



CENTRE OF DIPLOMA STUDIES

PRINCIPLE OF PROGRAMMING

DAT 10603

SEMESTER 2 SESSION 2023

PROJECT TITLE :

REAL-WORLD FOOD AND BEVERAGES COMPANY

KILLER GOURMET BURGERS (KGB)

GROUP	1
SECTION	3
GROUP MEMBER	1. AFRINA FARZANA BINTI AZARUDDIN (AA221505) 2. EZAMIRUL SYAKIR BIN SHEIKH ABDUL HALIM (AA221947) 3. HAZMAN IRFAN BIN AHSAN (AA221623) 4. JUITA EIYSAH BINTI SUHAINI (AA220239) 5. MUHAMMAD QAID UQAIL BIN KHAIRUL ANUAR (AA221626) 6. NUR ALMAS SHAZIRENE FASHALIN BINTI MUHAMMAD MUZAFAR (AA222155)
LECTURER NAME	ENCIK FAWWAZ BIN MOHD NASIR

TABLE OF CONTENTS

Content	Page
1.0 Introduction	
1.1 Company Profile	3
1.2 Company Products	4
2.0 Algorithm Design	
• Main Flowchart Figure	5
• Adds-On Flowchart Figure	6
• Beef Burger Flowchart Figure	7
• Chicken Burger Flowchart Figure	7
• Vegetarian Burger Flowchart Figure	8
• Drink Category Choice Flowchart Figure	9
• Food Category Choice Flowchart Figure	10
• Input Validation Flowchart Figure	11
• Payment Input Validation Flowchart Figure	12
• Quantity Input Validation Flowchart Figure	13
• Receipt Calculate Flowchart Figure	14
• Receipt Template Flowchart Figure	14
• Set Meal Flowchart Figure	15
• Beef Burger Set Meal Flowchart Figure	15
• Chicken Burger Set Meal Flowchart Figure	16
• Vegetarian Burger Set Meal Flowchart Figure	17
• Staff Name Flowchart Figure	17
• Summary Value Flowchart Figure	18
3.0 Program Code	19-46
4.0 Sample Output	47-56
5.0 References	56
6.0 Rubric	
• Product Rubric	57
• Report Rubric	58
• Presentation Rubric	59

1.0 INTRODUCTION

1.1 COMPANY PROFILE

KGB stands for Killer Gourmet Burgers. It is a casual dining space that serves handcrafted burgers, fries, and desserts. They are known for their juicy burgers made with serious passion for the passionate. They have several locations in Malaysia including IOI City Mall, Pavilion KL, Bangsar, TTDI, Gardens Midvalley, Subang SS15, and Courtyard Cyberjaya1.

KGB believes that the best burgers can only be made with the best ingredients. Their brioche buns are baked daily and their beef, chicken, and vegetarian patties are freshly prepared daily. They have a range of burger styles including beef burgers style', chicken burgers that are cooked on the grill or fried to golden, and grilled vegetarian patties and mushroom options. They also have a range of sides including regular fries, specialty handcrafted fries, milkshakes, and drinks. Their burger toppings come in the form of old-school combinations like caramelized onions and sharp cheddar to espresso bacon jam with maple-baked cornflakes.



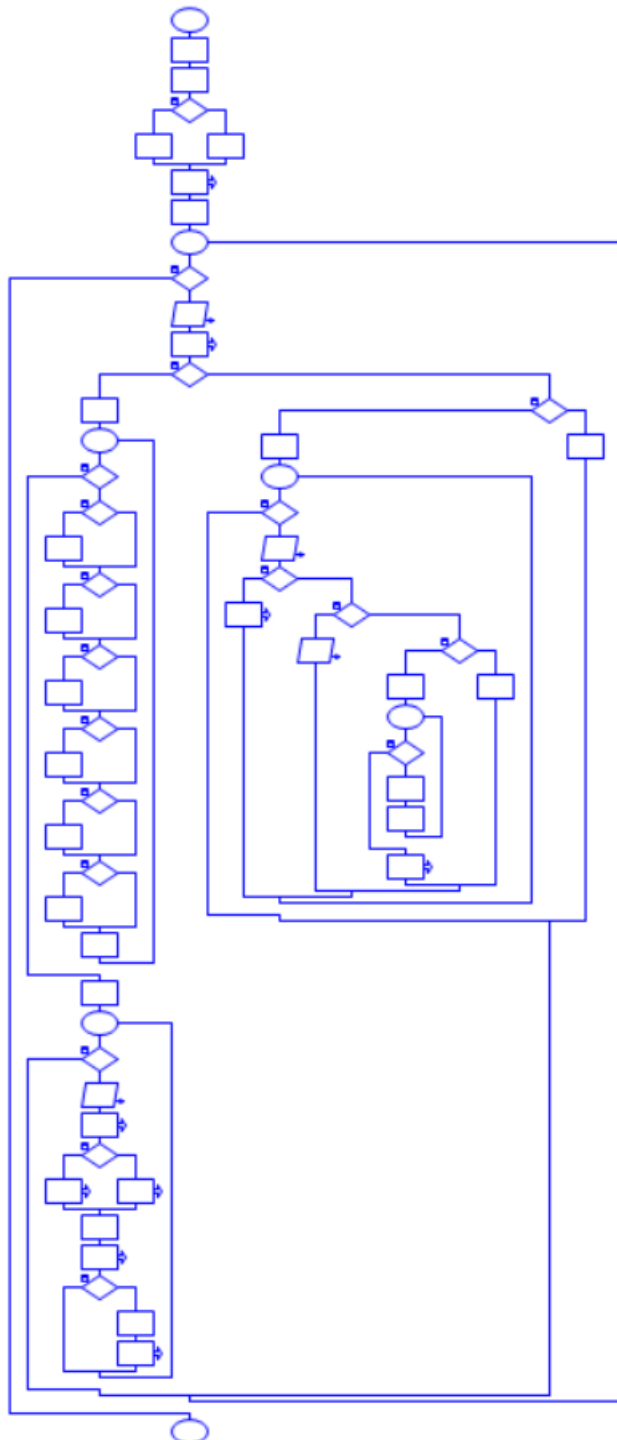
KGB Logo

1.2 COMPANY PRODUCTS

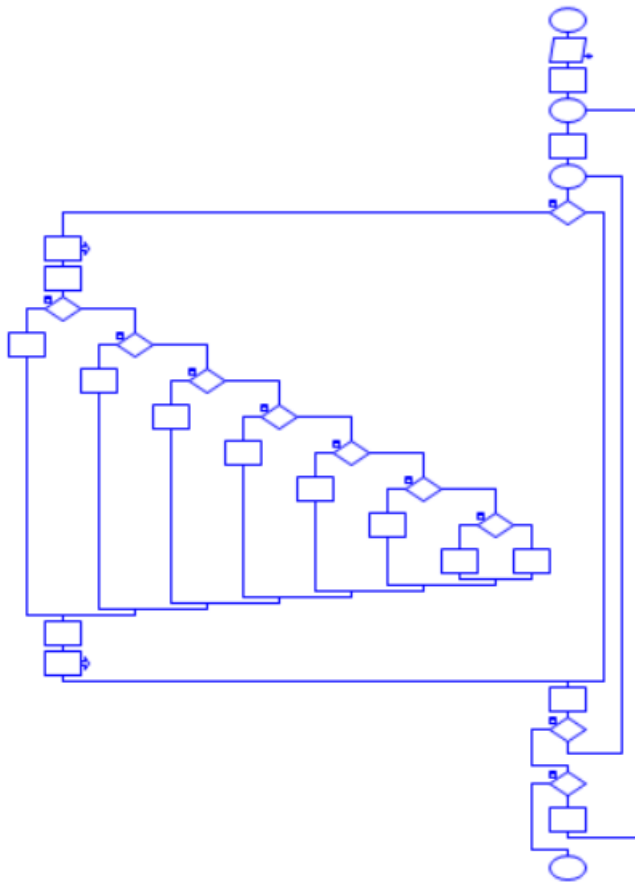
Category	Sub category		Items	Add-on		Set			Price (RM)			
				Items	Price (RM)	Meal	Beverage	Price (RM)				
Food	Burger	Beef	The Shack	Egg	2.00	Salt & pepper fries	Lemonade	+ RM 5.90	12.90			
			Animal Style						14.90			
			Double Animal	Chicken ham					19.90			
			KGB Spiked	Caramelised onion	2.50					15.90		
			Tornado	Sliced cheese						15.90		
			Bash			Herbed fries	Ice mango tea		18.90			
			Bella Bomb Tower	Avocado	3.00				21.90			
			Sacre bleu!						18.90			
			Truffle & swish mushroom	Bun		20.90						
			Bobcat	Blue cheese			15.90					
		Chicken	The Shack	Beef bacon	3.50	Spicy Cajun fries	Honey lemon		12.90			
			Tornado						15.90			
			Bella Bomb	Bacon jam relish					21.90			
			Bobcat						15.90			
			Klub	Portbello mushrooms	4.00	Malt vinegar fries	Ice lemon tea		18.90			
			Seoul		15.90							
			Hot Mess	Grilled chicken	6.00				18.90			
			Kickin'		15.90							
		Vegetarian	Sunshine Bella	Fried chicken	6.50				14.90			
			Vegan Belia Bomb	Smashed/Grilled beef	7.00				15.90			
		Sides		Onion rings (Buttermilk fried)							6.90	
				Chicken bites with dips							7.90	
Potato wedges with dips												
Salt & pepper fries												
Herbed fries												
Spicy Cajun fries												
Malt vinegar fries												
Truffle'd fries	10.90											
Kimchi fries with bulgogi beef	13.90											
Kimchi fries fully loaded	16.90											
Texas jalapeno cheese fries	14.90											
Drinks	Others			Lemonade								
		Ice mango tea										
		Honey lemon										
		Ice lemon tea		6.90								
		Ice lavender tea										
		Strawberry lemonade										
		Root beer float										
	Coke float											
	Milkshakes	Nutella S'mores							11.90			
		Strawberry season										
		Salted caramel Pralines										

2.0 ALGORITHM DESIGN

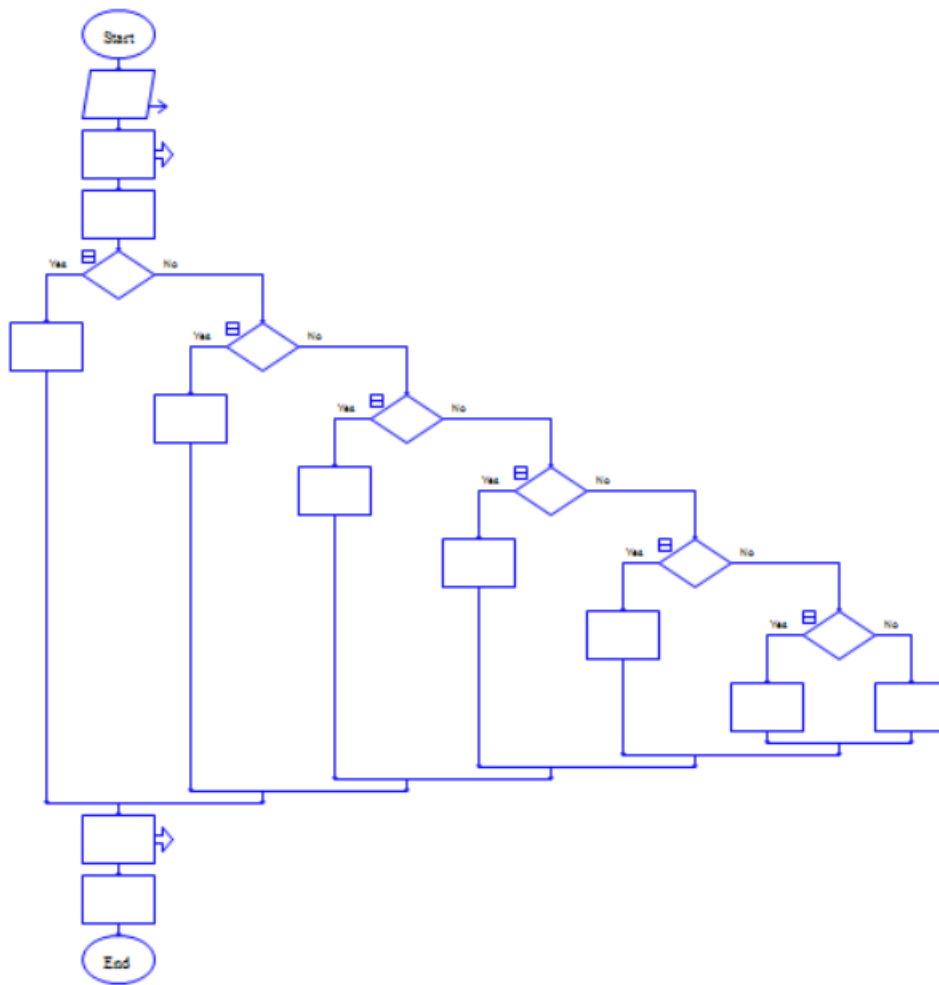
1. Main Flowchart Figure



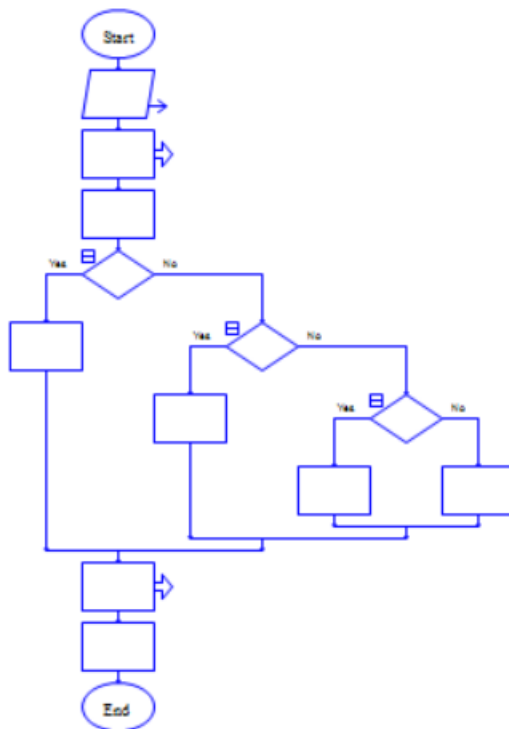
2. Adds-On Flowchart Figure



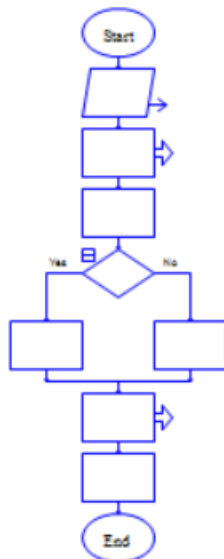
3. Beef Burger Flowchart Figure



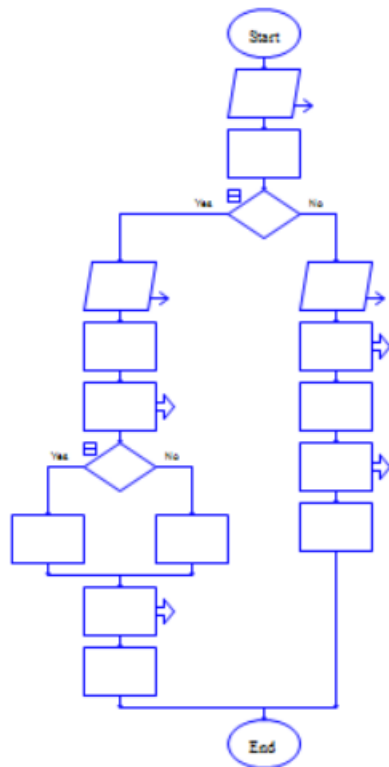
4. Chicken Burger Flowchart Figure



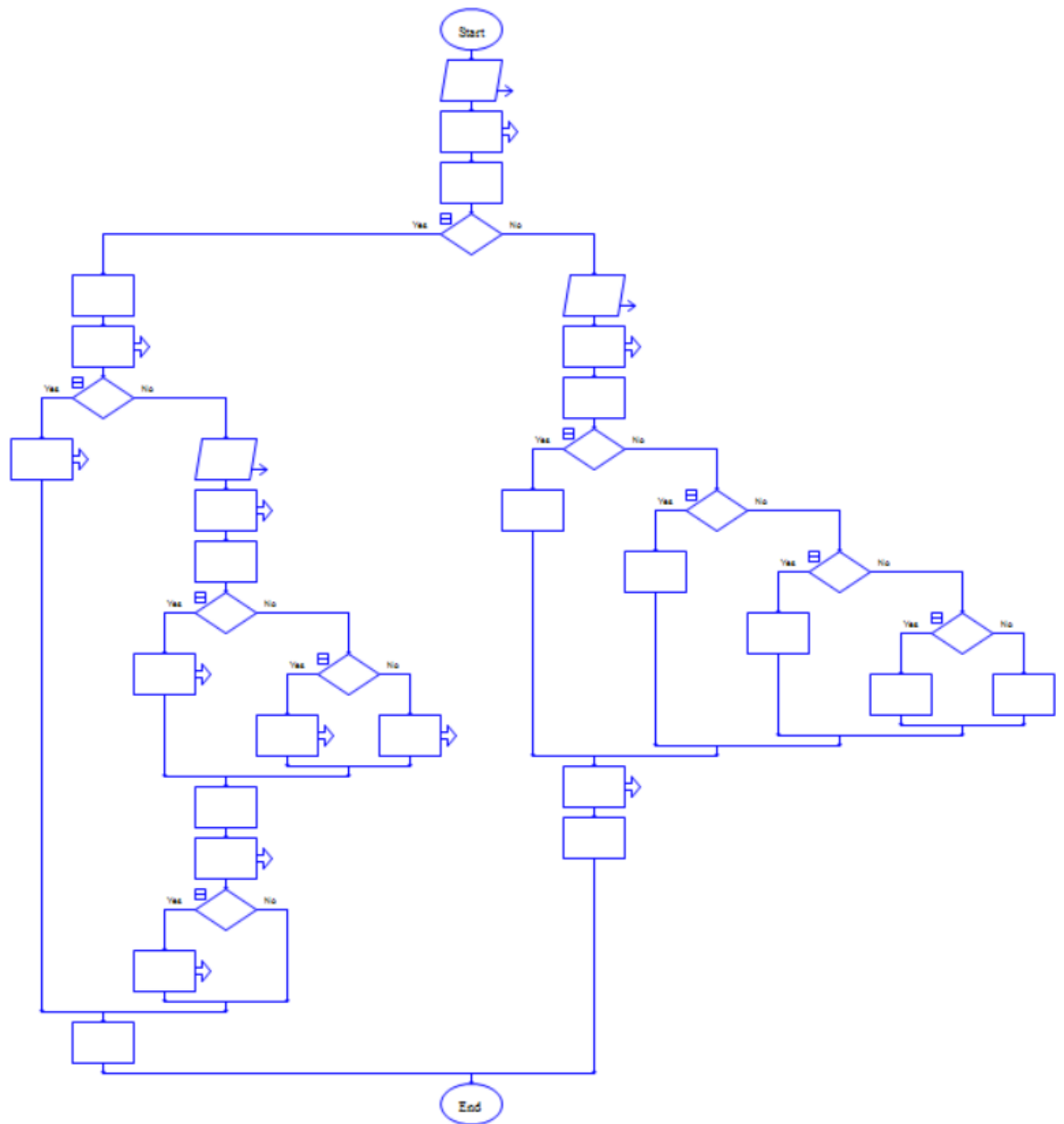
5. Vegetarian Burger Flowchart Figure



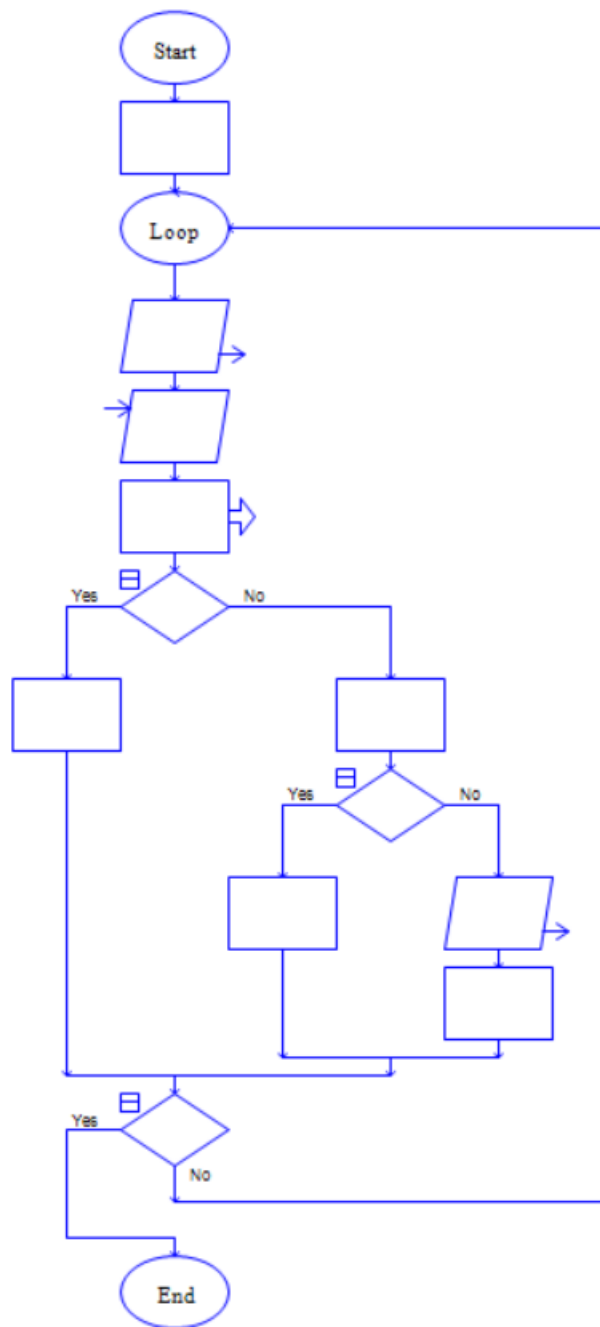
6. Drink category Choice Flowchart Figure



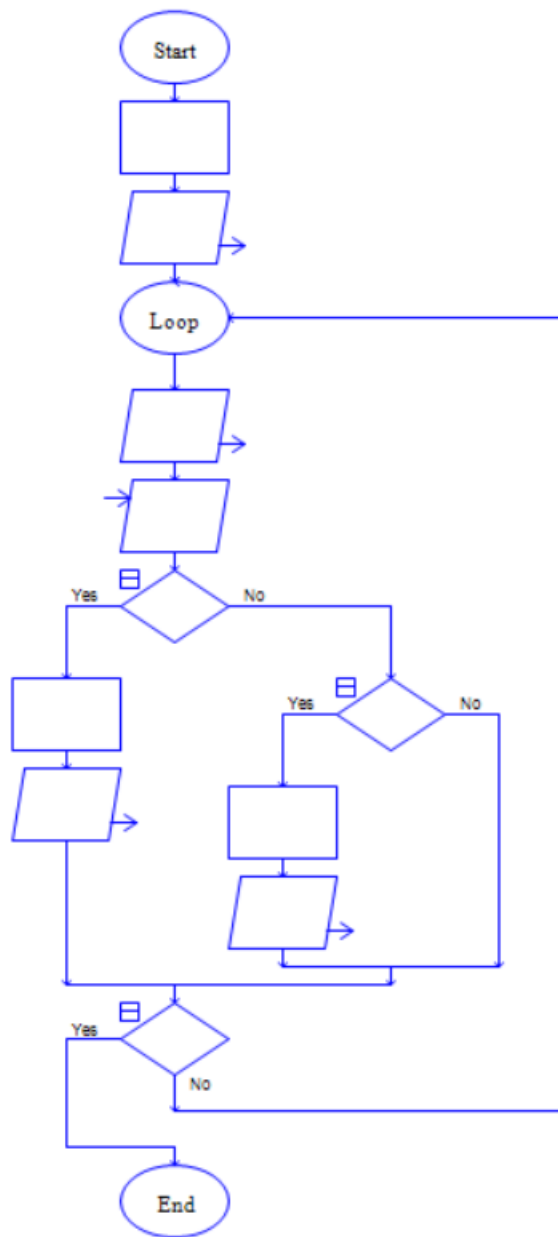
7. Food Category Choice Flowchart Figure



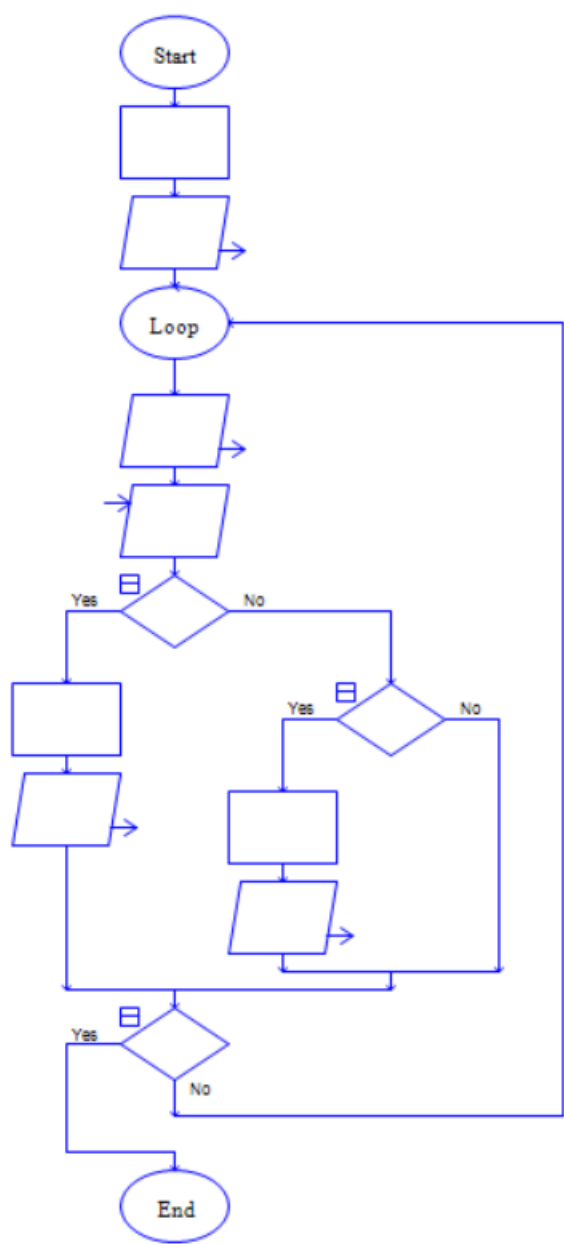
8. Input Validation Flowchart Figure



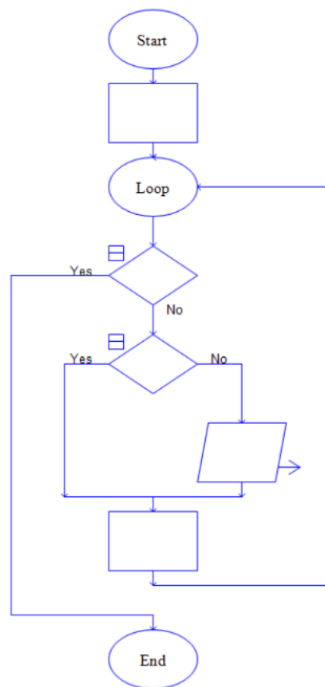
9. Payment Input Validation Flowchart Figure



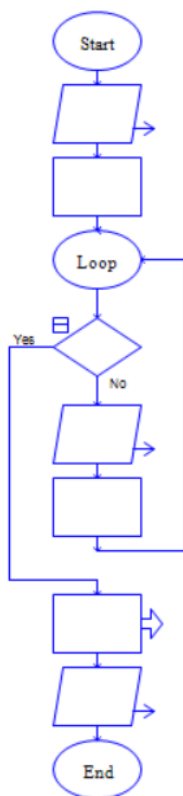
10. Quantity Input Validation Flowchart Figure



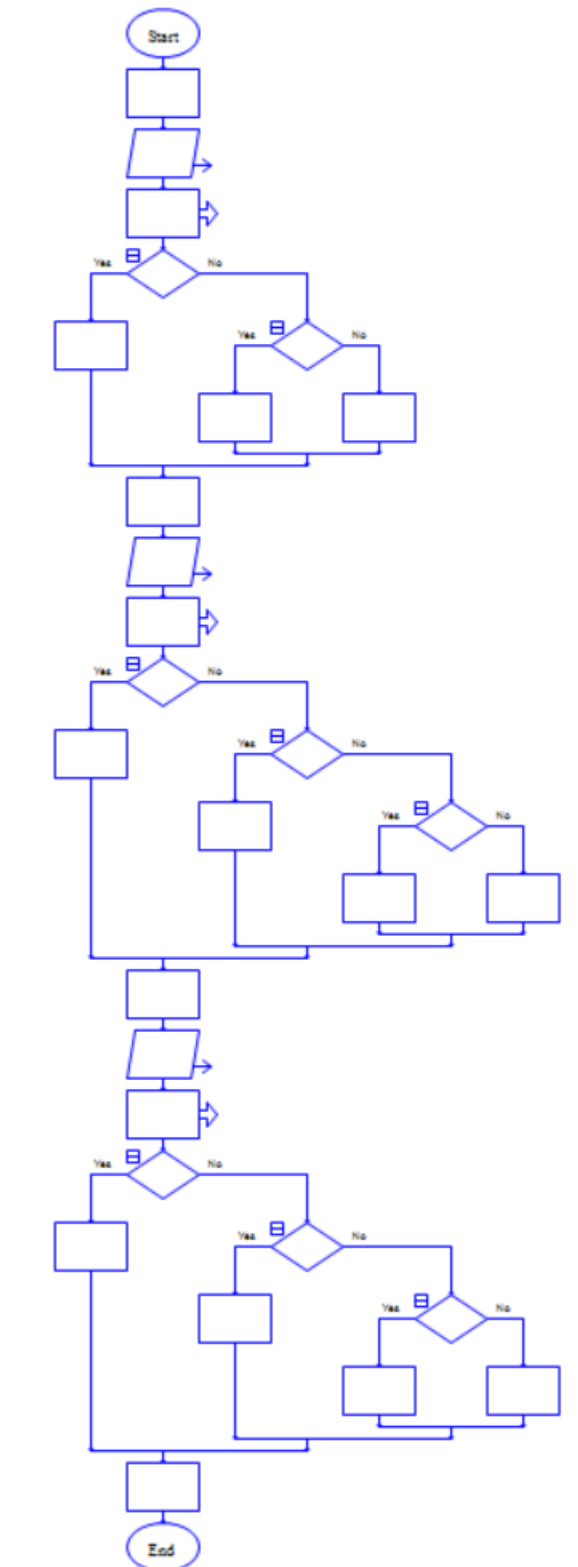
11. Receipt Calculate Flowchart Figure



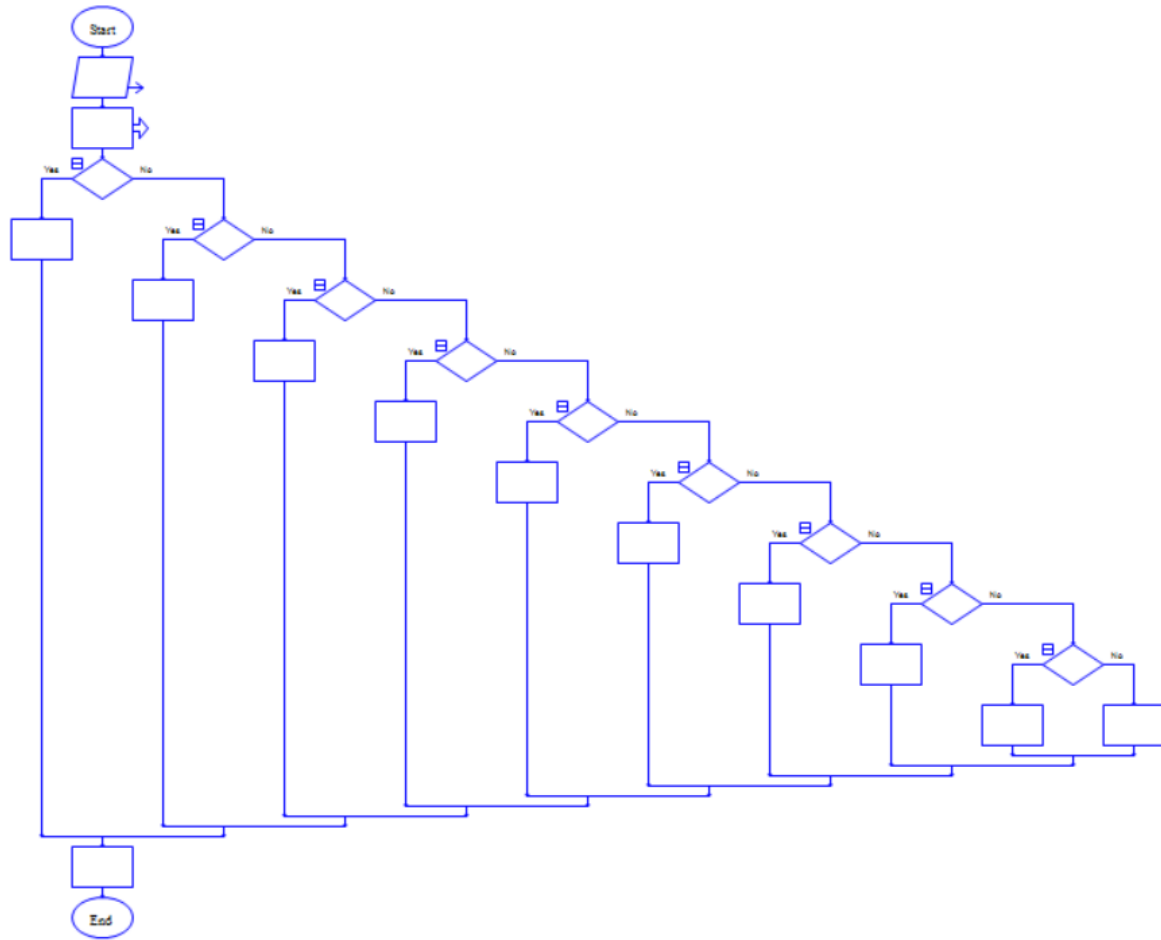
12. Receipt Template Flowchart Figure



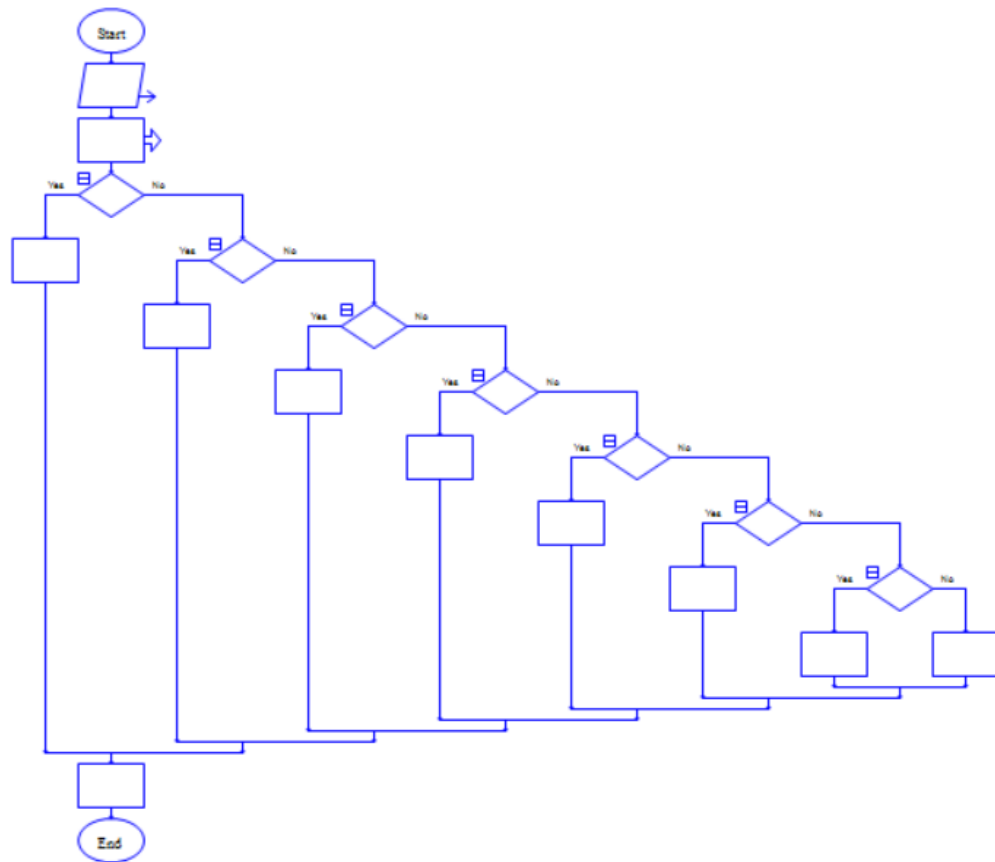
13. Set Meal Flowchart Figure



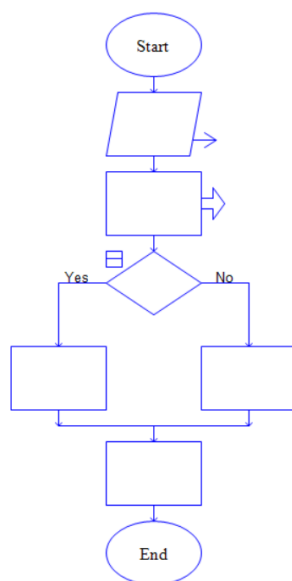
14. Beef Burger Set Meal Flowchart Figure



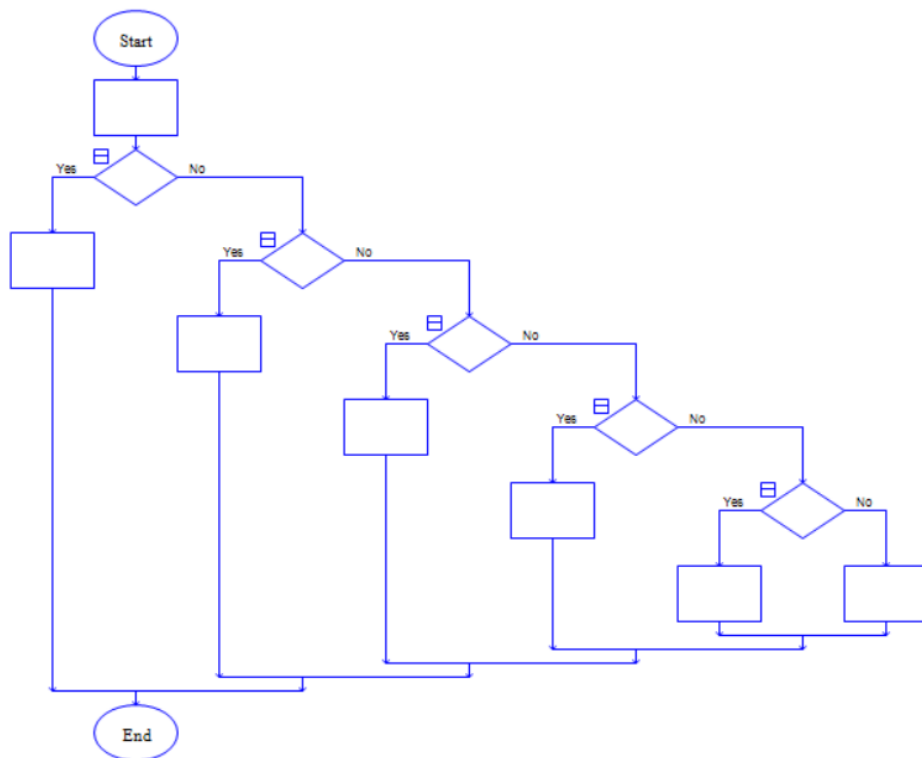
15. Chicken Burger Set Meal Flowchart Figure



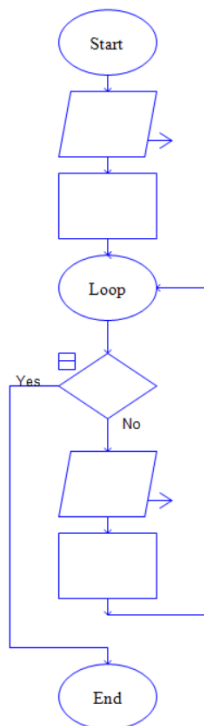
16. Vegetarian Burger Set Meal Flowchart Figure



17. Staff Name Flowchart Figure



18. Summary Value Flowchart Figure



2.0 PROGRAM CODE

```
//SECTION 3 GROUP 1
//EZAMIRUL SYAKIR BIN SHEKH ABDUL HALIM
//AA221947
//MUHAMMAD QAIT QAITIL BIN KHAIRUL ANUAR
//AA221636
//HAZMAN Irfan Bin Ahsan
//AA221623
//NUR ALIMS SHAZIRENE FASHALIN BINTI MUHAMMAD MUZAFAR
//AA222155
//JUITA ETYSAH BINTI SURIAINI
//AA228259
//AFRINA FARZANA BINTI AZARUDDIN
//AA221585
#include<iostream>
#include<iomanip>
#include<string>
#include<ctime>
using namespace std;

//Global Variables
int category[2] = { 0,0 }, subCategoryFood[2] = { 0,0 }, subCategoryDrinks[2] = { 0,0 }, subCategoryBurger[3] = { 0,0,0 }, burgerbeef[10] = { 0,0,0,0,0,0,0,0,0,0 },
burgerchicken[8] = { 0,0,0,0,0,0,0,0 }, burgervege[2] = { 0,0 }, addons[13] = { 0,0,0,0,0,0,0,0,0,0,0,0,0 }, setmealF[3] = { 0,0,0,0 },
setmealB[4] = { 0,0,0,0 }, sidesItems[11] = { 0,0,0,0,0,0,0,0,0,0,0 }, drinks[8] = { 0,0,0,0,0,0,0,0 }, drinkN[3] = { 0,0,0 }, maxSetmeal;
string category[2] = { "Food", "Drinks" }, subCategoryFood[2] = { "Burger", "Sides" }, subCategoryDrinks[2] = { "Others", "Milkshakes" },
subCategoryBurger[3] = { "Burger Beef", "Burger Chicken", "Burger Vegetarian" },
burgerbeefN[10] = { "The Shack [Beef]", "Animal Style", "Double Animal", "Tornado [Beef]", "Bash",
"Bella Bomb Tamer", "Sacre Bleu!", "Truffle & Swiss Mushroom", "Bobcat [Beef]", "MGB Spiced",
burgerchickenN[8] = { "The Shack [Chicken]", "Tornado [Chicken]", "Bella Bomb", "Bobcat [Chicken]", "Klub",
"Hot Mess", "Kickin'",
burgervegeN[2] = { "Sunshine Bella", "Vegan Bella Bomb",
addonsN[13] = { "Egg", "Chicken Ham", "Caramelized Onion", "Sliced Cheese", "Avocado", "Bun",
"BBlue Cheese",
"Beef Bacon", "Bacon Jam Relish", "Portobello Mushroom", "Grilled Chicken", "Fried Chicken", "Smashed Beef",
setmealF[3] = { "Salt & Pepper Fries", "Herbed Fries", "Spicy Cajun Fries", "Malt Vinegar Fries" }, setmealB[4] = { "Lemonade", "Ice Mango Tea", "Honey Lemon", "Ice Lemon Tea" },
sidesItems[11] = { "Onion Rings (Buttermilk Fried)", "Chicken Bites with Dips", "Potato Wedges with Dips", "Salt & Pepper Fries", "Herbed Fries",
"Spicy Cajun Fries", "Malt Vinegar Fries", "Truffle'd fries", "Kimchi Fries with Bulgogi Beef", "Kimchi Fries Fully Loaded", "Texas Jalapeno Cheese Fries",
drinkN[3] = { "Lemonade", "Ice Mango Tea", "Honey Lemon", "Ice Lemon Tea", "Ice Lemon Tea", "Ice Lavender Tea", "Strawberry Lemonade",
"Root Beer Float", "Coke Float",
drinksN[3] = { "MuteLLa S'mores", "Strawberry Season", "Salted Caramel Pralines",
setmealFries[999], setmealBeverages[999], setmealS[999];
double priceItemsBeef[10] = { 12.99, 14.99, 19.99, 15.99, 15.99, 18.99, 21.99, 18.99, 20.99, 15.99 }, priceItemsChicken[8] = { 12.99, 15.99, 21.99, 15.99, 18.99, 15.99, 18.99, 15.99 }, priceItemsVege[2] = { 14.99, 15.99 },
priceItemsAddons[13] = { 2.99, 2.99, 2.50, 2.50, 3.99, 3.99, 3.99, 3.59, 3.59, 4.99, 6.99, 6.59, 7.99 },
priceItemsSides[11] = { 6.99, 7.99, 7.99, 7.99, 7.99, 7.99, 10.99, 13.99, 16.99, 14.99 }, priceItemsDrinks[11] = { 5.99, 5.99, 5.99, 5.99, 6.99, 6.99, 6.99, 11.99, 11.99, 11.99 }, setmealPrice[100];
const int EIGHT = 8, TWO = 2, THIRTEEN = 13, ELEVEN = 11, THREE = 3, TEN = 10;
int quantitybeef[TEN] = { 0,0,0,0,0,0,0,0,0,0 }, quantitychicken[EIGHT] = { 0,0,0,0,0,0,0,0 },
quantityvege[TWO] = { 0,0 },
quantityaddons[THIRTEEN] = { 0,0,0,0,0,0,0,0,0,0,0,0,0 },
quantitysides[ELEVEN] = { 0,0,0,0,0,0,0,0,0,0,0 },
quantitydrinksO[EIGHT] = { 0,0,0,0,0,0,0,0 },
quantitydrinksN[THREE] = { 0,0,0 };
```

Figure 1: Main Console

In this C++ program, the program used #include iostream, iomanip, string and ctime. The program also used using namespace std. Then, the program declared all global variables.

```
//Define Module Function
int inputValidation(string ask, int maxValue, int lower, string error);
int inputValidationQ(string ask);
double inputValidationPayment(double currenttotalPrice);
string staffName(double hour);
void receiptTemplate(double currenttotalPrice, double hour, double minute, string time, int month, int year, string staff, double totalpriceItems, int day, double payment);
void receiptCalculate(int maxValue, string itemsN[], double price[], int quantity[]);
void rankingSummaries(int maxValue, int itemsQ[], string itemsN[], string itemsRanking);
void categoryFoodChoice(double& totalpriceItems);
void categoryDrinksChoice(double& totalpriceItems);
double burgerBeef(int& value);
double burgerChicken(int& value);
double burgerVegetarian(int& value);
void addOns(double& totalpriceItems, int quantityburger, double priceItems);
void setmeal(double& totalpriceItems, double priceItems);
double setmealburgerBeef(double priceItems);
double setmealburgerChicken(double priceItems);
double setmealburgerVege(double priceItems);
void summaryValue(int maxValue, int itemsQ[], string itemsN[], string itemsRanking);
```

Figure 2: Main Console

In this C++ program, the program define module functions to prototype functions.

```

int main() {
    #pragma warning(disable: 4996)
    //current time
    time_t now = time(0);
    tm* localTime = localtime(&now);
    int hour = localTime->tm_hour;
    int minute = localTime->tm_min;
    int year = localTime->tm_year;
    int day = localTime->tm_mday;
    int month = localTime->tm_mon;
    string time;
    if (hour > 11.59)
    {
        time = "PM";
    }
    else
    {
        time = "AM";
    }
    string staff = staffName(hour);
    int value = 0;
    double summaryTotalSales = 0;
    bool loopMenu = true;
}

```

Figure 3: Main Function (time)

In this C++ program, it defines the current time by using time_t function and tm*localtime(&now).

We use integer hour, minute, year, day, month as variables defined.

If else statements were used to determine the current AM/PM format.

String staff declared and assigned by function staffName.

Variables value, summaryTotalSales and loopMenu were declared for the looping process.

```

while (loopMenu == true) {
    cout << "\n\nMAIN MENU " << endl;
    cout << "1. Receive order from the customer \n2. Generate reports\n3. Exit the program\n" << endl;
    value = inputValidation("Enter Choice ", 3, 1, "Choice");
}

```

Figure 4: Main Function (main menu)

In this C++ program, it displays Main Menu choices which is to receive order from the customer, generate reports and exit the program. Value is determined by inputValidation function to determine the user choice.

```

if (value == 1) {
    for (int i = 0; i < 13; i++)
    {
        if (i < 2)
        {
            quantityvege[i] = 0;
        }
        if (i < 3) {
            quantitydrinksM[i] = 0;
        }
        if (i < 8){
            quantitychicken[i] = 0;
            quantitydrinksO[i] = 0;
        }
        if (i < 10){
            quantitybeef[i] = 0;
        }
        if (i < 11)
        {
            quantitiesides[i] = 0;
        }
        if (i < 13)
        {
            quantityaddons[i] = 0;
        }
    }

    double totalpriceItems = 0, payment, currenttotalPrice = 0;
    maxsetmeal = 0;
    int order = 1;
}

```

Figure 5: Main Function (main menu choice 1)

In this C++ program, for loop statement use to declare and define variables for quantity in receipts.

Declaration and define to 0 for variables totalpriceItems, payment, currenttotalPrice and maxsetmeal.

Integer order defined to 1 for loop in food menu order.

```

while (order == 1)
{
#pragma warning(disable:6385)
    cout << "\n\nWELCOME TO MGB" << "\nMy name is " << staff << endl;
    cout << "-----" << endl;
    cout << "Category\n" << endl;
    cout << "1. Food\n2. Drinks" << endl;
    value = inputValidation("Insert category code (1-2): ", 2, 1, "category");
    if (value == 1)
    {
        categoryFoodChoice(totalpriceItems);
    }
    else if (value == 2) {
        categoryDrinksChoice(totalpriceItems);
    }
    value = value - 1;
    category[value] += 1;
    order = inputValidation("Add more order? (Yes-1/No-2)", 2, 1, "choice");
    if (order == 1) {
    }
    else {
        currenttotalPrice = totalpriceItems + totalpriceItems * 0.06;
        payment = inputValidationPayment(currenttotalPrice);
        receiptTemplate(currenttotalPrice, hour, minute, time, month, year, staff, totalpriceItems, day, payment);
        summaryTotalSales += currenttotalPrice;
    }
}
}

```

Figure 6: Main Function (Menu Choice 1)

In this C++ program, it displays name of restaurant and staff operated in that hour. Then, it displays category choice which is food and drinks. Value is determined by inputValidation function to determine the user choice. If else statement used to call function within user's choice. The category[] array is plus one for the choice chosen. Choice is determined by inputValidation function to determine the user choice. If else statement choice used to run user's choice in which to add more order or continue to receipt. Currenttotalprice variable is defined by sum of totalpriceItems and service tax 6%.Payment is determined by inputValidationPayment function to determine the user payment is valid or not.Call function receiptTemplate for formatted receipt display and summaryTotalSales is added with current currenttotalPrice value.

```

else if (value == 2)
{
    bool loopQ = true;
    while (loopQ == true)
    {
        cout << "\n\nGENERATE REPORTS" << endl;
        cout << "\n\n1. Quantity sold for each product for the day\n2. Total sales for the day\n3. Ranking of the most ordered product for the day\n4. Return to Main Menu" << endl;
        value = inputValidation("Enter Choice ", 4, 1, "Choice");
    }
}

```

Figure 7: Main Function (main menu choice 2)

In this C++ program, Boolean loopQ is defined as true for loopQ. The program displays generate reports sub-menu choices which it has three choices. Choice 1 for the quantity sold for each product for the day, choice 2 is the total sales for the day and choice 3 is the ranking of the most ordered product of the day and choice 4 is a return to the main menu. Value is determined by inputValidation function to determine the user choice.

```

if (value == 1)
{
    cout << "\nSUMMARY\n" << endl;
    summaryValue(2, category, categoryN, "CATEGORY");
    summaryValue(2, subCategoryFood, subCategoryFoodN, "FOOD");
    summaryValue(2, subCategoryDrinks, subCategoryDrinksN, "DRINKS");
    summaryValue(3, subCategoryBurger, subCategoryBurgerN, "BURGER");
    summaryValue(10, burgerbeef, burgerbeefN, "BEEF BURGER");
    summaryValue(8, burgerchicken, burgerchickenN, "CHICKEN BURGER");
    summaryValue(2, burgervege, burgervegeN, "VEGETARIAN BURGER");
    summaryValue(13, addons, addonsN, "ADD-ONS");
    summaryValue(4, setmealF, setmealFN, "SET MEAL FRIES");
    summaryValue(4, setmealBev, setmealBevN, "SET MEAL BEVERAGES");
    summaryValue(11, sidesItems, sidesItemsN, "SIDES");
    summaryValue(8, drinks, drinksN, "OTHER DRINKS");
    summaryValue(3, drinks, drinksNM, "MILKSHAKES");
    cout << "\n\n";
}

```

Figure 8: Main Function (main menu choice 2)

In this C++ program, if else statement used to call submenu user's choice. In Choice 1, the program displays summary for quantity sold for each product for the day using function summaryValue function.

```

else if (value == 2)
{
    cout << "\nTotal Summary Sales : RM" << fixed << setprecision(2) << summaryTotalSales;
}

```

Figure 9: Main Function (main menu choice 2)

In this C++ program, Choice 2 is used to display total sales for the day.

```

else if (value == 3)
{
    int categoryR[2], subCategoryFoodR[2], subCategoryDrinksR[2], subCategoryBurgerR[3], burgerbeefR[10],
    burgerchickenR[8], burgervegeR[2], addonsR[13], setmealFR[4],
    setmealBevR[4], sidesItemsR[11], drinksR[8], drinksNR[3];
    string categoryNR[2], subCategoryFoodNR[2], subCategoryDrinksNR[2], subCategoryBurgerNR[3],
    burgerbeefNR[10],
    burgerchickenNR[8],
    burgervegeNR[2],
    addonsNR[13],
    setmealFNR[4], setmealBevNR[4],
    sidesItemsNR[11],
    drinksNR[8],
    drinksNMR[3];

    for (int i = 0; i < 13; i++)
    {
        if (i < 2)
        {
            categoryR[i] = category[i];
            categoryNR[i] = categoryN[i];
            subCategoryFoodR[i] = subCategoryFood[i];
            subCategoryFoodNR[i] = subCategoryFoodN[i];
            subCategoryDrinksR[i] = subCategoryDrinks[i];
            subCategoryDrinksNR[i] = subCategoryDrinksN[i];
            burgervegeR[i] = burgervege[i];
            burgervegeNR[i] = burgervegeN[i];
        }

        if (i < 3)
        {
            subCategoryBurgerR[i] = subCategoryBurger[i];
            subCategoryBurgerNR[i] = subCategoryBurgerN[i];
            drinksNR[i] = drinksN[i];
            drinksNMR[i] = drinksNM[i];
        }

        if (i < 4)
        {
            setmealFR[i] = setmealF[i];
            setmealFNR[i] = setmealFN[i];
            setmealBevR[i] = setmealBev[i];
            setmealBevNR[i] = setmealBevN[i];
        }

        if (i < 8)
        {
            burgerchickenR[i] = burgerchicken[i];
            burgerchickenNR[i] = burgerchickenN[i];
            drinksR[i] = drinks[i];
            drinksNR[i] = drinksN[i];
        }

        if (i < 10)
        {
            burgerbeefR[i] = burgerbeef[i];
            burgerbeefNR[i] = burgerbeefN[i];
        }

        if (i < 11)
        {
            sidesItemsR[i] = sidesItems[i];
            sidesItemsNR[i] = sidesItemsN[i];
        }

        if (i < 13)
        {
            addonsR[i] = addons[i];
            addonsNR[i] = addonsN[i];
        }
    }
}

```

Figure 10: Main Function (main menu choice 2)

In this C++ program, choice 3 section program declared new set of arrays for temporary summary. The new arrays are defined by summary arrays for quantity sold for each product for the day.

```

cout << "\nRanking Summary" << endl;
rankingSummaries(2, categoryR, categoryNR, "CATEGORY");
rankingSummaries(2, subCategoryFoodR, subCategoryFoodNR, "FOOD");
rankingSummaries(2, subCategoryDrinksR, subCategoryDrinksNR, "DRINKS");
rankingSummaries(3, subCategoryBurgerR, subCategoryBurgerNR, "BURGER");
rankingSummaries(10, burgerbeefR, burgerbeefNR, "BEEF BURGER");
rankingSummaries(8, burgerchickenR, burgerchickenNR, "CHICKEN BURGER");
rankingSummaries(2, burgervegeR, burgervegeNR, "VEGETARIAN BURGER");
rankingSummaries(13, addonsR, addonsNR, "ADD-ONS");
rankingSummaries(4, setmealFR, setmealFNR, "SET MEAL FRIES");
rankingSummaries(4, setmealBevR, setmealBevNR, "SET MEAL BEVERAGES");
rankingSummaries(11, sidesItemsR, sidesItemsNR, "SIDES");
rankingSummaries(8, drinksR, drinksNR, "OTHER DRINKS");
rankingSummaries(3, drinksMR, drinksNMR, "MILKSHAKES");
cout << "\n\n\n";
}
else
{
    loopQ = false;
}
}
else
{
    loopMenu = false;
}
}

```

Figure 11: Main Function (main menu choice 2)

In this C++ program, the program use new temporary arrays to be sort in descending order by rankingSummaries function.

```

int inputValidation(string ask, int maxValue, int lower, string error)
{
    int value;
    bool inputV = false;
    do
    {
        cout << endl << ask << endl;
        cin >> value;
        cout << "\033[2J\033[1;1H";
        if (cin.fail() || value < lower || value > maxValue)
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid " << error << ".Please enter 1 to " << maxValue << " only.";
        }
        else
        {
            string remainingInput;
            getline(cin, remainingInput);
            if (remainingInput.empty()) {
                break; // Input is a valid integer
                inputV = true;
            }
            else {
                cout << "Invalid input. Please enter an integer value." << endl;
                inputV = false;
            }
        }
    } while (cin.fail() || value < lower || value > maxValue || not inputV);

    return value;
}

```

Figure 12: Function inputValidation

In this C++ program, the program gets variables ask, maxValue, lower and error as parameters. Variables value and inputV is declared. The line `cout << "\033[2J\033[1;1H";` used to clears the entire screen and moves the cursor to the top-left corner of the screen. By printing these escape codes to the console, we achieve a similar effect to clearing the screen. The program display ask and input the user's input to value, then if else statement used to determine whether the user's input is valid or not. The else statement used if user's input wrong data type.

```

int inputValidationQ(string ask)
{
    int value;
    do
    {
        cout << endl << ask << endl;
        cin >> value;
        cout << "\033[2J\033[1;1H";
        if (cin.fail())
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid quantity.Please enter number for quantity";
        }
        else if (value < 1)
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid quantity.Please enter at least 1";
        }
        else {}
    } while (cin.fail());
    return value;
}

```

Figure 13: Function inputValidationQ

In this C++ program, the program gets variables ask, maxValue, lower and error as parameters. Variables value and inputV is declared. The line `cout << "\033[2J\033[1;1H";` used to clears the entire screen and moves the cursor to the top-left corner of the screen. By printing these escape codes to the console, we achieve a similar effect to clearing the screen. The program display ask and input the user's input to value, then if else statement used to determine whether the user's input is valid or not. The else if statement used if user's input is less than 1.


```
double inputValidationPayment(double currenttotalPrice) {
    double payment;
    cout << "Total price : RM" << fixed << setprecision(2) << currenttotalPrice;
    do
    {
        cout << ".\nEnter payment : RM";
        cin >> payment;
        if (cin.fail())
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\nInvalid Amount.Please Pay RM" << currenttotalPrice;
        }
        else if (payment < currenttotalPrice)
        {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "\nInsufficient Amount.Please Pay RM" << currenttotalPrice;
        }
    } while (cin.fail() || payment < currenttotalPrice);
    return payment;
}
```

Figure 14: Function double inputValidationPayment

In this C++ program, the program define double variable payment. Then, it display the total price for the order. The do while loop is used to repeat if user's input payment is invalid. If else statement used to validate the user's input payment. The return function return payment to main.

```
string staffName(double hour) {
    string staff;

    if (hour >= 0 && hour <= 3.59)
    {
        staff = "Ezamirul Syakir";
    }
    else if (hour >= 4 && hour <= 7.59)
    {
        staff = "Hazman Irfan";
    }
    else if (hour >= 8 && hour <= 11.59)
    {
        staff = "Nur Almas";
    }
    else if (hour >= 12 && hour <= 15.59)
    {
        staff = "Qaid Uqail";
    }
    else if (hour >= 16 && hour <= 19.59)
    {
        staff = "Juita Eiyasah";
    }
    else if (hour >= 20 && hour <= 23.59)
    {
        staff = "Afrina Farzana";
    }
    else {}
    return staff;
}
```

Figure 15: Function String staff Name

In this C++ program, it gets data hour from parameter and declare new string staff. If else statements were used to determine which staff was working in current hour. Then, the staff name will be defined by string staff and return it to main.

```

void categoryFoodChoice(double& totalpriceItems)
{
    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Food" << endl;
    cout << " " << endl;
    cout << "1. Burger " << endl;
    cout << "2. Sides " << endl;
    cout << " " << endl;
    int value = inputValidation("Enter food code(1-2): ", 2, 1, "food code");
    int choice = value;
    if (value == 1)
    {
        double priceItems = 0;
        value = inputValidation("Make it a set meal? (1-Yes, 2-No) ", 2, 1, "choice");
        if (value == 1)
        {
            int maxsetmeal = 0;
            setmeal(totalpriceItems, priceItems);
        }
        else
        {

```

Figure 16: Category Food Choice

In this C++ program, the program displays a menu for the "Food" category, allows the user to choose between burger or sides items, and provides the option to make a selected burger item a set meal. The chosen options and input validations ensure the user's choices are correctly processed in the subsequent code execution.

```

    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Burger " << endl;
    cout << " " << endl;
    cout << "1. Beef " << endl;
    cout << "2. Chicken " << endl;
    cout << "3. Vegetarian " << endl;
    cout << " " << endl;
    value = inputValidation("Insert burger code(1-3): ", 3, 1, "burger's category code");
    int choice = value - 1;
    subCategoryBurger[choice] += 1;
    cout << "-----" << endl;
    cout << " " << endl;

```

Figure 17: Category Food Choice

In this C++ program, the program displays a menu for the "Burger" category and asks the user to choose a burger type (Beef, Chicken, or Vegetarian) by entering a code (1, 2, or 3). The code then updates the count of the chosen burger type in an array. It essentially helps the user select a burger and keeps track of the number of times each burger type has been chosen.

```

if (value == 1)
{
    priceItems = burgerBeef(value);
}
else if (value == 2)
{
    priceItems = burgerChicken(value);
}
else
{
    priceItems = burgerVegetarian(value);
}
totalpriceItems = totalpriceItems + priceItems * value;

```

Figure 18: Category Food Choice

In this C++ program, the code determines the type of burger chosen by the user (beef, chicken, or vegetarian) based on the entered value. It then calculates the price of the chosen burger and updates the total price of the items selected

```

    int quantityburger = value;
    value = inputValidation("Add-Ons to burger?(Yes - 1/No - 2)", 2, 1, "choice");
    if (value == 1)
    {
        addOns(totalPriceItems, quantityburger, priceItems);
    }
    else {};
    choice = choice - 1;
    subCategoryFood[choice] += 1;
}

```

Figure 19: Category Food Choice

In this C++ program, the code handles the user's decision regarding add-ons for the burger and updates the counters accordingly

```

else {
    cout << "-----" << endl;
    cout << "Sides Items" << endl;
    cout << "1. Onion Rings (Buttermilk Fried) RM6.90" << endl;
    cout << "2. Chicken Bites with Dips RM7.90" << endl;
    cout << "3. Potato Wedges with Dips RM7.90" << endl;
    cout << "4. Salt & Pepper Fries RM7.90" << endl;
    cout << "5. Herbed Fries RM7.90" << endl;
    cout << "6. Spicy Cajun Fries RM7.90" << endl;
    cout << "7. Malt Vinegar Fries RM7.90" << endl;
    cout << "8. Truffle Fries RM10.90" << endl;
    cout << "9. Kimchi Fries with Bulgogi Beef RM13.90" << endl;
    cout << "10. Kimchi Fries Fully Loaded RM16.90" << endl;
    cout << "11. Texas Jalapeno Cheese Fries RM14.90" << endl;
    value = inputValidation("Enter sides code(1-11):", 11, 1, "sides code");
    int choice = value;
}

```

Figure 20: Category Food Choice

In this C++ program, the code displays a menu of side items to the user. The user is asked to enter a code corresponding to their desired side item. The code then stores the chosen code in a variable and assigns the price of the selected side item to another variable. This allows the program to keep track of the user's selection and the price associated with it for further calculations or processing.

```

78     double priceItems;
79     if (choice == 1)
80     {
81         priceItems = 6.9;
82     }
83     else if (choice <= 7)
84     {
85         priceItems = 7.9;
86     }
87     else if (choice == 8)
88     {
89         priceItems = 10.9;
90     }
91     else if (choice == 9)
92     {
93         priceItems = 13.9;
94     }
95     else if (choice == 10)
96     {
97         priceItems = 16.9;
98     }
99     else
100    {
101        priceItems = 14.9;
102    }
103    value = inputValidationQ("Enter the quantity");
104    totalPriceItems = totalPriceItems + priceItems * value;
105    choice = choice - 1;
106    quantitysides[choice] += value;
107    sidesItems[choice] += value;
108 }
109 }

```

Figure 21: Category Food Choice

In this C++ program, the program determines the price of a selected item based on the user's choice. It assigns different prices to different items using conditional statements. After that, it prompts the user to enter the quantity of the selected item. The code then calculates the total price by multiplying the price and quantity, and adds it to the overall total price. It also keeps track of the quantity of the selected item in two arrays: quantitysides and sidesItems.

```

void categoryDrinksChoice(double& totalPriceItems)
{
    cout << "-----" << endl;
    cout << "Drinks" << endl;
    cout << "1. Others" << endl;
    cout << "2. Milkshakes" << endl;
    int value = inputValidation("Enter drinks code(1-2):", 2, 1, "drinks code");
    subCategoryDrinks[value] += 1;
    cout << "-----" << endl;
    cout << "

```

Figure 22: Category Drinks choice

In this C++ program, the code displays a menu for choosing between two drink categories: "Others" and "Milkshakes". It asks the user to enter a code (1 or 2) representing their choice. After validating the input, it increments a counter for the chosen drink category. The code also includes formatting lines for visual separation.

```

if (value == 1)
{
    cout << "Other drinks" << endl;
    cout << " " << endl;
    cout << "Item" << "Price(RM)" << endl;
    cout << " " << endl;
    cout << "1. Lemonade" << "5.90" << endl;
    cout << "2. Ice mango tea" << "5.90" << endl;
    cout << "3. Honey lemon" << "5.90" << endl;
    cout << "4. Ice lemon tea" << "5.90" << endl;
    cout << "5. Ice lavender tea" << "6.90" << endl;
    cout << "6. Strawberry lemonade" << "6.90" << endl;
    cout << "7. Root beer float" << "6.90" << endl;
    cout << "8. Coke float" << "6.90" << endl;
    value = inputValidation("Enter the drink's code(1-8): ", 8, 1, "drinks code");
    int choice = value;
}

```

Figure 23: Other Drinks Category

In this C++ program, if the user selects the option for "Other drinks," a list of various drink options with their corresponding prices is displayed. The user is then prompted to enter the code for the desired drink from 1 to 8. The entered value is stored in the variable "choice." This choice represents the specific drink selected by the user.

```

double priceItems;
if (choice <= 4)
{
    priceItems = 5.9;
}
else
{
    priceItems = 6.9;
}
value = inputValidationQ("Enter quantity");
totalpriceItems = totalpriceItems + priceItems * value;
choice = choice - 1;
quantitydrinksO[choice] += value;
drinks[choice] += value;

```

Figure 24: Other Drinks Category

In this C++ program, the price of the selected drink is determined based on the value of the "choice" variable. If the value is less than or equal to 4, the price is set to 5.9; otherwise, it is set to 6.9. The user is then prompted to enter the quantity of the selected drink. The total price of the items is updated by adding the product of the price per item and the quantity. Additionally, the quantity of the selected drink is stored in the "quantitydrinksO" array at the corresponding index (choice - 1), and the total quantity of the drink is stored in the "drinks" array at the same index.

```

else
{
    cout << "Milkshakes" << endl;
    cout << " " << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. Nutella S' mores" << endl;
    cout << "2. Strawberry season" << endl;
    cout << "3. Salted Caramel Pralines" << endl;
    value = inputValidation("Enter milkshake code(1-3): ", 3, 1, "milkshake code");
    int choice = value;
    value = inputValidationQ("Enter quantity");
    totalpriceItems = totalpriceItems + 11.9 * value;
    choice = choice - 1;
    quantitydrinksM[choice] += value;
    drinks[choice] += value;
}
}

```

Figure 25: Milkshakes Category

In the provided code snippet, if the value is not equal to 1, the user is prompted to select a milkshake option. The available milkshake options are displayed, along with their respective prices. The user is then prompted to enter the milkshake code (1-3). The selected milkshake's price is set to 11.9, and the user is prompted to enter the quantity of the selected milkshake. The total price of the items is updated by adding the product of the milkshake price and the quantity. Additionally, the quantity of the selected milkshake is stored in the "quantitydrinksM" array at the corresponding index (choice - 1), and the total quantity of the milkshake is stored in the "drinks" array at the same index.

```

double burgerBeef(int& value)
{
    cout << "Beef burger" << endl;
    cout << " " << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Animal Style" << endl;
    cout << "3. Double Animal" << endl;
    cout << "4. KGB Spiked" << endl;
    cout << "5. Tornado" << endl;
    cout << "6. Bash" << endl;
    cout << "7. Bella Bomb Tower" << endl;
    cout << "8. Sacre Bleu!" << endl;
    cout << "9. Truffle & Swish Mushroom" << endl;
    cout << "10. Bobcat" << endl;
    value = inputValidation("Enter Beef burger code (1-10): ", 10, 1, "beef burger code");
}

```

Figure 26: Ala Carte Beef Burger

In this C++ program, the program displaying a menu of beef burger options and receives user input to select a specific beef burger within the range of 1 to 10.

```

int choice = value;
double priceItems;
if (choice == 1)
{
    priceItems = 12.9;
}
else if (choice == 2)
{
    priceItems = 14.9;
}
else if (choice == 3)
{
    priceItems = 19.9;
}
else if (choice == 4 || choice == 10 || choice == 5)
{
    priceItems = 15.9;
}
else if (choice == 6 || choice == 8)
{
    priceItems = 18.9;
}
else if (choice == 7)
{
    priceItems = 21.9;
}
else
{
    priceItems = 20.9;
}

value = inputValidationQ("Enter quantity");
choice = choice - 1;
quantitybeef[choice] += value;
burgerbeef[choice] += value;

return priceItems;

```

Figure 27: Ala Carte Beef Burger

In this C++ program, the code determines the price of a chosen item based on a variable called "value." It assigns the value to a variable called "choice" and then initializes a variable called "priceItems" based on the value of "choice." Depending on the value of "choice," different prices are set. After that, the program asks the user to enter the quantity of the chosen item and updates the corresponding arrays to keep track of the quantity. Finally, the program returns the value of "priceItems."

```

double burgerChicken(int& value)
{
    cout << "Chicken burger" << endl;
    cout << " " << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Tornado" << endl;
    cout << "3. Bella Bomb" << endl;
    cout << "4. Bobcat" << endl;
    cout << "5. Klub" << endl;
    cout << "6. Seoul" << endl;
    cout << "7. Hot Mess" << endl;
    cout << "8. Kickin" << endl;
    value = inputValidation("Enter Chicken burger code (1-8): ", 8, 1, "chicken burger code");
    int choice = value;
    double priceItems;
    if (choice == 1)
    {
        priceItems = 12.9;
    }
    else
    {
        if (choice == 2 || choice == 4 || choice == 6 || choice == 8)
        {
            priceItems = 15.9;
        }
        else
        {
            if (choice == 5 || choice == 7)
            {
                priceItems = 18.9;
            }
            else
            {
                priceItems = 21.9;
            }
        }
    }
    value = inputValidationQ("Enter quantity");
    choice = choice - 1;
    quantitychicken[choice] += value;
    burgerchicken[choice] += value;

    return priceItems;
}

```

Figure 28: Ala Carte Chicken Burger

In this C++ program, this program defines the functions called burgerChicken that shown a display menu of Ala Carte Chicken Burger with its corresponding prices.

```

double burgerChicken(int& value)
{
    cout << "Chicken burger" << endl;
    cout << " " << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Tornado" << endl;
    cout << "3. Bella Bomb" << endl;
    cout << "4. Bobcat" << endl;
    cout << "5. Klub" << endl;
    cout << "6. Seoul" << endl;
    cout << "7. Hot Mess" << endl;
    cout << "8. Kickin" << endl;
    value = inputValidation("Enter Chicken burger code (1-8): ", 8, 1, "chicken burger code");
    cout << " " << endl;
    cout << "Price(RM)" << endl;
    cout << "12.90" << endl;
    cout << "15.90" << endl;
    cout << "21.90" << endl;
    cout << "15.90" << endl;
    cout << "18.90" << endl;
    cout << "15.90" << endl;
    cout << "18.90" << endl;
    cout << "15.90" << endl;
}

```

Figure 29: Ala Carte Chicken Burger

In this C++ program, the program displaying a menu of chicken burger options and receives user input to select a specific chicken burger within the range of 1 to 8.


```

int choice = value;
double priceItems;
if (choice == 1)
{
    priceItems = 12.9;
}
else
{
    if (choice == 2 || choice == 4 || choice == 6 || choice == 8)
    {
        priceItems = 15.9;
    }
    else
    {
        if (choice == 5 || choice == 7)
        {
            priceItems = 18.9;
        }
        else
        {
            priceItems = 21.9;
        }
    }
}

value = inputValidationQ("Enter quantity");
choice = choice - 1;
quantitychicken[choice] += value;
burgerchicken[choice] += value;

return priceItems;

```

Figure 30: Ala Carte Chicken Burger

This C++ program assigns the value of a variable called "value" to a new variable named "choice." It then declares a variable called "priceItems" and assigns it a value based on the value of "choice." If "choice" is equal to 1, "priceItems" is set to 12.9. If "choice" is 2, 4, 6, or 8, "priceItems" is set to 15.9. If "choice" is 5 or 7, "priceItems" is set to 18.9. For any other value of "choice," "priceItems" is set to 21.9. The program then prompts the user to enter the quantity of the chosen item and updates the "quantitychicken" and "burgerchicken" arrays accordingly. Finally, the function returns the value of "priceItems."

```

double burgerVegetarian(int& value)
{
    cout << "Vegetarian burger" << endl;
    cout << " " << endl;
    cout << "Item" << " " << "Price(RM)" << endl;
    cout << " " << " " << endl;
    cout << "1. Sunshine Bella" << " " << "14.90" << endl;
    cout << "2. Vegan Belia Bomb" << " " << "15.90" << endl;
    value = inputValidation("Enter Vegetarian Burger code (1-2): ", 2, 1, "vegetarian burger code");
    int choice = value;
    double priceItems;
    if (choice == 1)
    {
        priceItems = 14.9;
    }
    else
    {
        priceItems = 15.9;
    }

    value = inputValidationQ("Enter quantity");
    choice = choice - 1;
    quantityvege[choice] += value;
    burgervege[choice] += value;
    return priceItems;
}

```

Figure 31: Ala Carte Vegetarian Burger

In this C++ program, this program defines the function called burgerVegetarian that shown a display menu of the Ala Carte Vegetarian Burger with its corresponding prices.

```
double burgerVegetarian(int& value)
{
    cout << "Vegetarian burger" << endl;
    cout << " " << endl;
    cout << "Item" << "Price(RM)" << endl;
    cout << " " << endl;
    cout << "1. Sunshine Bella" << "14.90" << endl;
    cout << "2. Vegan Belia Bomb" << "15.90" << endl;
    value = inputValidation("Enter Vegetarian Burger code (1-2): ", 2, 1, "vegetarian burger code");
}
```

Figure 32: Ala Carte Vegetarian Burger

In this C++ program, the program displays a menu of vegetarian burger options and receives user input to select a specific vegetarian burger within the range of 1 to 2.

```
int choice = value;
double priceItems;
if (choice == 1)
{
    priceItems = 14.9;
}
else
{
    priceItems = 15.9;
}
value = inputValidationQ("Enter quantity");
choice = choice - 1;
quantityvege[choice] += value;
burgervege[choice] += value;
return priceItems;
}
```

Figure 33: Ala Carte Vegetarian Burger

In this C++ program, this program assigns the value of the variable "value" to a new variable called "choice." Then, a variable named "priceItems" is declared. If "choice" is equal to 1, the value of "priceItems" is set to 14.9. Otherwise, it is set to 15.9. The program then prompts the user to enter the quantity of the chosen item and updates the "quantityvege" and "burgervege" arrays accordingly. Finally, the function returns the value of "priceItems."

```
cout << "-----" << endl;
cout << "Adds - Ons\n";
cout << "1. Egg RM2.00\n";
cout << "2. Chicken Ham RM2.00\n";
cout << "3. Caramelized onion RM2.50\n";
cout << "4. Sliced cheese RM2.50\n";
cout << "5. Avocado RM3.00\n";
cout << "6. Bun RM3.00\n";
cout << "7. Blue cheese RM3.00\n";
cout << "8. Beef bacon RM3.50\n";
cout << "9. Bacon jam relish RM3.50\n";
cout << "10. Portobello mushroom RM4.00\n";
cout << "11. Grilled chicken RM6.00\n";
cout << "12. Fried chicken RM6.50\n";
cout << "13. Smashed beef RM7.00\n";
```

Figure 34: Adds-On choice

This code prints a list of the available add-ons and their prices.

```
int choiceAO[13] = { 0,0,0,0,0,0,0,0,0,0,0,0,0 };
int countAO = 0;
int choice{};
int value = inputValidation("Add-Ons for Burger " + to_string(count) + "?(1-Yes/2-No)", 2, 1, "choice");
```

Figure 35: Add-Ons input validation

This code prompts the user to select an add-on. The inputValidation() function ensures that the user enters a valid choice.

```

if (choice == 1 || choice == 2) {
    priceItems = 2.00;
}
else if (choice == 3 || choice == 4) {
    priceItems = 2.50;
}
else if (choice == 5 || choice == 6 || choice == 7) {
    priceItems = 3.00;
}
else if (choice == 8 || choice == 9) {
    priceItems = 3.50;
}
else if (choice == 10) {
    priceItems = 4.00;
}
else if (choice == 11) {
    priceItems = 6.00;
}
else if (choice == 12) {
    priceItems = 6.50;
}
else {
    priceItems = 7.00;
}

```

Figure 36: Add-On Prices

This code calculates the price of the add-on based on the user's selection.

```

totalpriceItems += priceItems;
choice = choice - 1;
quantityaddons[choice] += 1;
addons[choice] += 1;

```

Figure 37: Add-On Total Item Prices, Quantity and Add-On

This code updates the totalpriceItems, quantityaddons, and addons variables to reflect the addition of the add-on.

```

void setmeal(double& totalpriceItems, double priceItems) {
    maxsetmeal++;
    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Burger " << endl;
    cout << " " << endl;
    cout << "1. Beef " << endl;
    cout << "2. Chicken " << endl;
    cout << "3. Vegetarian " << endl;
    cout << " " << endl;
    int value = inputValidation("Insert burger code (1-3): ", 3, 1, "category burger");
    cout << "-----" << endl;
    cout << " " << endl;
    if (value == 1)
    {
        priceItems = setmealburgerBeef(priceItems);
    }
    else if (value == 2)
    {
        priceItems = setmealburgerChicken(priceItems);
    }
    else
    {
        priceItems = setmealburgerVege(priceItems);
    }
}

```

Figure 38: Set Meal Burger Choice

In this C++ program, the program defines a function called setmeal that allows users to select a set meal consisting of a burger, such as beef, chicken, or vegetarian.

```

void setmeal(double& totalpriceItems, double priceItems) {
    maxsetmeal++;
    cout << "-----" << endl;
    cout << "Burger " << endl;
    cout << "1. Beef " << endl;
    cout << "2. Chicken " << endl;
    cout << "3. Vegetarian " << endl;
    int value = inputValidation("Insert burger code (1-3): ", 3, 1, "category burger");
    cout << "-----" << endl;
    cout << "

```

Figure 39: Set Meal Burger Choice

In this C++ program, the program presents a menu for burger choices, allowing the user to select between different burger types (beef, chicken, or vegetarian) by entering a corresponding code. The input validation ensures that the code entered falls within the acceptable range.

```

if (value == 1)
{
    priceItems = setmealburgerBeef(priceItems);
}
else if (value == 2)
{
    priceItems = setmealburgerChicken(priceItems);
}
else
{
    priceItems = setmealburgerVege(priceItems);
}

```

Figure 40: Set Meal Burger Choice

In this C++ program, the program checks the value of a variable called "value." If the value is 1, it performs an action related to a beef burger set meal, updating the priceItems variable. If the value is 2, it performs an action related to a chicken burger set meal, again updating the priceItems variable. Finally, if the value is neither 1 nor 2, it performs an action related to a vegetarian burger set meal,

```

double setmealburgerChicken(double priceItems)
{
    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Chicken burger" << endl;
    cout << "" << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Tornado" << endl;
    cout << "3. Bella Bomb" << endl;
    cout << "4. Bobcat" << endl;
    cout << "5. Klub" << endl;
    cout << "6. Seoul" << endl;
    cout << "7. Hot Mess" << endl;
    cout << "8. Kickin" << endl;
    int value = inputValidation("Enter Chicken burger code (1-8): ", 8, 1, "chicken burger code");
    if (value == 1)
    {
        priceItems = 12.9;
        setmealB[maxsetmeal] = "The Shack [Chicken]";
    }
    else if (value == 2)
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Tornado [Chicken]";
    }
    else if (value == 3)
    {
        priceItems = 21.9;
        setmealB[maxsetmeal] = "Bella Bomb";
    }
    else if (value == 4)
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Bobcat [Chicken]";
    }
    else if (value == 5)
    {
        priceItems = 18.9;
        setmealB[maxsetmeal] = "Klub";
    }
    else if (value == 6)
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Seoul";
    }
    else if (value == 7)
    {
        priceItems = 18.9;
        setmealB[maxsetmeal] = "Hot Mess";
    }
    else
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Kickin'";
    }
    value = value - 1;
    burgerchicken[value] += 1;
    return priceItems;
}

```

Figure 41: Set Meal Burger Chicken

In this C++ program, the program defines a function called setmealburgerChicken that displays a menu of set meal burger chicken with corresponding prices.

```

double setmealburgerChicken(double priceItems)
{
    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Chicken burger" << endl;
    cout << "" << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Tornado" << endl;
    cout << "3. Bella Bomb" << endl;
    cout << "4. Bobcat" << endl;
    cout << "5. Klub" << endl;
    cout << "6. Seoul" << endl;
    cout << "7. Hot Mess" << endl;
    cout << "8. Kickin" << endl;
    int value = inputValidation("Enter Chicken burger code (1-8): ", 8, 1, "chicken burger code");
}

```

Figure 42: Set Meal Burger Chicken

In this C++ program, the program displaying a menu of chicken burger options and receiving user input to select a specific burger within the range of 1 to 8.

```

if (value == 1)
{
    priceItems = 12.9;
    setmealB[maxsetmeal] = "The Shack [Chicken]";
}
else if (value == 2)
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Tornado [Chicken]";
}
else if (value == 3)
{
    priceItems = 21.9;
    setmealB[maxsetmeal] = "Bella Bomb";
}
else if (value == 4)
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Bobcat [Chicken]";
}
else if (value == 5)
{
    priceItems = 18.9;
    setmealB[maxsetmeal] = "Klub";
}
else if (value == 6)
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Seoul";
}
else if (value == 7)
{
    priceItems = 18.9;
    setmealB[maxsetmeal] = "Hot Mess";
}
else
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Kickin'";
}
value = value - 1;
burgerchicken[value] += 1;
return priceItems;
}

```

Figure 43: Set Meal Burger Chicken

In this C++ program, the program checks the value of a variable called "value". Each value corresponds to a specific chicken burger option. The program, updates the priceItems variable with the price of the selected chicken burger and stores its name in an array called setmealB. It also keeps track of how many times each chicken burger option has been selected by incrementing the corresponding counter in the burgerchicken array. Finally, the updated price of the selected chicken burger is returned by the function.

```

double setmealburgerBeef(double priceItems) {
    cout << " " << endl;
    cout << "Beef burger" << endl;
    cout << " " << endl;
    cout << "Item" << endl;
    cout << " " << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Animal Style" << endl;
    cout << "3. Double Animal" << endl;
    cout << "4. HGB Spiked" << endl;
    cout << "5. Tornado" << endl;
    cout << "6. Bash" << endl;
    cout << "7. Bella Bomb Tower" << endl;
    cout << "8. Sacre Bleu!" << endl;
    cout << "9. Truffle & Swish Mushroom" << endl;
    cout << "10. Bobcat" << endl;

    Price(RM) << endl;
    12.90* << endl;
    14.90* << endl;
    19.90* << endl;
    15.90* << endl;
    15.90* << endl;
    18.90* << endl;
    21.90* << endl;
    18.90* << endl;
    20.90* << endl;
    15.90* << endl;

    int value = inputValidation("Enter Beef burger code (1-10): ", 10, 1, "beef burger code");
    if (value == 1)
    {
        priceItems = 12.9;
        setmealB[maxsetmeal] = "The Shack [Beef]";
    }
    else if (value == 2)
    {
        priceItems = 14.9;
        setmealB[maxsetmeal] = "Animal Style";
    }
    else if (value == 3)
    {
        priceItems = 19.9;
        setmealB[maxsetmeal] = "Double Animal";
    }
    else if (value == 4)
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "HGB Spiked";
    }
    else if (value == 5)
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Tornado [beef]";
    }
    else if (value == 6)
    {
        priceItems = 18.9;
        setmealB[maxsetmeal] = "Bash";
    }
    else if (value == 7)
    {
        priceItems = 21.9;
        setmealB[maxsetmeal] = "Bella Bomb Tower";
    }
    else if (value == 8)
    {
        priceItems = 18.9;
        setmealB[maxsetmeal] = "Sacre Bleu!";
    }
    else if (value == 9)
    {
        priceItems = 20.9;
        setmealB[maxsetmeal] = "Truffle and Swish Mushroom";
    }
    else
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Bobcat [Beef]";
    }
    value = value - 1;
    burgerbeef[value] += 1;
    return priceItems;
}

```

Figure 44: Set Meal Burger Beef

The provided C++ code defines a function called `setmealburgerBeef` that displays a menu of set meal burger beef with corresponding prices.

```
double setmealburgerBeef(double priceItems) {
    cout << "-----" << endl;
    cout << "Beef burger" << endl;
    cout << "Item" << endl;
    cout << "Price(RM)" << endl;
    cout << "1. The Shack" << endl;
    cout << "2. Animal Style" << endl;
    cout << "3. Double Animal" << endl;
    cout << "4. KGB Spiked" << endl;
    cout << "5. Tornado" << endl;
    cout << "6. Bash" << endl;
    cout << "7. Bella Bomb Tower" << endl;
    cout << "8. Sacre Bleu!" << endl;
    cout << "9. Truffle & Swish Mushroom" << endl;
    cout << "10. Bobcat" << endl;
    int value = inputValidation("Enter Beef burger code (1-10): ", 10, 1, "beef burger code");
}
```

Figure 45: Set Meal Burger Beef

In this C++ program, the program displaying a menu of beef burger options and receiving user input to select a specific burger within the range of 1 to 10.

```
if (value == 1)
{
    priceItems = 12.9;
    setmealB[maxsetmeal] = "The Shack [Beef]";
}
else if (value == 2)
{
    priceItems = 14.9;
    setmealB[maxsetmeal] = "Animal Style";
}
else if (value == 3)
{
    priceItems = 19.9;
    setmealB[maxsetmeal] = "Double Animal";
}
else if (value == 4)
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "KGB Spiked";
}
else if (value == 5)
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Tornado [beef]";
}
else if (value == 6)
{
    priceItems = 18.9;
    setmealB[maxsetmeal] = "Bash";
}
else if (value == 7)
{
    priceItems = 21.9;
    setmealB[maxsetmeal] = "Bella Bomb Tower";
}
else if (value == 8)
{
    priceItems = 18.9;
    setmealB[maxsetmeal] = "Sacre Bleu!";
}
else if (value == 9)
{
    priceItems = 20.9;
    setmealB[maxsetmeal] = "Truffle and Swish Mushroom";
}
else
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Bobcat [Beef]";
}
value = value - 1;
burgerbeef[value] += 1;
return priceItems;
}
```

Figure 46: Set Meal Burger Beef

In this C++ program, the program update the priceItems variable and the setmealB and burgerbeef arrays based on the value variable. The program checks the value against different numbers to determine which beef burger option has been selected. For each value, the code sets the priceItems variable to the corresponding price and stores the name of the selected burger in the setmealB array. It also increments the burgerbeef array at the corresponding index to keep track of the number of times each beef burger option has been chosen. Finally, the value is decreased by 1, and the updated priceItems value is returned by the function.


```

double setmealburgerVege(double priceItems)
{
    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Vegetarian Burger" << endl;
    cout << " " << endl;
    cout << "Item" << "Price(RM)" << endl;
    cout << " " << endl;
    cout << "1. Sunshine Bella" << "14.90" << endl;
    cout << "2. Vegan Belia Bomb" << "15.90" << endl;
    int value = inputValidation("Enter Vegetarian Burger code (1-2): ", 2, 1, "vegetarian burger code");
    if (value == 1)
    {
        priceItems = 14.9;
        setmealB[maxsetmeal] = "Sunshine Bella";
    }
    else
    {
        priceItems = 15.9;
        setmealB[maxsetmeal] = "Vegan Belia Bomb";
    }
    value = value - 1;
    burgervege[value] += 1;
    return priceItems;
}

```

Figure 47: Set Meal Burger Vegetarian

The provided C++ code defines a function called `setmealburgerVege` that displays a menu of set meal burger beef with corresponding prices.

```

double setmealburgerVege(double priceItems)
{
    cout << "-----" << endl;
    cout << " " << endl;
    cout << "Vegetarian Burger" << endl;
    cout << " " << endl;
    cout << "Item" << "Price(RM)" << endl;
    cout << " " << endl;
    cout << "1. Sunshine Bella" << "14.90" << endl;
    cout << "2. Vegan Belia Bomb" << "15.90" << endl;
    int value = inputValidation("Enter Vegetarian Burger code (1-2): ", 2, 1, "vegetarian burger code");
}

```

Figure 48: Set Meal Burger Vegetarian

In this C++ program, the program displaying a menu of vegetarian burger options and receiving user input to select a specific burger within the range of 1 to 2.

```

if (value == 1)
{
    priceItems = 14.9;
    setmealB[maxsetmeal] = "Sunshine Bella";
}
else
{
    priceItems = 15.9;
    setmealB[maxsetmeal] = "Vegan Belia Bomb";
}
value = value - 1;
burgervege[value] += 1;
return priceItems;
}

```

Figure 49: Set Meal Burger Vegetarian

In this C++ program, the program checks the value variable. If the value is equal to 1, the code sets the `priceItems` variable to 14.9 and assigns the name "Sunshine Bella" to the `setmealB` array. Otherwise, if the value is not equal to 1, the code sets the `priceItems` variable to 15.9 and assigns the name "Vegan Belia Bomb" to the `setmealB` array. Then, the program subtracts 1 from the value, increments the `burgervege` array at the corresponding index, and returns the updated `priceItems` value.

```

value = value - 1;
subCategoryBurger[value] += 1;
setmealprice[maxsetmeal] = priceItems + 5.90;
totalpriceItems += priceItems + 5.90;
cout << "-----" << endl;
cout << " " << endl;
cout << "Set Meal Fries " << endl;
cout << " " << endl;
cout << "1. Salt & Pepper Fries " << endl;
cout << "2. Herbed Fries " << endl;
cout << "3. Spicy Cajun Fries " << endl;
cout << "4. Malt Vinegar Fries " << endl;
value = inputValidation("Enter fries code (1-4): ", 4, 1, "fries code");
if (value == 1)
{
    setmealFries[maxsetmeal] = "Salt & Pepper Fries";
}
else if (value == 2)
{
    setmealFries[maxsetmeal] = "Herbed Fries";
}
else if (value == 3)
{
    setmealFries[maxsetmeal] = "Spicy Cajun Fries";
}
else
{
    setmealFries[maxsetmeal] = "Malt Vinegar Fries";
}
value = value - 1;
setmealF[value] += 1;
cout << "-----" << endl;
cout << " " << endl;
cout << "Set Meal Beverages " << endl;
cout << " " << endl;
cout << "1. Lemonade " << endl;
cout << "2. Ice Mango Tea " << endl;
cout << "3. Honey Lemon " << endl;
cout << "4. Ice Lemon Tea " << endl;
value = inputValidation("Enter beverage code (1-4): ", 4, 1, "beverage code");
if (value == 1)
{
    setmealBeverages[maxsetmeal] = "Lemonade";
}
else if (value == 2)
{
    setmealBeverages[maxsetmeal] = "Ice Mango Tea";
}
else if (value == 3)
{
    setmealBeverages[maxsetmeal] = "Honey Lemon";
}
else
{
    setmealBeverages[maxsetmeal] = "Ice Lemon Tea";
}
value = value - 1;
setmealBev[value] += 1;
}

```

Figure 50: Set Meal Fries & Beverages

In this C++ program, the program allows the user to select fries and beverages for the set meal.

```

value = value - 1;
subCategoryBurger[value] += 1;
setmealprice[maxsetmeal] = priceItems + 5.90;
totalpriceItems += priceItems + 5.90;
cout << "-----" << endl;
cout << " " << endl;
cout << "Set Meal Fries " << endl;
cout << " " << endl;
cout << "1. Salt & Pepper Fries " << endl;
cout << "2. Herbed Fries " << endl;
cout << "3. Spicy Cajun Fries " << endl;
cout << "4. Malt Vinegar Fries " << endl;
value = inputValidation("Enter fries code (1-4): ", 4, 1, "fries code");

```

Figure 51: Set Meal Fries

In this C++ program, the program displaying a menu of set meal fries options and receiving user input to select a specific fries within the range of 1 to 4.

```

if (value == 1)
{
    setmealFries[maxsetmeal] = "Salt & Pepper Fries";
}
else if (value == 2)
{
    setmealFries[maxsetmeal] = "Herbed Fries";
}
else if (value == 3)
{
    setmealFries[maxsetmeal] = "Spicy Cajun Fries";
}
else
{
    setmealFries[maxsetmeal] = "Malt Vinegar Fries";
}

```

Figure 52: Set Meal Fries

In this C++ program, the program update the setmealFries array based on the value variable. The program checks the value against different numbers to determine which type of fries option has been selected. If the value is equal to 1, the code assigns the name "Salt & Pepper Fries" to the setmealFries array. If the value is 2, the code assigns the name "Herbed Fries". If the value is 3, the code assigns the name "Spicy Cajun Fries". Otherwise, if the value is not equal to 1, 2, or 3, the code assigns the name "Malt Vinegar Fries".

```

value = value - 1;
setmealF[value] += 1;
cout << "-----" << endl;
cout << " " << endl;
cout << "Set Meal Beverages " << endl;
cout << " " << endl;
cout << "1. Lemonade " << endl;
cout << "2. Ice Mango Tea " << endl;
cout << "3. Honey Lemon " << endl;
cout << "4. Ice Lemon Tea " << endl;
value = inputValidation("Enter beverage code (1-4): ", 4, 1, "beverage code");

```

Figure 53: Set Meal Beverages

In this C++ program, the program displaying a menu of set meal beverages options and receiving user input to select a specific beverages within the range of 1 to 4.

```

if (value == 1)
{
    setmealBeverages[maxsetmeal] = "Lemonade";
}
else if (value == 2)
{
    setmealBeverages[maxsetmeal] = "Ice Mango Tea";
}
else if (value == 3)
{
    setmealBeverages[maxsetmeal] = "Honey Lemon";
}
else
{
    setmealBeverages[maxsetmeal] = "Ice Lemon Tea";
}
value = value - 1;
setmealBev[value] += 1;
}

```

Figure 54: Set Meal Beverages

In this C++ program, this program update the setmealBeverages array based on the value variable. The code checks the value against different numbers to determine which type of beverage option has been selected. If the value is equal to 1, the code assigns the name "Lemonade" to the setmealBeverages array. If the value is 2, the code assigns the name "Ice Mango Tea". If the value is 3, the code assigns the name "Honey Lemon". Otherwise, if the value is not equal to 1, 2, or 3, the code assigns the name "Ice Lemon Tea". After updating the setmealBeverages array, the code decreases the value by 1 and increments the setmealBev array at the corresponding index to track the number of times each beverage option has been selected.

```

void receiptTemplate(double currenttotalPrice, double hour, double minute, string time, int month, int year, string staff, double totalpriceitems, int day, double payment)
{
    cout << "\n\n ";
    cout << "
                                KGB" << endl;
    cout << "                                KGB Holding Sdh Bhd" << endl;
    cout << "                                1892679-U" << endl;
    cout << "                                The Courtyard by KGB" << endl;
    cout << "                                GP1-03 Level 3 , Tamarind Square," << endl;
    cout << "                                Persiaran Multimedia , Cyberjaya , " << endl;
    cout << "                                63000,Selangor" << endl;
    cout << "-----" << endl;
    cout << "                                INVOICE" << endl;
    cout << "-----" << endl;
    cout << "Date          : " << day << "/" << month << "/" << year << " " << hour << ":" << minute << " " << time << endl;
    cout << "Staff: " << staff << endl;
    cout << "Order ID: " << rand() << endl;
    cout << "-----" << endl;
    cout << "Item                                Qty          Unit P. (RM)          Price(RM)" << endl;
    cout << "-----" << endl;
}

```

Figure 55: Receipt Template

In this C++ program, the program displays the header section of the receipt, including the company and location details, invoice title, date and time, staff information, order ID, and a table header for the item details.

```

for (int count = 1; count <= maxsetmeal; count++)
{
    cout << "Set Meal" << endl;
    cout << setmealB[count] + "                                1                                " << setmealprice[count] << endl;
    cout << setmealFries[count] << endl;
    cout << setmealBeverages[count] << endl;
    cout << "                                " << endl;
}

```

Figure 56: Receipt Template

In this C++ program, the program displays the details of various set meals in a receipt. It goes through each set meal and displays its name, quantity (which is always 1), unit price, and additional details like fries and beverages

```
receiptCalculate(10, burgerbeefN, priceItemsBeef, quantitybeef);
receiptCalculate(8, burgerchickenN, priceItemsChicken, quantitychicken);
receiptCalculate(2, burgervegeN, priceItemsVege, quantityvege);
receiptCalculate(13, addonsN, priceItemsAddons, quantityaddons);
receiptCalculate(11, sidesItemsN, priceItemsSides, quantitysides);
receiptCalculate(6, drinksN, priceItemsDrinks, quantitydrinks0);
receiptCalculate(3, drinksNM, priceItemsDrinks, quantitydrinksM);
```

Figure 57: Receipt Template

In this C++ program, these lines of code are calling the receiptCalculate function multiple times with different arguments. Each function call represents a specific item category and provides the necessary parameters to calculate and display the details of those items in the receipt.

```
cout << fixed << setprecision(2) << "-----" << endl;
cout << "Subtotal" << " " << totalpriceitems << endl;
cout << "6% Service Tax" << " " << totalpriceitems * 0.06 << endl;
cout << "Total" << " " << currenttotalPrice << endl;
cout << "CASH" << " " << payment << endl;
cout << "Change" << " " << payment - currenttotalPrice << endl;
```

Figure 58: Receipt Template

In this C++ program, the program displays the subtotal, service tax, total amount, cash payment, and change in the receipt format.

```
void receiptCalculate(int maxValue, string items[], double price[], int quantity[])
{
    int count = 0;
    while (count < maxValue)
    {
        if (quantity[count] == 0)
        {
        }
        else
        {
            cout << items[count] << " " << quantity[count] << " " << price[count] << " " << quantity[count] * price[count] << endl;
            count++;
        }
    }
}
```

Figure 59: Receipt Calculate

The provided C++ code defines a function called `receiptCalculate` to calculate and display the details of specific items in a receipt.

```

void rankingSummarys(int maxValue, int itemsQ[], string itemsN[], string itemsRanking)
{
    int outer = 0;
    while (outer < maxValue)
    {
        int inner = outer + 1;
        while (inner < maxValue)
        {
            if (itemsQ[inner] > itemsQ[outer])
            {
                int temporary = itemsQ[inner];
                itemsQ[inner] = itemsQ[outer];
                itemsQ[outer] = temporary;
                string temporaryN = itemsN[inner];
                itemsN[inner] = itemsN[outer];
                itemsN[outer] = temporaryN;
            }
            else
            {
                inner++;
            }
        }
        outer++;
    }
}

```

Figure 60: Ranking Summary

In this C++ program, the code compares quantities and swaps items until they are arranged from the highest quantity to the lowest. This helps create a ranking based on the quantities of the items.

```

int index = 1;
cout << " " << endl;
cout << itemsRanking << endl;
cout << "-----" << endl;
for (int index = 0; index < maxValue; index++)
{
    cout << itemsN[index] << " " << itemsQ[index] << endl;
}

```

Figure 61: Ranking Summary

In this C++ program, the program displays the ranked items and their corresponding quantities in a neat and organized format. It begins with a heading, followed by a list of items and their quantities, clearly summarising the ranking.

```

void summaryValue(int maxValue, int itemsQ[], string itemsN[], string itemsRanking)
{
    cout << " " << endl;
    cout << itemsRanking << endl;
    cout << "-----" << endl;
    for (int index = 0; index < maxValue; index++)
    {
        cout << itemsN[index] << " " << itemsQ[index] << endl;
    }
}

```

Figure 62: Function summaryValue

In this C++ program, the program gets maxValue, itemsQ[], itemsN[] and itemsRanking as parameters. The program then displays the output using for loop and maxValue as condition.

4.0 SAMPLE OUTPUT

```
MAIN MENU
1. Receive order from the customer
2. Generate reports
3. Exit the program

Enter Choice
```

Figure 63: Main Menu Output

The C++ output display Main Menu and 3 choices. It asks users to choose between 3 choices.

```
GENERATE REPORTS

1. Quantity sold for each product for the day
2. Total sales for the day
3. Ranking of the most ordered product for the day
4. Return to Main Menu

Enter Choice
|
```

Figure 64: Main Menu Output

The C++ output display submenu for main menu choice 2 and display 4 choices. It asks users to choose between 4 choices.

```
Category

1. Food
2. Drinks

Insert category code (1-2):
```

Figure 65: Category Choice Output

The C++ output display category. Ask users to input the category code from 1 to 2.

```
Food
1. Burger
2. Sides
```

Figure 66: Category Food Choice Output

The C++ output display food category. Ask users to input the food category code from 1 to 2.

```
Category
1. Food
2. Drinks
Insert category code (1-2):
2
-----
Drinks
1. Others
2. Milkshakes
```

Figure 67: Category Drinks Choice Output

The C++ output display category and drinks. Ask users to input the category code from 1 to 2.

```
Other drinks
Item                                Price(RM)
1. Lemonade                        5.90
2. Ice mango tea                    5.90
3. Honey lemon                     5.90
4. Ice lemon tea                    5.90
5. Ice lavender tea                 6.90
6. Stawberry lemonade               6.90
7. Root beer float                  6.90
8. Coke float                       6.90
Enter the drink's code(1-8):
```

Figure 68: Others Drinks Output

The C++ output display other drinks. Ask users to input the other drinks code from 1 to 8.

```
Milkshakes
Item                                Price(RM)
1. Nutella S' mores                 11.90
2. Stawberry season                  11.90
3. Salted Caramel Pralines           11.90
Enter milkshake code(1-3):
```

Figure 69: Milkshakes Output

The C++ output display milkshake. Ask users to input the milkshake code from 1 to 3.


```

Make it a set meal? (1-Yes, 2-No)
2
-----

Burger

1. Beef
2. Chicken
3. Vegetarian

Insert burger code(1-3):

```

Figure 70: Ala Carte Burger Choice Output

This C++ shown the output for all of the Burger Set provided and asks the user which burger's set and type of burger either Beef, Chicken or Vegetarian Burger they want.

```

Beef burger

Item                                                    Price(RM)

1. The Shack                                           12.90
2. Animal Style                                       14.90
3. Double Animal                                       19.90
4. KGB Spiked                                         15.90
5. Tornado                                            15.90
6. Bash                                               18.90
7. Bella Bomb Tower                                   21.90
8. Sacre Bleu!                                       18.90
9. Truffle & Swish Mushroom                           20.90
10. Bobcat                                            15.90

Enter Beef burger code (1-10):

```

Figure 71: Ala Carte Burger Beef

This C++ output shown all of the Ala Carte Beef Burger Set and asks users to input the beef burger code from 1 to 10 of their chosen menus.

```

Chicken burger

Item                                                    Price(RM)

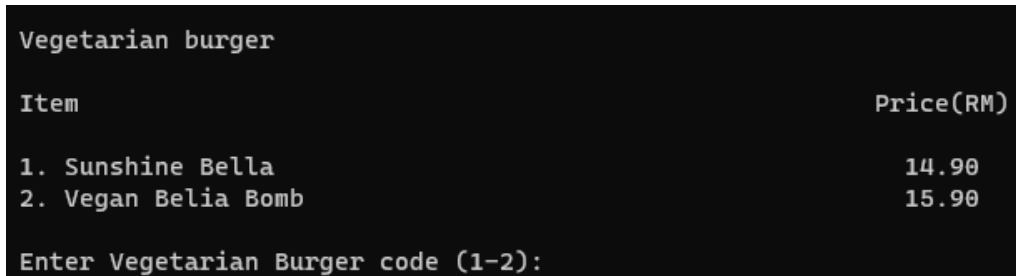
1. The Shack                                           12.90
2. Tornado                                            15.90
3. Bella Bomb                                         21.90
4. Bobcat                                             15.90
5. Klub                                               18.90
6. Seoul                                              15.90
7. Hot Mess                                           18.90
8. Kickin                                             15.90

Enter Chicken burger code (1-8):

```

Figure 72: Ala Carte Chicken Beef

This C++ output shown all of the Ala Carte Chicken Burger Set and asks users to input the chicken burger code from 1 to 8 of their chosen menus.



```
Vegetarian burger

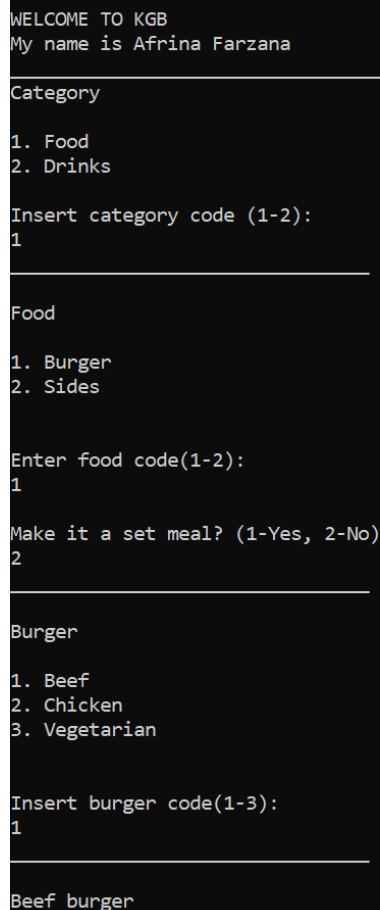
Item                                     Price(RM)

1. Sunshine Bella                       14.90
2. Vegan Belia Bomb                     15.90

Enter Vegetarian Burger code (1-2):
```

Figure 73: Ala Carte Burger Vegetarian

This C++ output shown all of the Ala Carte Vegetarian Burger Set and asks users to input the vegetarian burger code either 1 or 2 of their chosen menus.



```
WELCOME TO KGB
My name is Afrina Farzana

Category
1. Food
2. Drinks

Insert category code (1-2):
1

Food
1. Burger
2. Sides

Enter food code(1-2):
1

Make it a set meal? (1-Yes, 2-No)
2

Burger
1. Beef
2. Chicken
3. Vegetarian

Insert burger code(1-3):
1

Beef burger
```

Figure 74: Adds- On Output

When Users ordered Burger, the system will them to make it as a set meal or ala- carte.

```

Add-Ons to burger?(Yes - 1/No - 2)
1
-----
Adds - Ons
1. Egg RM2.00
2. Chicken Ham RM2.00
3. Caramelized onion RM2.50
4. Sliced cheese RM2.50
5. Avocado RM3.00
6. Bun RM3.00
7. Blue cheese RM3.00
8. Beef bacon RM3.50
9. Bacon jam relish RM3.50
10. Portobello mushroom RM4.00
11. Grilled chicken RM6.00
12. Fried chicken RM6.50
13. Smashed beef RM7.00

Add-Ons for Burger 1?(1-Yes/2-No)
3
Invalid choice.Please enter 1 to 2 only.
Add-Ons for Burger 1?(1-Yes/2-No)
1

Enter Add-Ons code between 1-13
13

Add more add-ons for burger 1? (1-Yes/2-No)
1

Enter Add-Ons code between 1-13
9

Add more add-ons for burger 1? (1-Yes/2-No)
2

```

Figure 75: Adds- On Output

The user can add the add-on as many as they want to. The user just needs to enter from 1 to 13 and input the item they would like to order.

```

Make it a set meal? (1-Yes, 2-No)
1
-----

Burger

1. Beef
2. Chicken
3. Vegetarian

Insert burger code (1-3):
|

```

Figure 76: Set Meal Burger Choice Output

This C++ output displays all the set meal burger and asks users to make it to set meal or not and which set meal they want.

```

Insert burger code (1-3):
1
-----
-----

Beef burger

Item                                                    Price(RM)
1. The Shack                                           12.90
2. Animal Style                                        14.90
3. Double Animal                                       19.90
4. KGB Spiked                                          15.90
5. Tornado                                             15.90
6. Bash                                                18.90
7. Bella Bomb Tower                                    21.90
8. Sacre Bleu!                                         18.90
9. Truffle & Swish Mushroom                           20.90
10. Bobcat                                             15.90

Enter Beef burger code (1-10):
|

```

Figure 77: Set Meal Burger Beef Output

This C++ output displays all the set meal burger beef and asks users to input the beef burger code from 1 to 10.

```

Chicken burger

Item                                                    Price(RM)
1. The Shack                                           12.90
2. Tornado                                             15.90
3. Bella Bomb                                          21.90
4. Bobcat                                              15.90
5. Klub                                                18.90
6. Seoul                                               15.90
7. Hot Mess                                            18.90
8. Kickin                                              15.90

Enter Chicken burger code (1-8):
|

```

Figure 78: Set Meal Burger Chicken Output

This C++ output displays all the set meal burger chicken and asks users to input the chicken burger code from 1 to 8.

```
Vegetarian Burger

Item                                                    Price(RM)
1. Sunshine Bella                                     14.90
2. Vegan Belia Bomb                                   15.90

Enter Vegetarian Burger code (1-2):
|
```

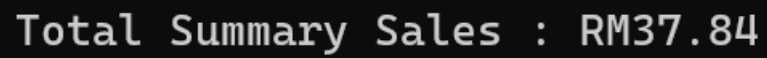
Figure 79: Set Meal Burger Vegetarian Output

This C++ output displays all the set meal burger vegetarian and asks users to input the vegetarian burger code from 1 to 2.

SUMMARY	
CATEGORY	
Food	2
Drinks	4
FOOD	
Burger	1
Sides	8
DRINKS	
Others	8
Milkshakes	3
BURGER	
Burger Beef	2
Burger Chicken	8
Burger Vegetarian	8
BEEF BURGER	
The Shack [Beef]	8
Animal Style	8
Double Animal	2
WGB Spiked	8
Tornado [Beef]	8
Bash	8
Bella Bomb Tower	8
Sacre Bleu!	8
Truffle & Swiss Mushroom	8
Bobcat [Beef]	8
CHICKEN BURGER	
The Shack [Chicken]	8
Tornado [Chicken]	8
Bella Bomb	8
Bobcat [Chicken]	8
Klub	8
Seoul	8
Hot Mess	8
Kickin'	8
VEGETARIAN BURGER	
Sunshine Bella	8
Vegan Bella Bomb	8
ADD-ONS	
Egg	1
Chicken Ham	8
Caramelized Onion	8
Sliced Cheese	1
Avocado	2
Bun	12
Blue Cheese	8
Beef Bacon	1
Bacon Jam Relish	1
Portobello Mushroom	1
Grilled Chicken	8
Fried Chicken	1
Smashed Beef	8
SET MEAL FRIES	
Salt & Pepper Fries	8
Herbed Fries	8
Spicy Cajun Fries	8
Rait Vinegar Fries	8
SET MEAL BEVERAGES	
Lenonade	8
Ice Mango Tea	8
Honey Lemon	8
Ice Lemon Tea	8
SIDES	
Onion Rings (Buttervilk Fried)	8
Chicken Bites with Dips	8
Potato Wedges with Dips	8
Salt & Pepper Fries	8
Herbed Fries	8
Spicy Cajun Fries	8
Rait Vinegar Fries	8
Truffle'd Fries	8
Kinchi Fries with Bulgogi Beef	8
Kinchi Fries Fully Loaded	12
Texas Jalapeno Cheese Fries	8
OTHER DRINKS	
Lenonade	8
Ice Mango Tea	5
Honey Lemon	8
Ice Lemon Tea	8
Ice Lavender Tea	2
Strawberry Lenonade	8
Root Beer Float	3
Coke Float	9
MILKSHAKES	
Mutella S'mores	8
Strawberry Season	5
Salted Caramel Pralines	8

Figure 80: Summary

This C++ output displays the summary of different food items and drinks. It displays the quantities of items like burgers, sides, beverages, and milkshakes have been ordered in that particular day.



```
Total Summary Sales : RM37.84
```

Figure 81: Total Sales For The Day

This C++ output displays total sales for that particular day.

RANKING	
CATEGORY	
Drinks	4
Food	2
FOOD	
Burger	1
Sides	6
DRINKS	
Milshakes	3
Others	6
BURGER	
Burger Beef	2
Burger Chicken	6
Burger Vegetarian	6
BEEF BURGER	
Double Animal	2
Animal Style	6
The Shack [Beef]	6
WGB Spiked	6
Tornado [Beef]	6
Bash	6
Bella Bomb Tower	6
Sacra Blau!	6
Truffle & Swiss Mushroom	6
Bobcat [Beef]	6
CHICKEN BURGER	
The Shack [Chicken]	6
Tornado [Chicken]	6
Bella Bomb	6
Bobcat [Chicken]	6
WLab	6
Scowl	6
Hot Mess	6
Kickin'	6
VEGETARIAN BURGER	
Sunshine Bella	6
Vegan Bella Bomb	6
ADD-ONS	
Bun	12
Avocado	2
Egg	1
Sliced Cheese	1
Beef Bacon	1
Bacon Jam Relish	1
Portobello Mushroom	1
Fried Chicken	1
Chicken Ham	6
Blue Cheese	6
Grilled Chicken	6
Caramelized Onion	6
Smashed Beef	6
SET MEAL FRIES	
Salt & Pepper Fries	6
Herbed Fries	6
Spicy Cajun Fries	6
Malt Vinegar Fries	6
SET MEAL BEVERAGES	
Leonade	6
Ice Mango Tea	6
Honey Lemon	6
Ice Lemon Tea	6
SIDES	
Kinchi Fries Fully Loaded	12
Chicken Bites with Dips	6
Potato Wedges with Dips	6
Salt & Pepper Fries	6
Herbed Fries	6
Spicy Cajun Fries	6
Malt Vinegar Fries	6
Truffle'd Fries	6
Kinchi Fries with Bulgogi Beef	6
Onion Rings (Buttermilk Fried)	6
Texas Jalapeno Cheese Fries	6
OTHER DRINKS	
Coke Float	9
Ice Mango Tea	5
Root Beer Float	3
Ice Lavender Tea	2
Leonade	6
Strawberry Leonade	6
Honey Lemon	6
Ice Lemon Tea	6
MILKSHAKES	
Nutella S'mores	6
Strawberry Season	6
Salted Caramel Pralines	6

Figure 82: Ranking

This C++ output displays information that helps identify the popularity or demand for specific items. The ranking gives a quick overview of the most popular items based on their quantities, providing insights into customer preferences and ordering patterns.

5.0 REFERENCES

- KGB. (2023, March 6). *KGB - Killer Gourmet Burgers, Best Burger in Town* | *kgb.my*. <https://kgb.my/>

6.0 RUBRIC

Product Rubric

CLO2: Construct computer programs using suitable compiler or IDE. 15% contribution to the continuous assessment.

Criteria	Psyc. Level	Poor 0	Unsatisfactory 1	Satisfactory 2	Good 3	Excellent 4	Score
Input; prompt and input statement method.	P3	Unable to accomplish.	<ul style="list-style-type: none"> Unclear prompt. Unsuitable input method. 	<ul style="list-style-type: none"> Clear prompt. Suitable input method. 	<ul style="list-style-type: none"> Unclear prompt. Suitable input method. 	<ul style="list-style-type: none"> Clear prompt. Suitable input method. 	
Input; control structure for validation.	P4	No input validation.	None are suitable.	Some are suitable.	Most are suitable.	All are suitable.	
Process; assignment statements or computations.	P4	Unable to accomplish.	None are correct.	Some are correct.	Most are correct.	All are correct.	
Output; in the form of a list or a table.	P3	Unable to accomplish.	<ul style="list-style-type: none"> Improper form. Incorrect value. Improper value format. 	<ul style="list-style-type: none"> Proper form. Incorrect value. Improper value format. 	<ul style="list-style-type: none"> Proper form. Correct value. Improper value format. 	<ul style="list-style-type: none"> Proper form. Correct value. Proper value format. 	
Sequence; the flow of ordering.	P3	Unable to accomplish.	Altogether different from the company's flow.	Mostly different from the company's flow.	Mostly similar to the company's flow.	Similar to the company's flow.	
Selection; type and condition.	P4	Not used.	None are suitable.	Some are suitable.	Most are suitable.	All are suitable.	
Repetition; type and elements (initialisation or priming read, condition, and update).	P4	Not used.	None are suitable.	Some are suitable.	Most are suitable.	All are suitable.	

Functions excluding main; prototype, definition, and call.	P5	Not used.	None are efficient.	Some are efficient.	Most are efficient.	All are efficient.	
Array; dimension, size, and traversal.	P4	Not used.	<ul style="list-style-type: none"> • Minimal usage. • Inefficient traversal. 	<ul style="list-style-type: none"> • Minimal usage. • Efficient traversal. 	<ul style="list-style-type: none"> • Optimal usage. • Inefficient traversal. 	<ul style="list-style-type: none"> • Optimal usage. • Efficient traversal. 	
Total Score							
Percentage = (Total Score ÷ 36 marks) × 15							

Report Rubric

CLO1: Program computer programs using principles of programming and syntax of chosen programming language.10% contribution to the continuous assessment.

Criteria	Cog. Level	Poor 0	Unsatisfactory 1	Satisfactory 2	Good 3	Excellent 4	Score
Introduction; company profile and products.	C2	Missing profile and product.	Missing profile or product.	Incomprehensive profile and product.	Comprehensive profile or product.	Comprehensive profile and products.	
Program Code; complete code with a brief explanation.	C4	Missing code and explanation.	<ul style="list-style-type: none"> Partial code. Missing explanation. 	<ul style="list-style-type: none"> Complete code. Missing explanation. 	<ul style="list-style-type: none"> Complete code. Bad explanation. 	<ul style="list-style-type: none"> Complete code. Good explanation. 	
Sample Output; incorrect and correct user inputs with a brief explanation.	C2	Missing sample and explanation.	<ul style="list-style-type: none"> Partial sample. Missing explanation. 	<ul style="list-style-type: none"> Complete sample. Missing explanation. 	<ul style="list-style-type: none"> Complete sample. Bad explanation. 	<ul style="list-style-type: none"> Complete sample. Good explanation. 	
Flow and Contents; as stated in requirements.	C2	<ul style="list-style-type: none"> Incorrect flow. Missing one or more contents. 	<ul style="list-style-type: none"> Incorrect flow. All contents included. 	<ul style="list-style-type: none"> Correct flow. Missing more than one content. 	<ul style="list-style-type: none"> Correct flow. Missing one content. 	<ul style="list-style-type: none"> Correct flow. All contents included. 	
Total Score							
Percentage = (Total Score ÷ 16 marks) × 10							

Presentation Rubric

CLO3: Write efficient computer programs to solve small-to-moderate scale problems. 10% contribution to the continuous assessment.

Criteria	Affe. Level	Poor 0	Unsatisfactory 1	Satisfactory 2	Good 3	Excellent 4	Score
Input; prompt, input statement method and validation.	A2	Not presented.	Cannot justify any usage.	Can justify some usage.	Can justify most usage.	Can justify all usage.	
Process; assignment statements and computations.	A3	Not presented.	Cannot justify any process.	Can justify some process.	Can justify most processes.	Can justify all processes.	
Output; form and value format.	A2	Not presented.	Cannot justify any usage.	Can justify some usage.	Can justify most usage.	Can justify all usage.	
Sequence; the flow of ordering.	A1	Not presented.	Cannot justify any flow.	Can justify some flow.	Can justify most flow.	Can justify all flow.	
Selection; type and condition.	A3	Not presented.	Cannot justify the chosen type and condition.	Can justify some chosen type and condition.	Can justify most chosen types and conditions.	Can justify all chosen types and conditions.	
Repetition; type and elements (initialisation or priming read, condition, and update).	A3	Not presented.	Cannot justify the chosen type and element.	Can justify some chosen type and element.	Can justify most chosen types and elements.	Can justify all chosen types and elements.	
Array; dimension, size, and traversal.	A3	Not presented.	Cannot justify any usage.	Can justify some usage.	Can justify most usage.	Can justify all usage.	
Functions; prototype, definition, and call.	A4	Not presented.	Cannot justify any usage.	Can justify some usage.	Can justify most usage.	Can justify all usage.	
Total Score							
Percentage = (Total Score ÷ 32 marks) × 10							

