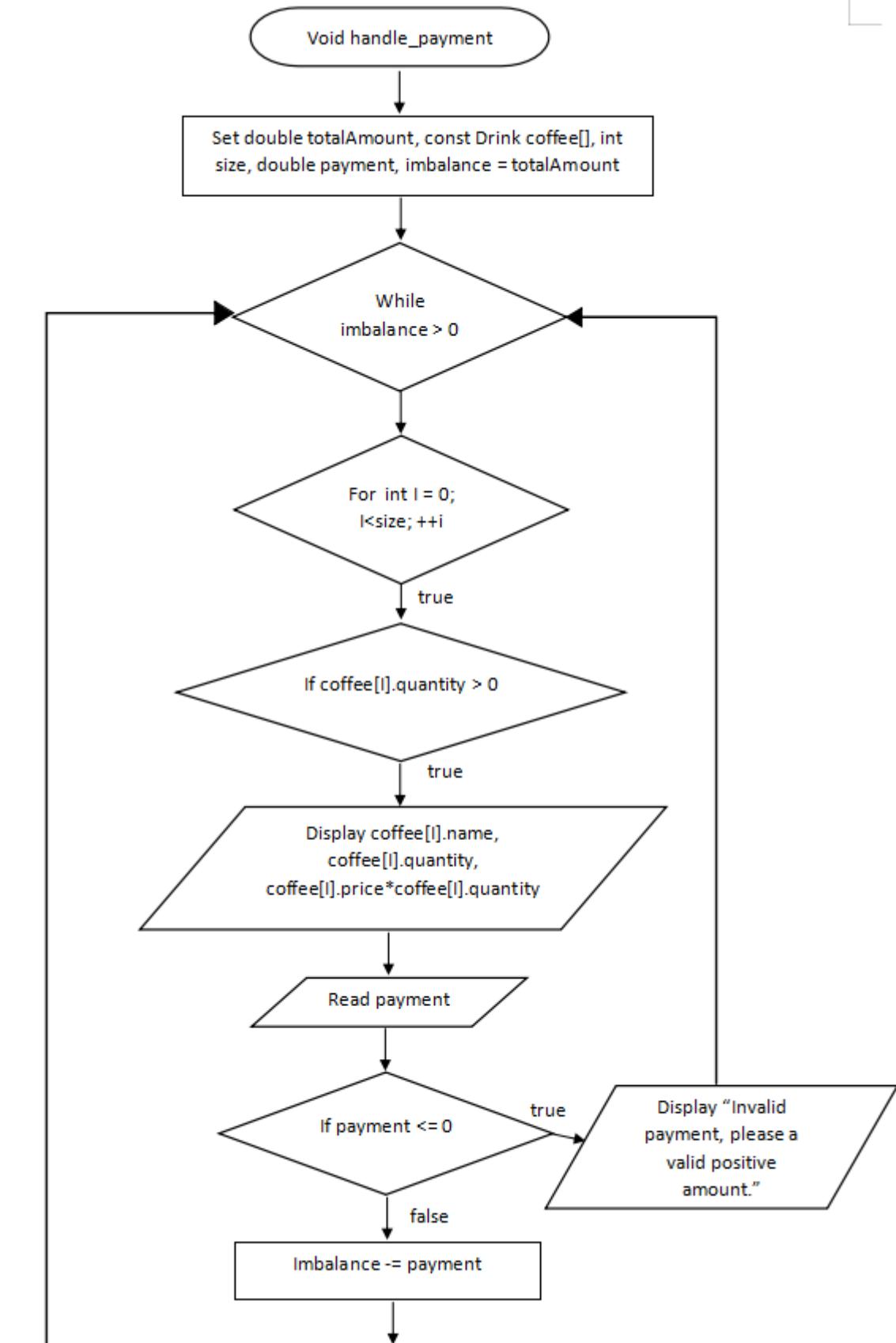


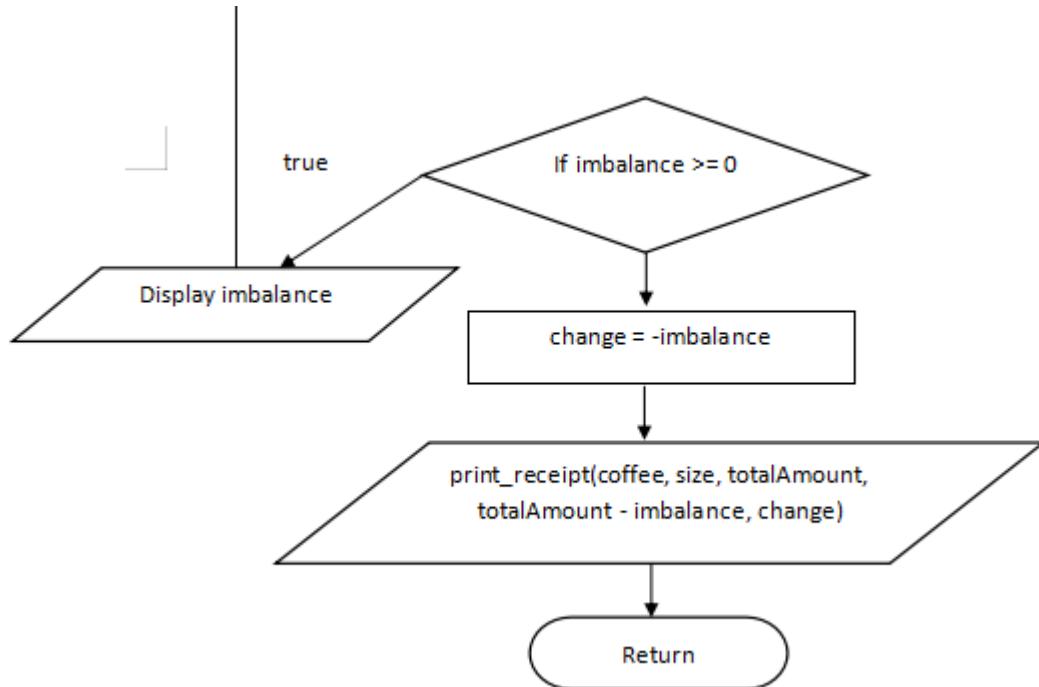
Print receipt





Calculate payment

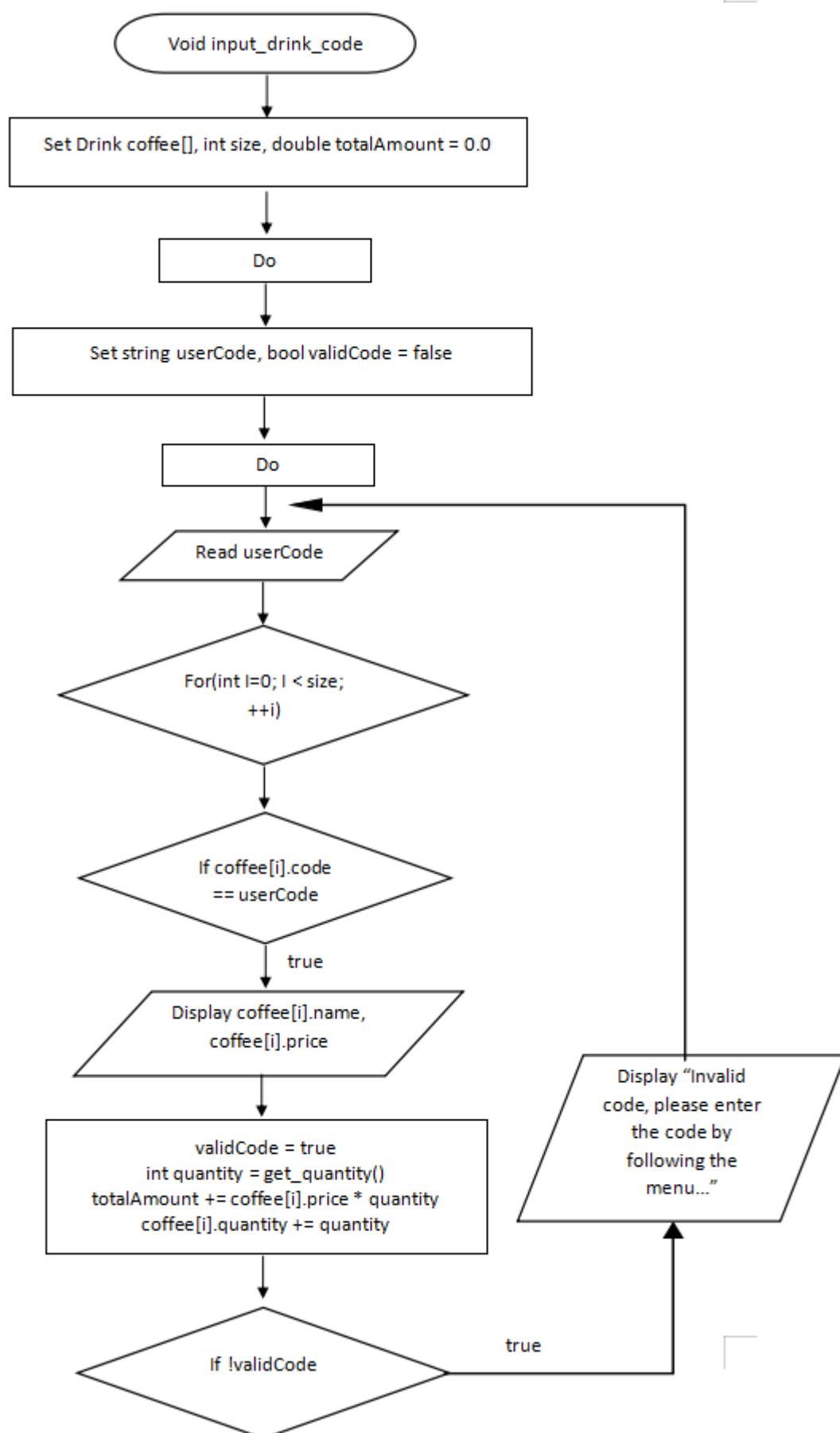


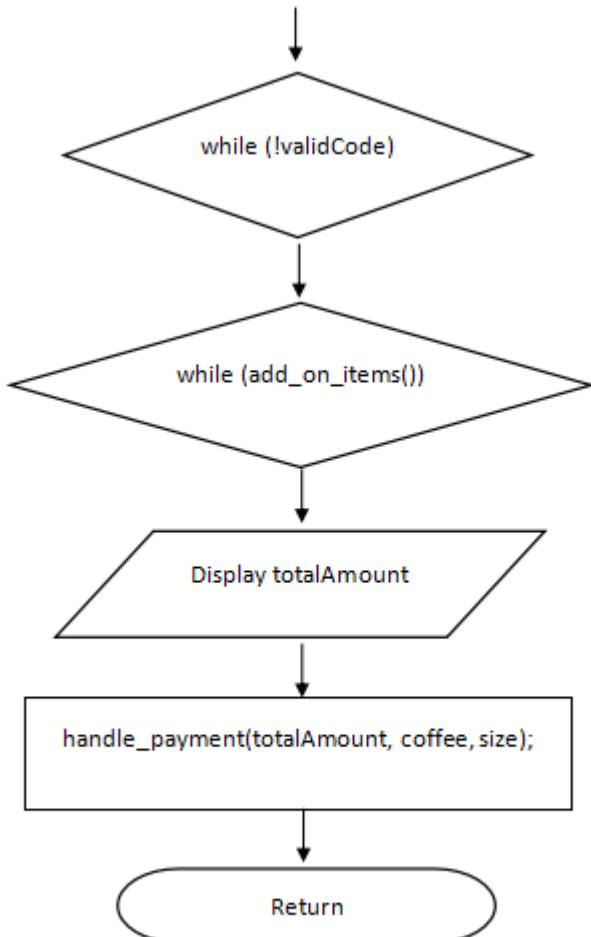




SCHOOL OF COMPUTING

Find a drink by code.

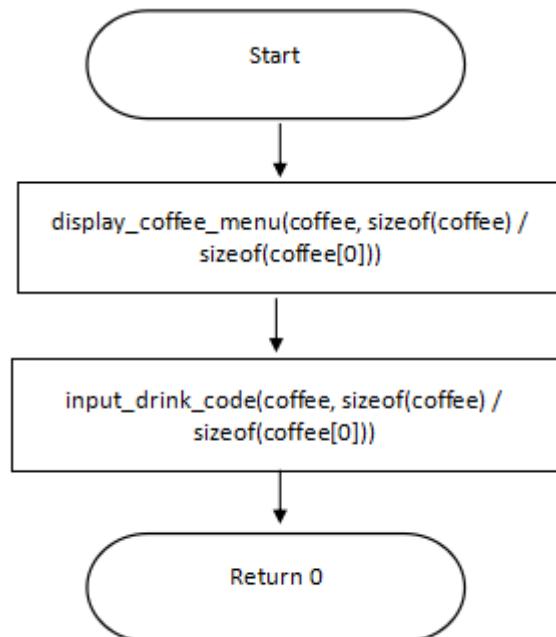






SCHOOL OF COMPUTING

Int main





CODE DESIGN

```
#include <iostream>
#include <iomanip> // For setprecision
#include <limits> // For numeric_limits
#include <fstream> // For file handling
#include <ctime> // For date and time
using namespace std;

// Define a struct for drinks
struct Drink {
    string code;
    string name;
    double price;
    int quantity;
};

// Function: Display coffee menu
void display_coffee_menu(const Drink coffee[], int size) {
    cout << "Drinks Menu:" << endl;
    for (int i = 0; i < size; ++i) {
        cout << coffee[i].code << " = " << coffee[i].name << " (RM" << fixed << setprecision(2)
        << coffee[i].price << ")" << endl;
    }
}

// Function: Get more items
bool add_on_items() {
    char choice;

    while (true) {
        cout << "Anything else you would like to add on? (Y/N): ";
        cin >> choice;

        if (choice == 'Y' || choice == 'y') {
            return true;
        }
    }
}
```



SCHOOL OF COMPUTING

```
    } else if (choice == 'N' || choice == 'n') {  
        return false;  
    } else {  
        cout << "Invalid input, please enter (Y/N) to add on items or not..." << endl;  
        cin.clear(); //Clear the error state flags  
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer  
    }  
}  
  
// Function: Get quantity  
int get_quantity() {  
    int quantity;  
  
    do {  
        cout << "Enter quantity: ";  
        cin >> quantity;  
  
        if (quantity <= 0) {  
            cout << "Invalid quantity, please a valid positive quantity..." << endl;  
            cin.clear(); //Clear the error state flags  
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer  
        }  
    } while (quantity <= 0);  
    return quantity;  
}  
  
// Function: Print receipt  
void print_receipt(const Drink coffee[], int size, double totalAmount, double payment, double change) {  
    ofstream receiptFile("receipt.txt"); // Open receipt from txt file  
  
    // Set time-zone to Malaysia  
    putenv("TZ=Asia/Kuala_Lumpur");
```



SCHOOL OF COMPUTING

```
// Get current date and time
time_t currentTime = time(nullptr);
tm* localTime = localtime(&currentTime);
char dateTime[100];
strftime(dateTime, sizeof(dateTime), "%c", localTime);

receiptFile << "Receipt: " << dateTime << endl; // Include date and time
receiptFile << "-----" << endl;
receiptFile << "Drinks Ordered:" << endl;

for (int i = 0; i < size; ++i) {
    if (coffee[i].quantity > 0) {
        receiptFile << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed <<
setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl;
    }
}

receiptFile << "\nTotal Amount: RM" << fixed << setprecision(2) << totalAmount << endl;
receiptFile << "Payment: RM" << fixed << setprecision(2) << payment << endl;
receiptFile << "Change: RM" << fixed << setprecision(2) << change << endl;

cout << "Thank you for your purchase! Here is your change: RM" << fixed <<
setprecision(2) << change << endl;
cout << "Do enjoy your coffee, have a nice day!" << endl;

receiptFile.close(); // Close the file
}

// Function: Calculate payment
void handle_payment(double totalAmount, const Drink coffee[], int size) {
    double payment, imbalance = totalAmount;

    while (imbalance > 0) {

        for (int i = 0; i < size; ++i) {
```



SCHOOL OF COMPUTING

```
if (coffee[i].quantity > 0) {  
    cout << coffee[i].name << " x" << coffee[i].quantity << " = RM" << fixed <<  
setprecision(2) << (coffee[i].price * coffee[i].quantity) << endl;  
}  
}  
  
cout << "Enter payment amount RM: ";  
cin >> payment;  
  
if (cin.fail() || payment <= 0) {  
    cout << "Invalid payment, please enter a valid positive amount." << endl;  
    cin.clear(); // Clear the error state flags  
    cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer  
} else {  
    imbalance -= payment;  
    if (imbalance > 0) {  
        cout << "Insufficient payment! You still have to pay RM" << fixed << setprecision(2)  
<< imbalance << " in order to purchase your coffee." << endl;  
    }  
}  
}  
  
// Calculate change  
double change = -imbalance;  
  
// Generate receipt from 'Function: print receipt'  
print_receipt(coffee, size, totalAmount, totalAmount - imbalance, change);  
}  
  
// Function: Find a drink by code  
void input_drink_code(Drink coffee[], int size) {  
    double totalAmount = 0.0;  
  
    do {  
        string userCode;
```



SCHOOL OF COMPUTING

```
bool validCode = false;

do {
    cout << "Enter your coffee by following the menu code:" << endl;
    cin >> userCode;

    for (int i = 0; i < size; ++i) {
        if (coffee[i].code == userCode) {
            cout << "You selected " << coffee[i].name << " (RM" << fixed << setprecision(2)
            << coffee[i].price << ")" << endl;
            validCode = true;
            // Ask quantity using 'Function: Get quantity'
            int quantity = get_quantity();
            // Calculate total amount
            totalAmount += coffee[i].price * quantity;
            // Update quantity in the coffee array
            coffee[i].quantity += quantity;
        }
    }
    if (!validCode) {
        cout << "Invalid code, please enter the code by following the menu..." << endl;
        cin.clear(); //Clear the error state flags
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Clear the input buffer
    }
} while (!validCode);

// Ask add more items using 'Function: Get more items'
} while (add_on_items());

cout << "Total amount is RM: " << fixed << setprecision(2) << totalAmount << endl;

handle_payment(totalAmount, coffee, size);
}

int main() {
    Drink coffee[] = {
```



SCHOOL OF COMPUTING

```
{"H1", "Hot-Light Coffee", 5.0},  
 {"H2", "Hot-Medium Coffee", 5.0},  
 {"H3", "Hot-Dark Coffee", 5.0},  
 {"H4", "Hot-Latte", 8.0},  
 {"H5", "Hot-Mocha", 10.0},  
  
 {"C1", "Ice-Light Coffee", 7.0},  
 {"C2", "Ice-Medium Coffee", 7.0},  
 {"C3", "Ice-Dark Coffee", 7.0},  
 {"C4", "Ice-Latte", 10.0},  
 {"C5", "Ice-Mocha", 12.0}  
};  
  
display_coffee_menu(coffee, sizeof(coffee) / sizeof(coffee[0]));  
input_drink_code(coffee, sizeof(coffee) / sizeof(coffee[0]));  
  
return 0;  
}
```



SCREENSHOT OUTPUT

```
Drinks Menu:  
H1 = Hot-Light Coffee (RM5.00)  
H2 = Hot-Medium Coffee (RM5.00)  
H3 = Hot-Dark Coffee (RM5.00)  
H4 = Hot-Latte (RM8.00)  
H5 = Hot-Mocha (RM10.00)  
C1 = Ice-Light Coffee (RM7.00)  
C2 = Ice-Medium Coffee (RM7.00)  
C3 = Ice-Dark Coffee (RM7.00)  
C4 = Ice-Latte (RM10.00)  
C5 = Ice-Mocha (RM12.00)  
Enter your coffee by following the menu code:
```

Figure 4.1

Figure 4.1 display about coffee menu, it display about the code of coffee, the type of coffee and the price of each coffee. User can input the coffee code at below to select the coffee that they want.

```
Enter your coffee by following the menu code:  
C5  
You selected Ice-Mocha (RM12.00)  
Enter quantity: 3  
Anything else you would like to add on? (Y/N): Y  
Enter your coffee by following the menu code:  
A10  
Invalid code, please enter the code by following the menu...  
Enter your coffee by following the menu code:  
H1  
You selected Hot-Light Coffee (RM5.00)  
Enter quantity: 2  
Anything else you would like to add on? (Y/N): N
```

Figure 4.2

When user already input the coffee code that they want, the program will start run to another process. First, the program will ask user about the quantity that user select just now. After input the quantity, program will ask user did user need add another coffee or not, if user need add more another coffee, then user need input the word “Y” or “y” to let program loop again let user input another coffee that they want, if no need add another coffee, then the user need to input the word “N” or “n” to let program stop looping and let program run it to next step.



```
Total amount is RM: 46.00
Hot-Light Coffee x2 = RM10.00
Ice-Mocha x3 = RM36.00
Enter payment amount RM: 40
Insufficient payment! You still have to pay RM6.00 in order to purchase your coffee.
Hot-Light Coffee x2 = RM10.00
Ice-Mocha x3 = RM36.00
Enter payment amount RM: 10
Thank you for your purchase! Here is your change: RM4.00
Do enjoy your coffee, have a nice day!
```

Figure 4.3

Figure 4.3 is show about totalAmount calculation and payment details. The program will output the final totalAmount first, the coffee that user select and the coffee quantity that user select will output in next roll. Next step is payment details, when user input value is less than totalAmount, then program will recommend user input the other value. If user input value is more than totalAmount, then program will find the change back and print out receipt in Figure 4.4.

```
Receipt: Tue Feb 27 13:26:04 2024
-----
Drinks Ordered:
Hot-Light Coffee x2 = RM10.00
Ice-Mocha x3 = RM36.00

Total Amount: RM46.00
Payment: RM50.00
Change: RM4.00
```

Figure 4.4

Figure 4.4 is show about receipt. Receipt will print out the current local time and date first, then program will print out the coffee details that user select just now. Program will also print out the payments details at below.



SUMMARY

We came to the realization that there are still a great deal of codes that we need to understand and explore after finishing the assignment that our instructor had given us. Through this research, we have gained an awareness of the infinite possibilities of codes and their practical applications in everyday life. We have studied numerous coding examples and tutorials during our hands-on lessons, and this has been included into our project. Some of the codes were difficult for us to understand during our investigation, but with perseverance and effort, we were able to use the codes correctly to solve the mistakes. When we implemented the functions, designs, arrays, and receipt codes, a great deal of new information was introduced. We had come up with a lot of designs for our project and finally settled on one.

We had no idea that programming would be as difficult as it is, and as a result, we greatly respect and admire those that work in this field. Hopefully, one day, we will be able to become experts in this. Even though we are still novices, we were able to do this with the assistance of W3School, YouTube, and the guidance of our older peers.

We are happy that, despite having several conversations, we can finish our job on schedule. A few essential elements for accomplishing this task would be effective group collaboration and understanding when one another is unable to complete a task on schedule. As long as we have the dedication to learn, we know we will succeed in the near future.