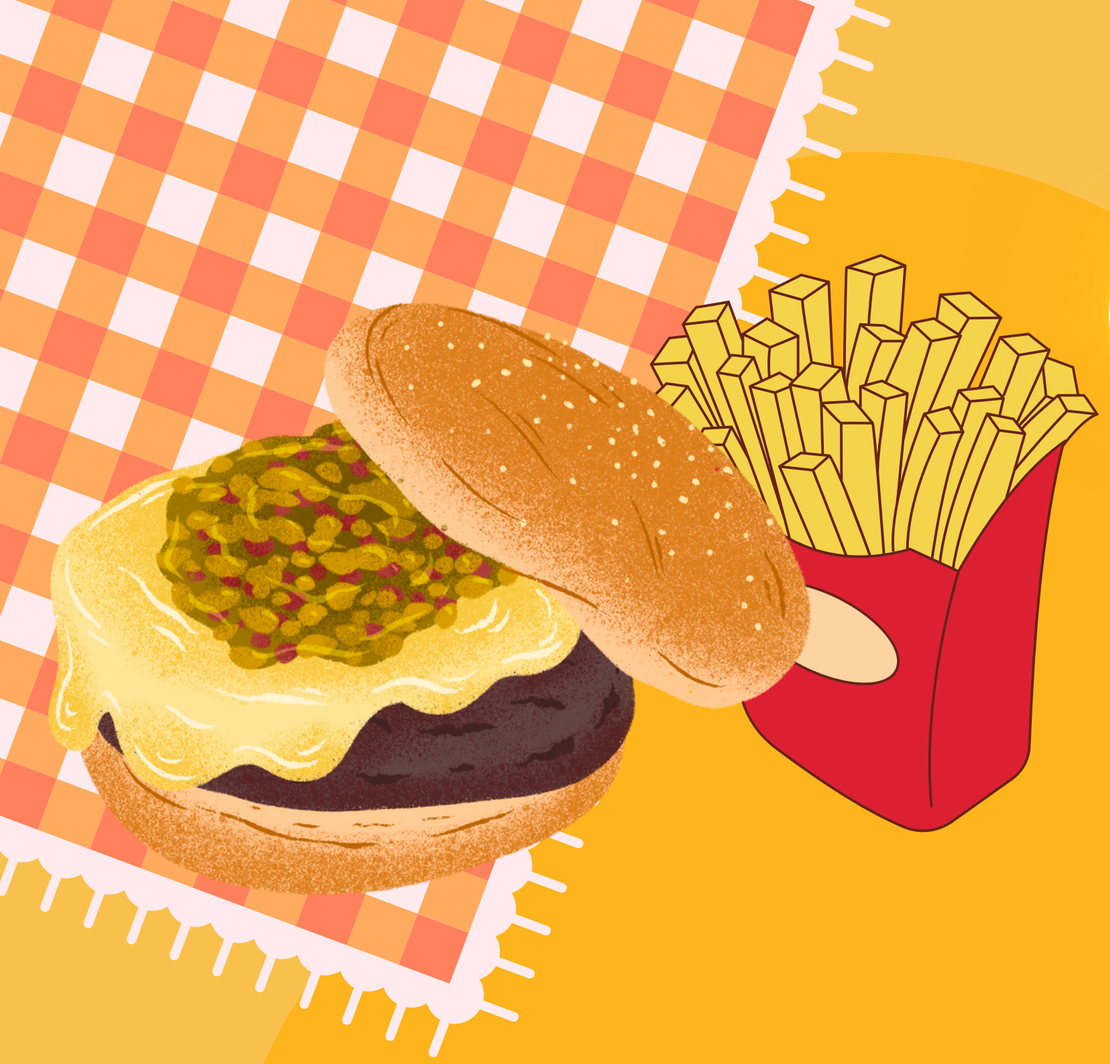
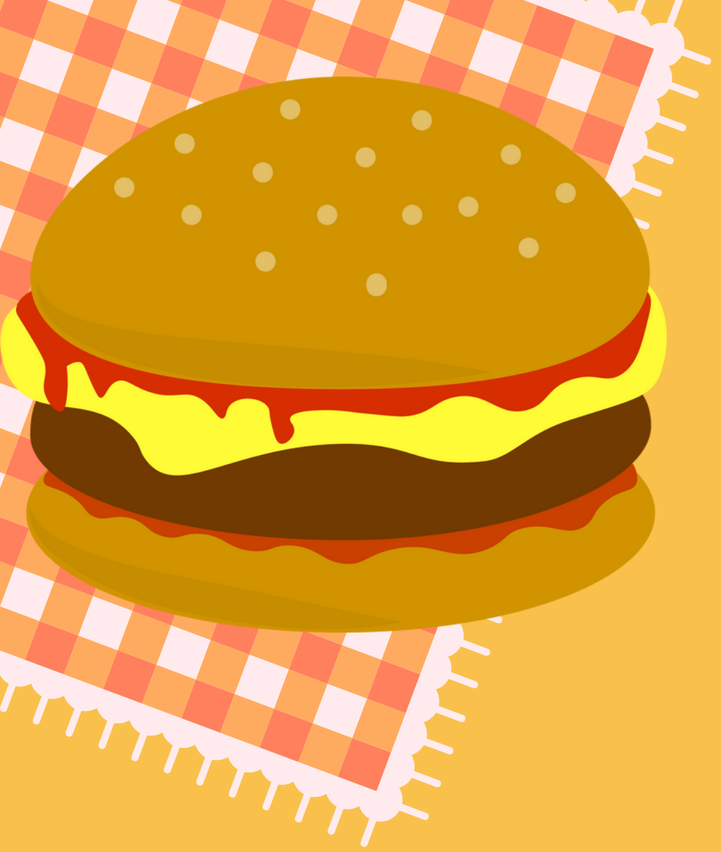


WELCOME!

G BURGERS

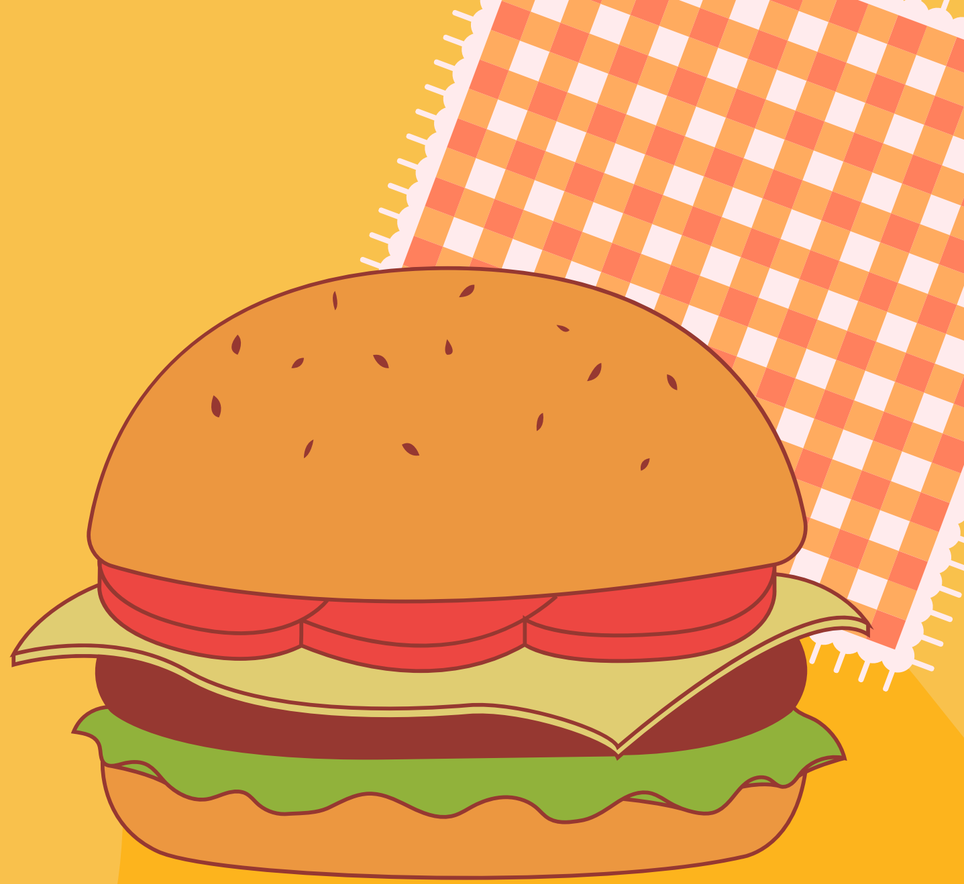
"Treat your taste buds to a journey of deliciousness with our handcrafted burgers."



PROPOSED PROJECT SUMMARY

Welcome to the world of burger stall management “The G Burgers”! As the owner of a busy burger stall, it's important to keep track of the daily inflow and outflow of ingredients such as buns, eggs, and meat. Keeping the customers informed about the current stock status and available menu items is also crucial to providing a good experience. In this scenario, we will explore how to overcome the challenges of managing inventory and sales, and how to provide a seamless experience for customers. By implementing a computerized system and keeping track of daily sales data, we can streamline the operations and ensure that the burger stall is always fully stocked and ready to serve its customers. We are proud to announce that our burger stall now offers a special burger with an egg! This delicious addition is sure to satisfy even the biggest appetites and is sure to become a customer favourite. And for those who are looking for a more classic option, our standard burger is still available and reasonably priced. With our juicy, succulent beef patty and soft, fluffy buns, our standard burger is a classic staple that never fails to impress. Whether you're in the mood for a classic burger or a new twist on an old favourite, our menu has something for everyone. And with our affordable prices, you can enjoy a delicious meal without breaking the bank. So come on in and treat yourself to a burger today!

The owner of a G burgers is facing a problem of keeping track of their daily stock of buns, eggs, and meat. They need to keep updating their available stock regularly to ensure they never run out of ingredients. At the same time, their customers are eager to know what the current stock is and what items are available on the menu. To solve this problem, the owner can consider implementing a computerized system to manage their inventory and sales. The system can keep track of the daily inflow and outflow of ingredients and provide real-time information on the available stock. The owner can also use the system to record the total sales for each day and analyse the data to make informed decisions about inventory management. Additionally, the owner can set up a display screen or a website where customers can view the current stock status and the menu items that are available. This will not only provide transparency to customers but also help the owner to manage their stock efficiently. By implementing these changes, the burger stall owner can streamline their operations and provide a better experience for their customers.



OBJECTIVES

1. KEEP TRACK OF THE AVAILABLE STOCK OF BUNS, EGGS, AND MEAT ON A DAILY BASIS AND ENSURE THAT THE INGREDIENTS ARE ALWAYS IN SUFFICIENT QUANTITY.
2. PROVIDE REAL-TIME INFORMATION TO CUSTOMERS ON THE CURRENT STOCK STATUS AND AVAILABLE MENU ITEMS.
3. STREAMLINE THE SALES AND INVENTORY MANAGEMENT PROCESS BY USING A COMPUTERIZED SYSTEM.
4. RECORD AND ANALYSE THE DAILY SALES DATA TO MAKE INFORMED DECISIONS ABOUT INVENTORY MANAGEMENT.
5. PROVIDE A BETTER EXPERIENCE FOR CUSTOMERS BY ENSURING THE AVAILABILITY OF INGREDIENTS AND TRANSPARENT INFORMATION ABOUT THE CURRENT STOCK STATUS.

ANALYSIS

INPUT

-Stocks items (buns,chicken,beef,egg)
-burger type (standard or special)
-meat type (chicken or beef)

PROCESS

-update available stock after customer purchased any burger variation
-Identify customer burger type and meet selected
-Calculate price of burger choosen by customer, calculate tax based on price of burger then calculate the final price of burger
-Calculate total sales in a day for the owner

OUTPUT

-BILLS FOR THE CUSTOMER
-TOTAL SALES MADE IN A DAY AND STOCKS LEFT FOR OWNER

DESIGN ALGORITHM (PSEUDOCODE)

PSUEDOCODE(MAIN)

```
START
 const int maxItem=4;
 int count=0 , stocks[maxItem] , burgerType;
 double price , taxAmount , totalPrice , totalSales;
 const double tax = 0.06;
 char meatOptn;
string burgerName , stockName[maxItem] = "Buns = " , "Chicken Meat = " , "Beef Meat = " , "Egg
= "

        ownerStock()
        WHILE ( repeatOrder == 'Y' || repeatOrder 'y')
                IF stocks [0]>0
                        IF (stocks [1]>0 || stocks [2]>0)
                        menuNstock()
                        brgOptn()
                        burgerName=burgerNaming()
                        priceCalc()
                        custBill()
                        DISPLAY "Make a new order? (Y to proceed):"
                        INPUT repetOrder
                        ELSE
                        repeatOrder='N'
                         DISPLAY "Sorry! Our buns ran out of stock"
                        DISPLAY "Thank You Come Again!"

                ELSE
 DISPLAY "Sorry! Our buns ran out of stock"

                DISPLAY "Thank You Come Again!"
                DISPLAY ENDLINE
grossNstock()
END WHILE
END IF ELSE
END
```

PSUEDOCODE_(OWNERSTOCK)

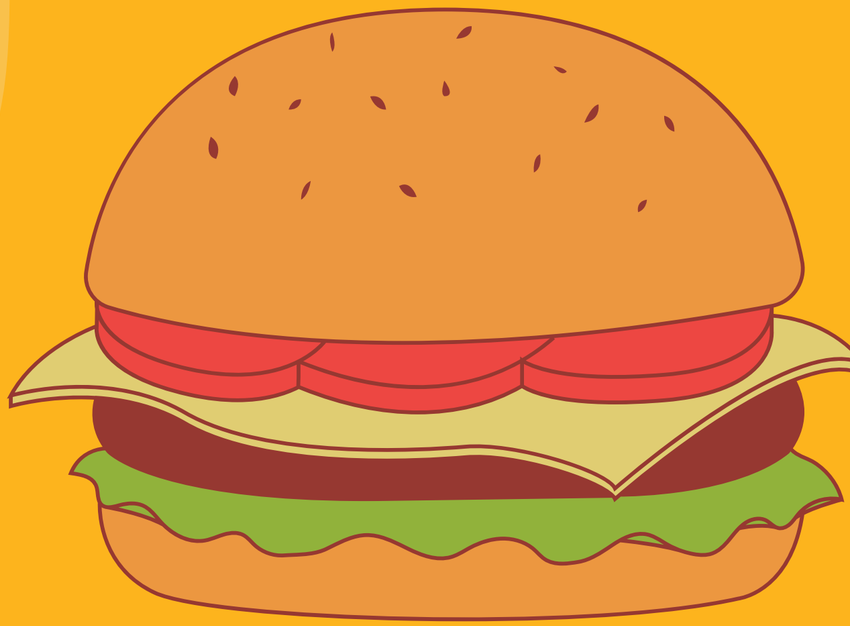
```
ownerStock()
char stockInput='Y'
DISPLAY "**************************************************"
DISPLAY “Good Morning Owner!”
 WHILE (stockInput=='Y' || stockInput=='y')
 DISPLAY "Please enter todays stock:"

 count=0
        FOR (count ; count<maxItem ; count++)
                DISPLAY stockName[count]
                INPUT stocks[count]
 DISPLAY ENDLINE
 IF (stocks[0] <0)
 DISPLAY "INVALID Buns Stock!"
 IF (stocks[1] <0)
 DISPLAY "INVALID Chicken Meat Stock!"
 IF (stocks[2] <0)
 DISPLAY "INVALID Beef Meat Stock!"
 IF (stocks[3] <0)
 DISPLAY "INVALID Egg Stocks!"
 IF (stocks[0]>0 && (stocks[1]>0 || stocks[2]>0))
 DISPLAY "Want to re-enter stocks? (Y to proceed) : "
 INPUT stockInput

 DISPLAY “Happy Selling!”
 DISPLAY "**************************************************"
```

PSEUDOCODE(MENUNSTOCK)

```
void menuNstock()
DISPLAY "=================================================="
DISPLAY “MENU”

DISPLAY "Chicken Burger  (RM3.50)            Stocks:"
                                IF (stocks[1]>0)
                                DISPLAY stocks[1]
                                ELSE
                                DISPLAY "OUT OF STOCK"
                                DISPLAY "Beef Burger     (RM4.50)            Stocks:"
                                IF (stocks[2]>0)
                                DISPLAY stocks[2]
                                ELSE
                                DISPLAY "OUT OF STOCK"
                                DISPLAY "~ Egg add-ons   (RM1.00)            Stocks:"
                                IF "~ Egg add-ons   (RM1.00)            Stocks:"
                                IF (stocks[3]>0)
                                DISPLAY stocks[3]
                                ELSE
                                DISPLAY "OUT OF STOCK"
DISPLAY "================================================="
DISPLAY “----ORDER----"
DISPLAY "1) Special (Egg add-ons)"
DISPLAY "2) Standard "
```

PSEUDOCODE_BRGOPTN()

```
void brgOptn()
Price =0
DISPLAY "Enter Burger Type (1/2) : "
INPUT burgerType
DISPLAY ENDLINE

SWITCH (burgerType)
case 1:
    stocks[3]--
    price++
case 2:
    meatSelector()
BREAK
DEFAULT :
 DISPLAY "INVALID BURGER TYPE"
 brgrOptn()
```



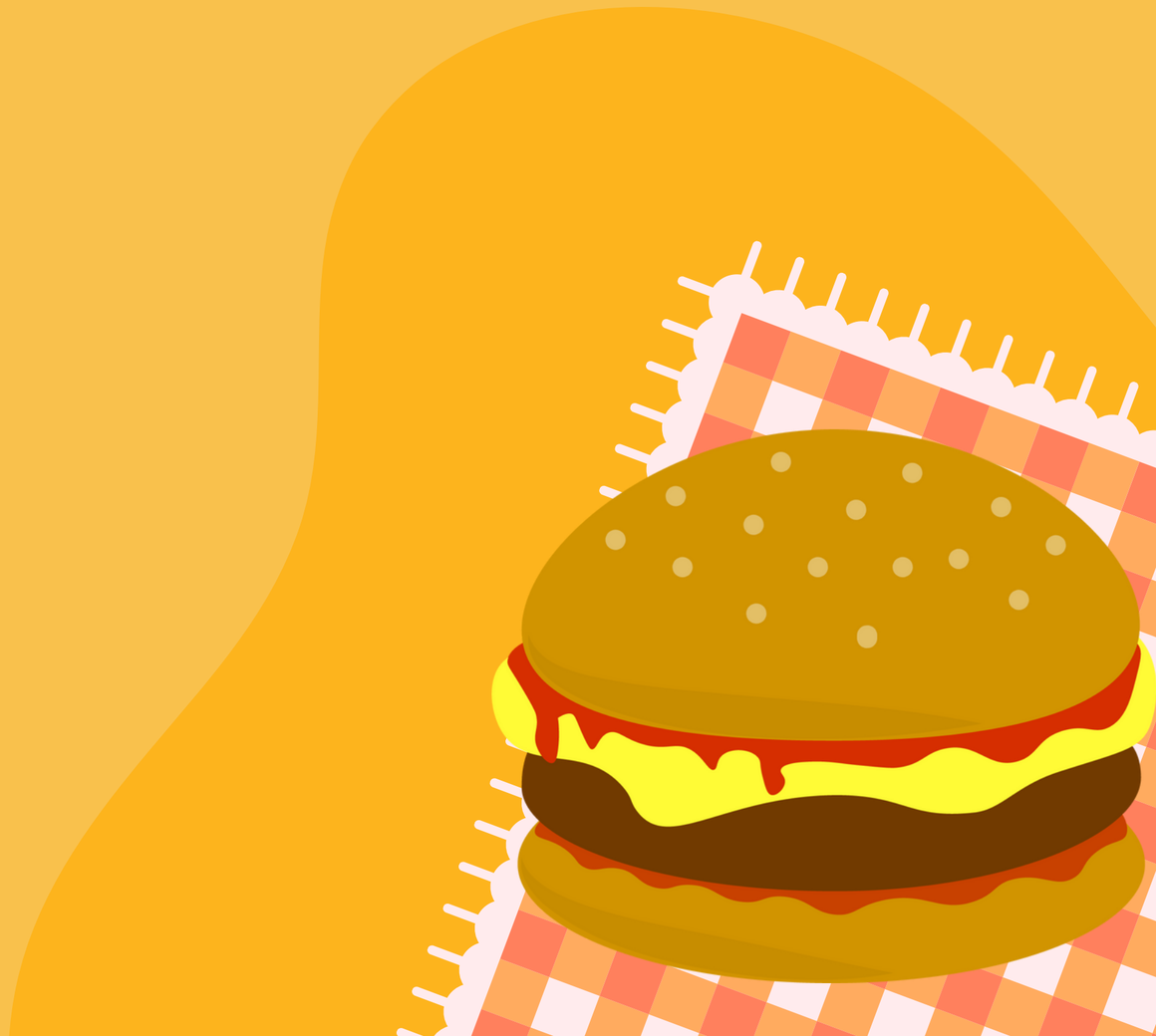
PSEUDOCE_MEATSELECTOR

```
DISPLAY "Enter Burger Meat (C/B) : "
INPUT meatOptn

        IF (meatOptn=='C' || meatOptn=='c')
                stocks[0]--
                stocks[1]--
                price=price+3.5
        ELSE IF (meatOptn=='B' || meatOptn=='b')
                stocks[0]--
                stocks[2]--
                price=price+4.5
        ELSE
 price=0;
                cout<<endl<<"INVALID BURGER MEAT"<<endl<<endl;
                meatSelector();
string burgerNaming()
        IF (burgerType==1)
                IF (meatOptn=='C' || meatOptn=='c')
                RETURN "Special Chicken Burger"
 ELSE IF (meatOptn=='B' || meatOptn=='b')
 RETURN "Special Beef Burger"

ELSE IF (burgerType==2)
        IF (meatOptn=='C' || meatOptn=='c')
        RETURN "Chicken Burger"
        ELSE IF (meatOptn=='B' || meatOptn=='b')
        RETURN return "Beef Burger"
```

PSEUDOCODE_PRICECALC()

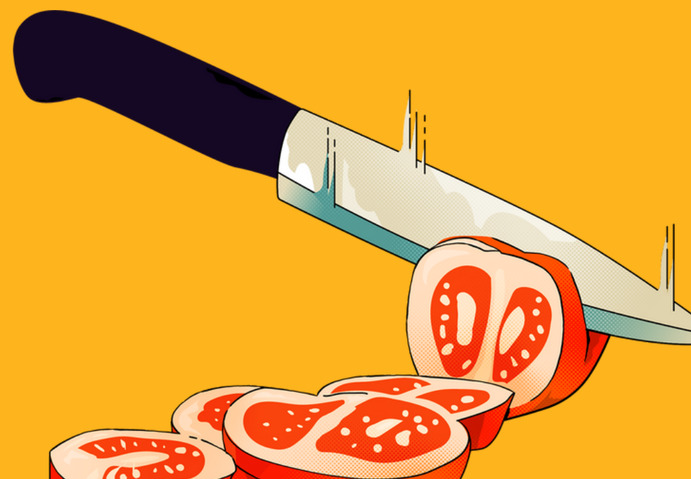
void priceCalc()

taxAmount=price*tax
totalPrice=price+taxAmount
totalSales=totalSales+totalPrice



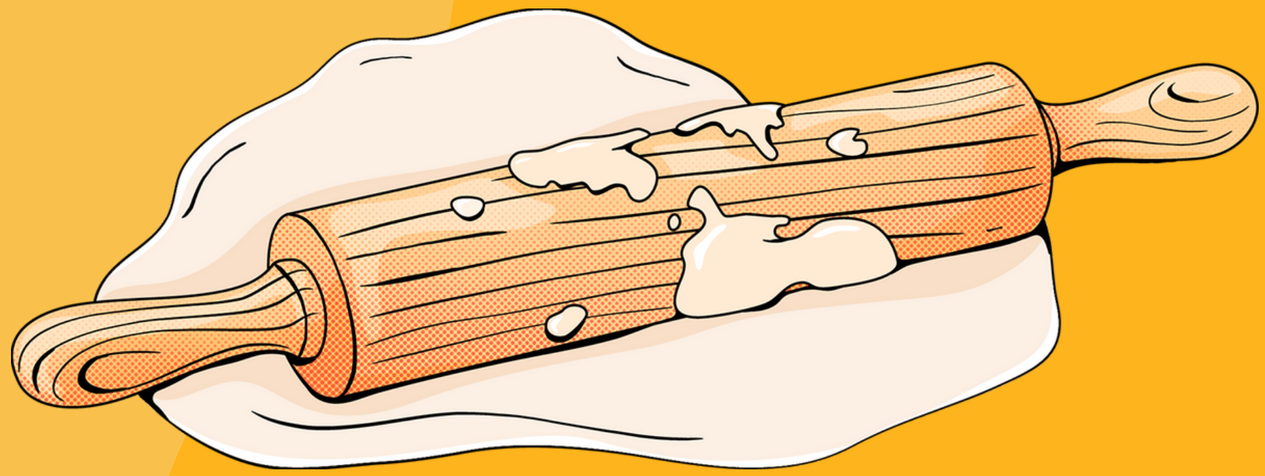
PSEUDOCODE_CUSTBILL()

```
Pseudocode_custBill()
void custBill()
DISPLAY
"#################################################"
DISPLAY “Bills"

DISPLAY burgerName
DISPLAY "Price :RM ", price
DISPLAY "Tax(6%) :RM ", taxAmount
DISPLAY "Total Price :RM ", totalPrice

DISPLAY
"#################################################"
```

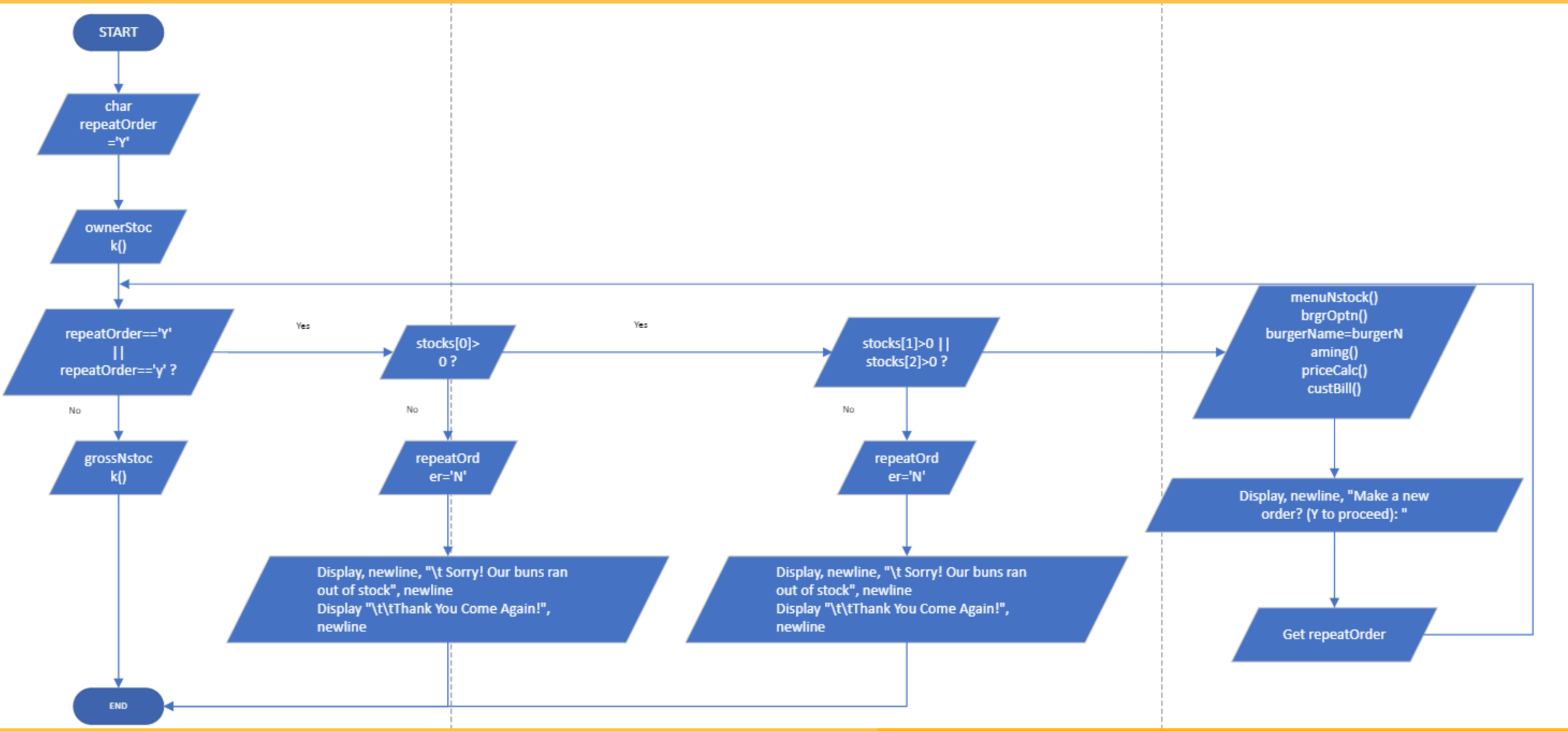


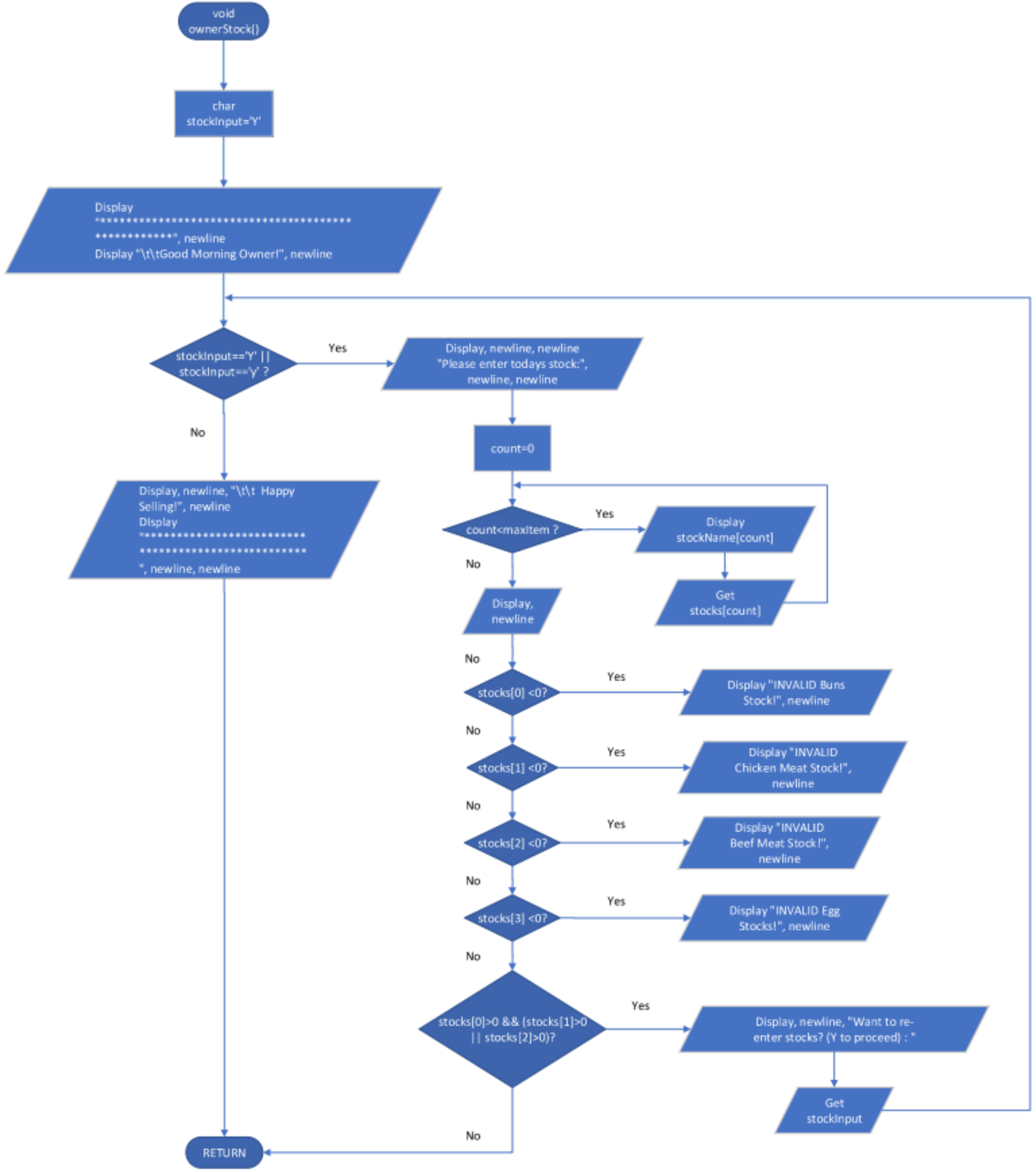
PSEUDOCODE_GROSSNSTOCK()

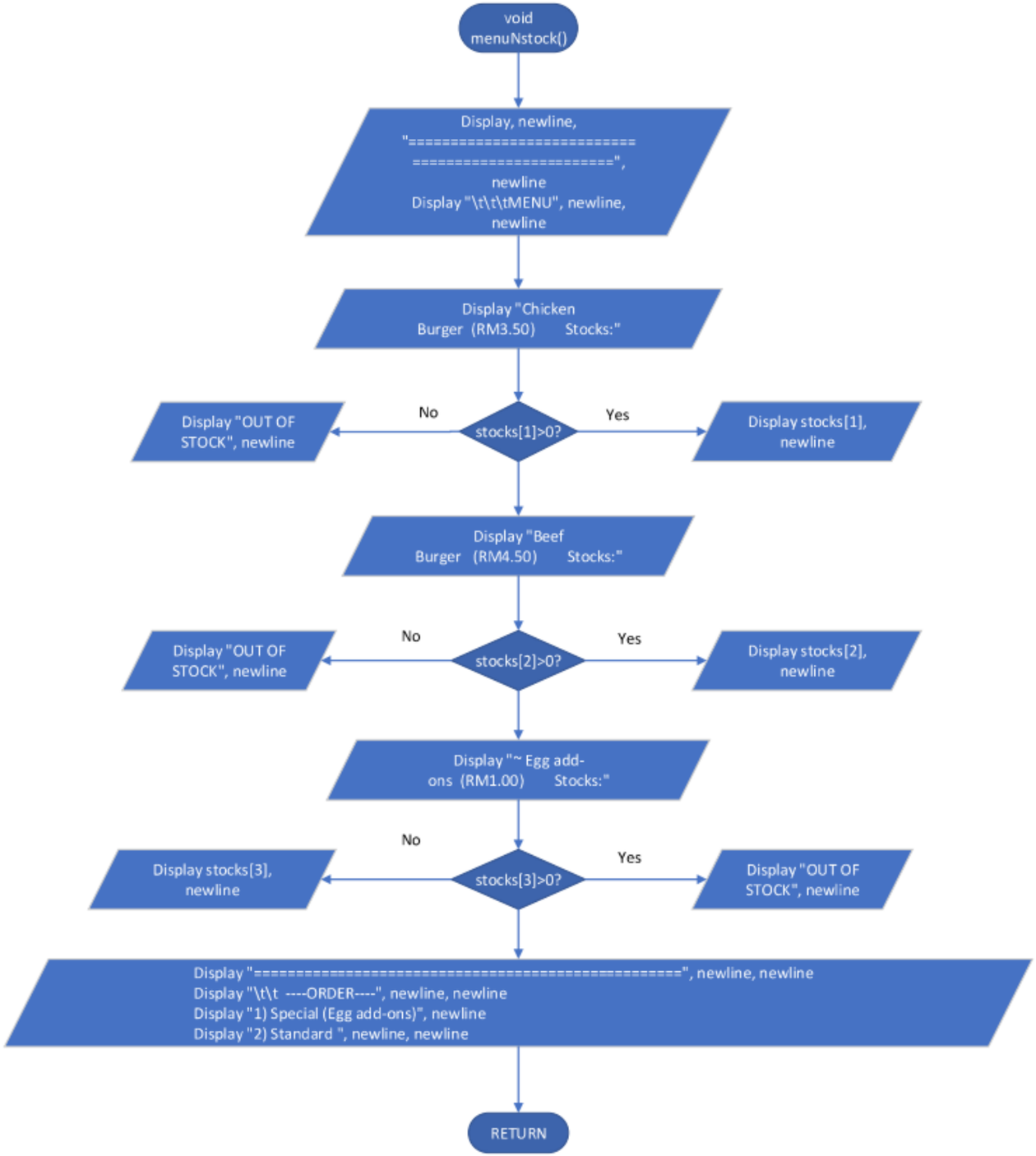
```
DISPLAY "-----------------------------------------------"
DISPLAY “-Todays Statistic|-"
DISPLAY "Total Sales: RM", totalSales
DISPLAY "Stocks Left:"
Count=0
        FOR (count ; count<maxItem ; count++)
DISPLAY stockName[count]
DISPLAY stocks[count]
DISPLAY "-----------------------------------------------"
```

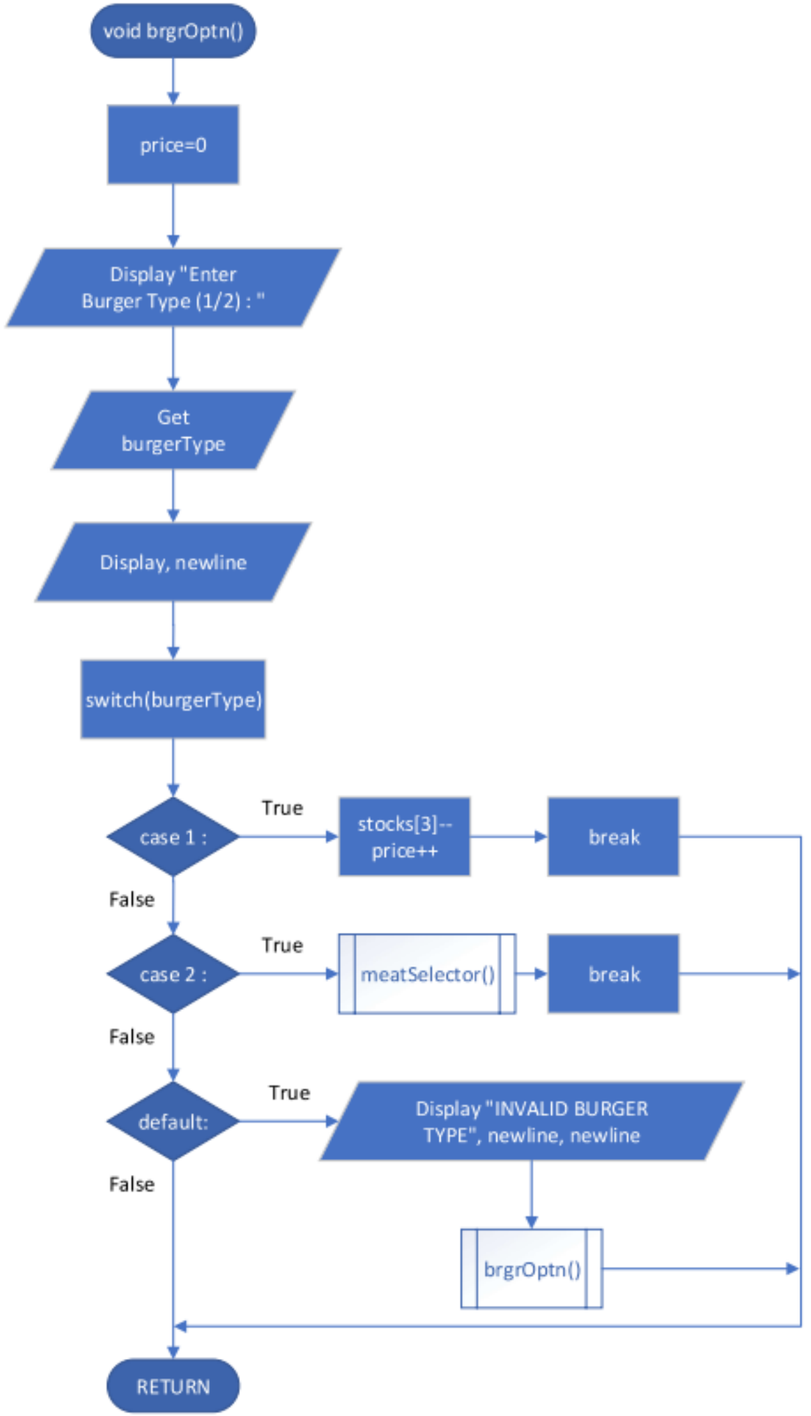


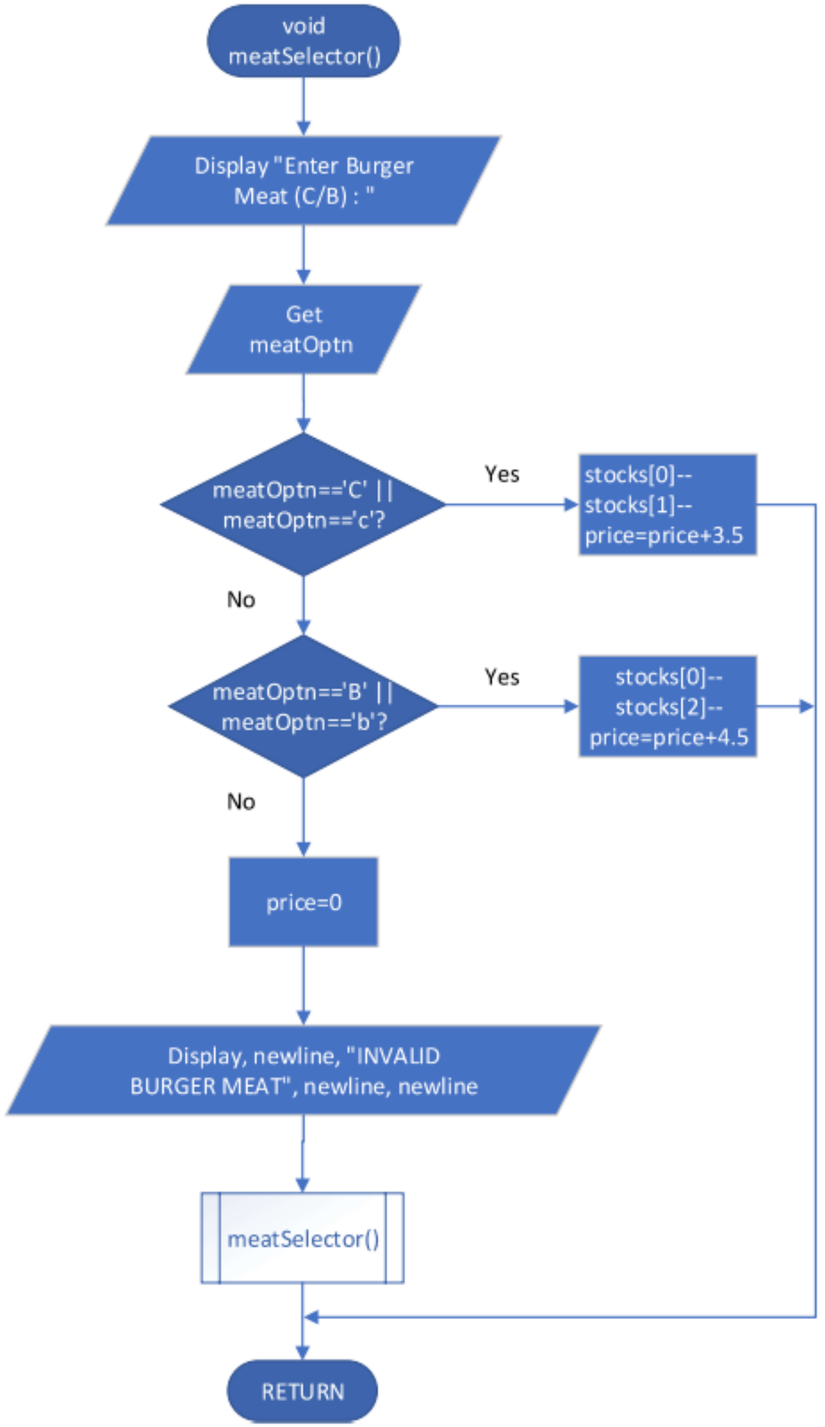
FLOWCHART

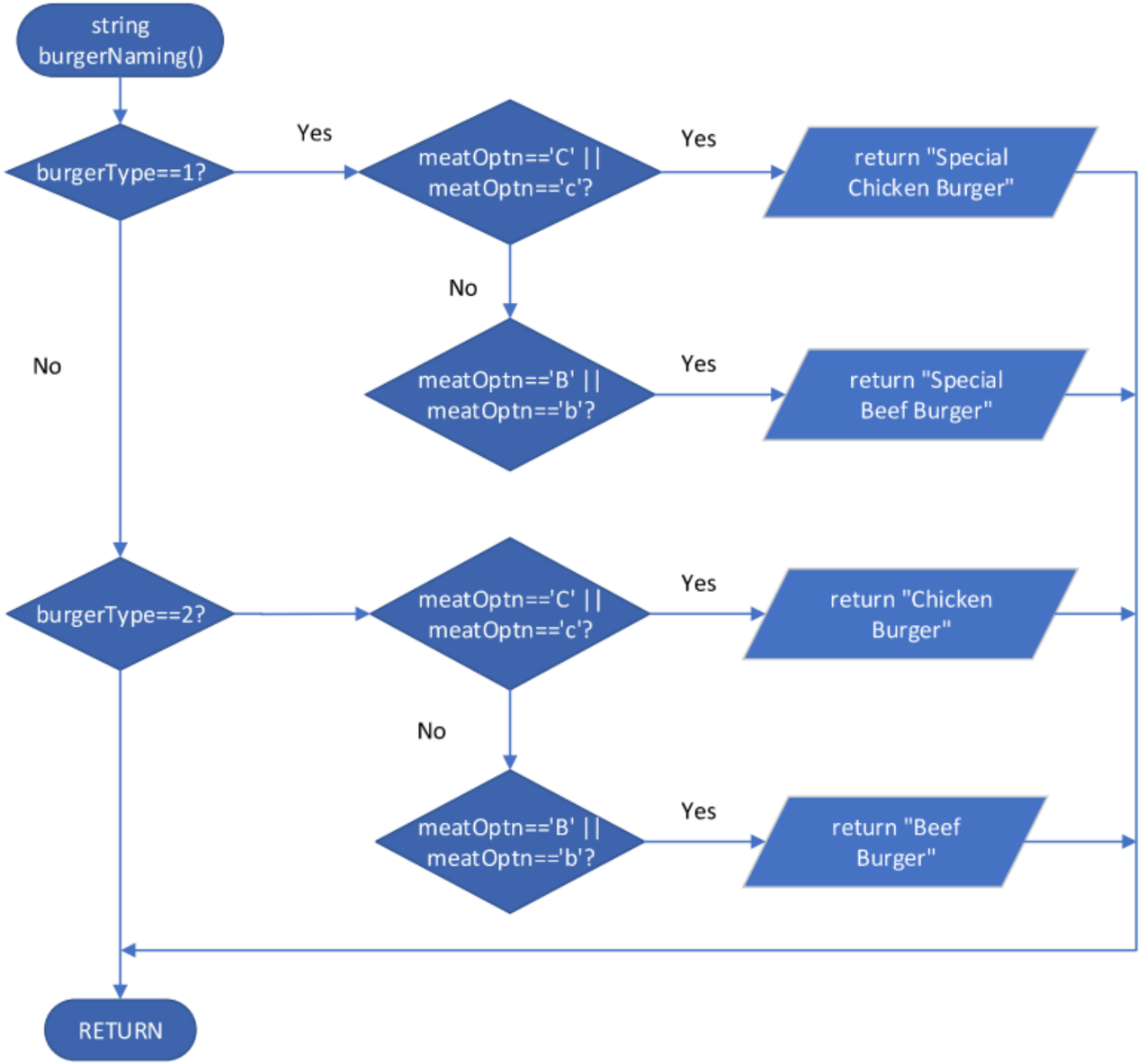

START
char repeatOrder ='Y'
ownerStock()
repeatOrder=='Y' || repeatOrder=='y' ?
Yes
No
stocks[0]> 0 ?
Yes
No
stocks[1]>0 || stocks[2]>0 ?
No
menuNstock()
brgrOptn()
burgerName=burgerNaming()
priceCalc()
custBill()
grossNstock()
repeatOrder='N'
repeatOrder='N'
Display, newline, "\t Sorry! Our buns ran out of stock", newline
Display "\t\tThank You Come Again!", newline
Display, newline, "\t Sorry! Our buns ran out of stock", newline
Display "\t\tThank You Come Again!", newline
Display, newline, "Make a new order? (Y to proceed): "
Get repeatOrder
END

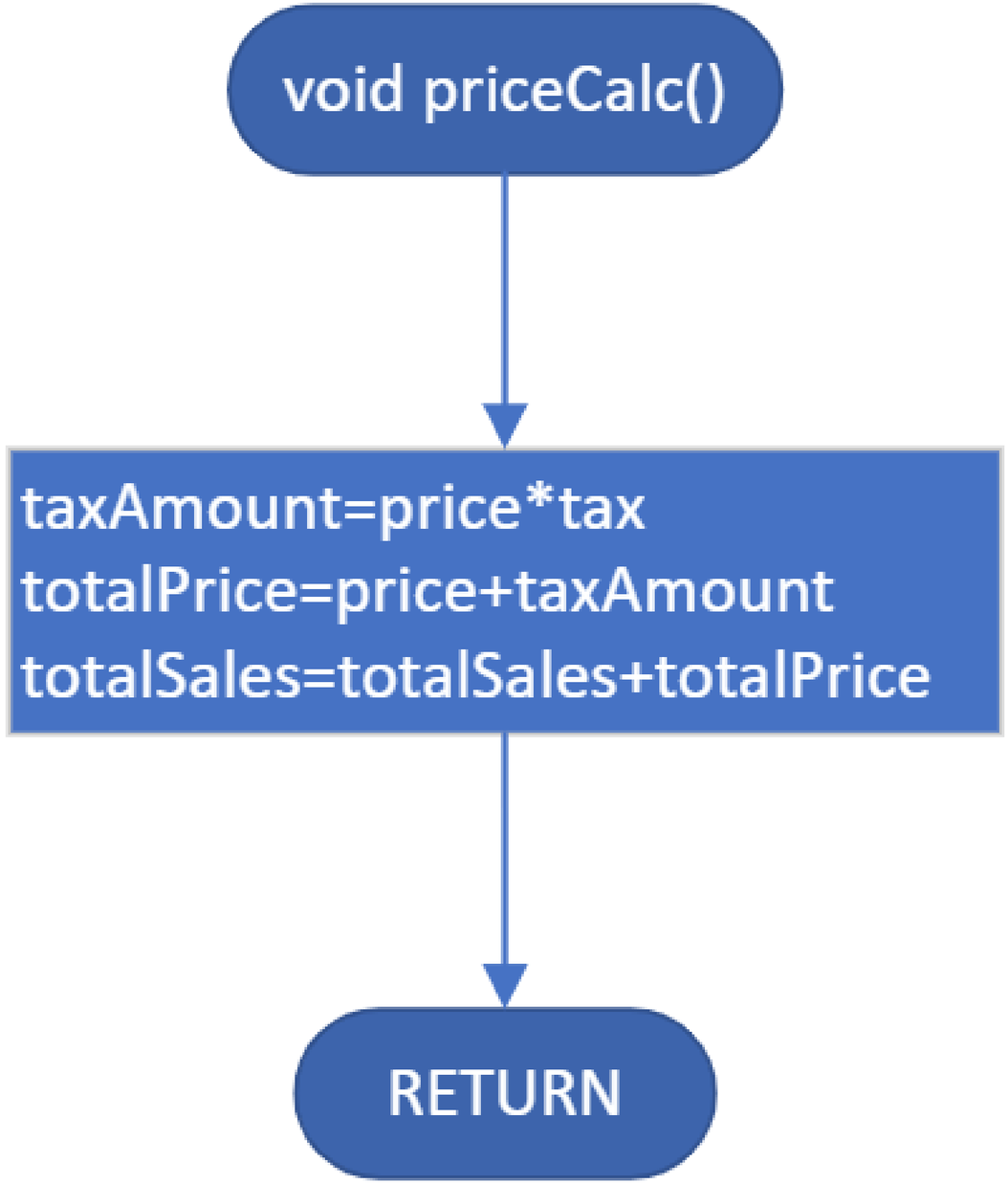

void ownerStock()
char stockInput='Y'
Display "**********", newline
Display "\t\tGood Morning Owner!", newline
stockInput=='Y' || stockInput=='y' ?
Yes
Display, newline, newline "Please enter todays stock:", newline, newline
No
Display, newline, "\t\t Happy Selling!", newline
DIsplay "**********", newline, newline
count=0
count<maxItem ?
Yes
Display stockName[count]
Get stocks[count]
No
Display, newline
No
stocks[0] <0?
Yes
Display "INVALID Buns Stock!", newline
No
stocks[1] <0?
Yes
Display "INVALID Chicken Meat Stock!", newline
No
stocks[2] <0?
Yes
Display "INVALID Beef Meat Stock!", newline
No
stocks[3] <0?
Yes
Display "INVALID Egg Stocks!", newline
No
stocks[0]>0 && (stocks[1]>0 || stocks[2]>0)?
Yes
Display, newline, "Want to re-enter stocks? (Y to proceed) : "
Get stockInput
No
RETURN

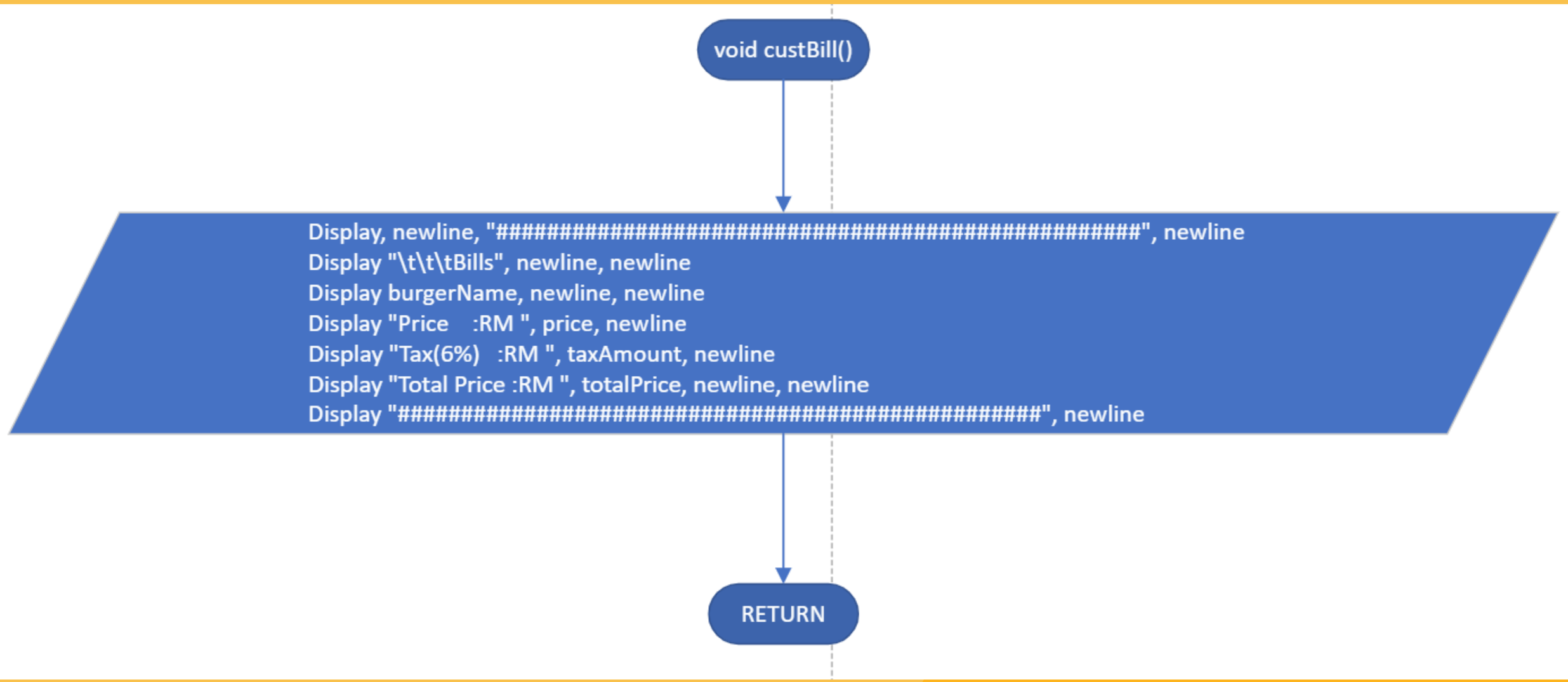

void menuNstock()
Display, newline, "========== ==========", newline
Display "\t\t\tMENU", newline, newline
Display "Chicken Burger (RM3.50) Stocks:"
stocks[1]>0?
No
Display "OUT OF STOCK", newline
Yes
Display stocks[1], newline
Display "Beef Burger (RM4.50) Stocks:"
stocks[2]>0?
No
Display "OUT OF STOCK", newline
Yes
Display stocks[2], newline
Display "~ Egg add-ons (RM1.00) Stocks:"
stocks[3]>0?
No
Display stocks[3], newline
Yes
Display "OUT OF STOCK", newline
Display "==========", newline, newline
Display "\t\t ----ORDER----", newline, newline
Display "1) Special (Egg add-ons)", newline
Display "2) Standard ", newline, newline
RETURN

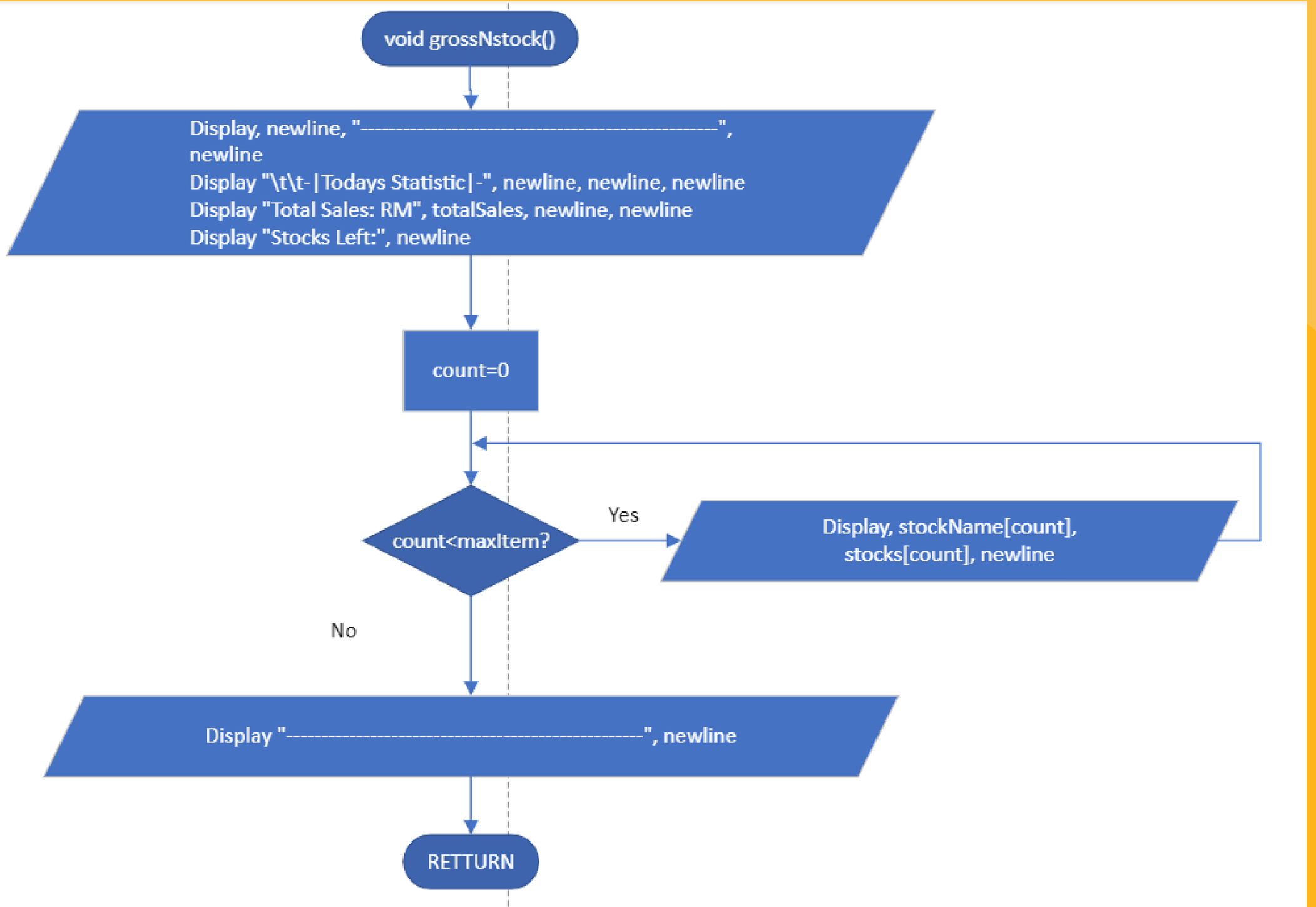

void brgrOptn()
price=0
Display "Enter Burger Type (1/2) : "
Get burgerType
Display, newline
switch(burgerType)
case 1 :
True
stocks[3]--
price++
break
False
case 2 :
True
meatSelector()
break
False
default:
True
Display "INVALID BURGER TYPE", newline, newline
brgrOptn()
False
RETURN


void
meatSelector()
Display "Enter Burger
Meat (C/B) : "
Get
meatOptn
meatOptn=='C' ||
meatOptn=='c'?
Yes
stocks[0]--
stocks[1]--
price=price+3.5
No
meatOptn=='B' ||
meatOptn=='b'?
Yes
stocks[0]--
stocks[2]--
price=price+4.5
No
price=0
Display, newline, "INVALID
BURGER MEAT", newline, newline
meatSelector()
RETURN


string
burgerNaming()
burgerType==1?
Yes
meatOptn=='C' ||
meatOptn=='c'?
Yes
return "Special
Chicken Burger"
No
meatOptn=='B' ||
meatOptn=='b'?
Yes
return "Special
Beef Burger"
No
burgerType==2?
meatOptn=='C' ||
meatOptn=='c'?
Yes
return "Chicken
Burger"
No
meatOptn=='B' ||
meatOptn=='b'?
Yes
return "Beef
Burger"
RETURN


void priceCalc()
taxAmount=price*tax
totalPrice=price+taxAmount
totalSales=totalSales+totalPrice
RETURN


void custBill()
Display, newline, "##", newline
Display "\t\t\tBills", newline, newline
Display burgerName, newline, newline
Display "Price   :RM ", price, newline
Display "Tax(6%)   :RM ", taxAmount, newline
Display "Total Price :RM ", totalPrice, newline, newline
Display "##", newline
RETURN


void grossNstock()
Display, newline, "----------",
newline
Display "\t\t-|Todays Statistic|-", newline, newline, newline
Display "Total Sales: RM", totalSales, newline, newline
Display "Stocks Left:", newline
count=0
count<maxItem?
Yes
Display, stockName[count],
stocks[count], newline
No
Display "----------", newline
RETTURN

Source Code

```cpp
#include<iostream>
using namespace std;

// all functions

void ownerStock();      // owner enter item stocks
void menuNstock();      // display menu and item stocks availability
void brgrOptn();        // customer input their choice of burger type
void meatSelector();    // customer input their meat of choice
void custBill();        // display burger name based on customer choice and total price after tax
string burgerNaming(); // determines the burger name based on customer's burger choice
void priceCalc();       // calculates price of burger, calculates tax based on burger price and summation of total sales
void grossNstock();     // display total sales made and remainding stocks left

// all data types used and its variable

const int maxItem=4;
int count=0 , stocks[maxItem] , burgerType;
double price , taxAmount , totalPrice , totalSales;
const double tax = 0.06;
char meatOptn;
string burgerName , stockName[maxItem] = {"Buns = " , "Chicken Meat = " , "Beef Meat = " , "Egg = "};

//

int main()
{
    char repeatOrder='Y';

    ownerStock();

```

```
31
32      while(repeatOrder=='Y' || repeatOrder=='y')
33      {
34          if(stocks[0]>0) // checks if buns is still in stock
35          {
36              if(stocks[1]>0 || stocks[2]>0) // checks if meats is still in stock
37              {
38                  menuNstock();
39                  brgrOptn();
40                  burgerName=burgerNaming();
41                  priceCalc();
42                  custBill();
43
44                  cout<<endl<<"Make a new order? (Y to proceed): "; // prompt customer to make new order
45                  cin>>repeatOrder;
46              }
47              else
48              {
49                  repeatOrder='N';
50                  cout<<endl<<"\t  Sorry! Our meats ran out of stock"<<endl;
51                  cout<<"\t\tThank You Come Again!"<<endl;
52              }
53          }
54          else
55          {
56              repeatOrder='N';
57              cout<<endl<<"\t  Sorry! Our buns ran out of stock"<<endl;
58              cout<<"\t\tThank You Come Again!"<<endl;
59          }
60      }
61      grossNstock();
```

```
61      grossNstock();
62  }
63
64  void ownerStock()
65  {
66      char stockInput='Y';
67
68      cout<<"*************************************************"<<endl;
69      cout<<"\t\tGood Morning Owner!"<<endl;
70
71      while(stockInput=='Y' || stockInput=='y')
72      {
73          cout<<endl<<"Please enter todays stock:"<<endl<<endl;
74
75          count=0;
76          for(count ; count<maxItem ; count++)  // owner input buns,chicken,beef and egg stock
77          {
78              cout<<stockName[count];
79              cin>>stocks[count];
80          }
81
82          cout<<endl;
83
84          // checks if stock input is valid
85
86          if(stocks[0] <0)
87              cout<<"INVALID Buns Stock!"<<endl;
88
89          if(stocks[1] <0)
90              cout<<"INVALID Chicken Meat Stock!"<<endl;
91
```

```
91
92          if(stocks[2] <0)
93              cout<<"INVALID Beef Meat Stock!"<<endl;
94
95          if(stocks[3] <0)
96              cout<<"INVALID Egg Stocks!"<<endl;
97
98          if(stocks[0]>0 && (stocks[1]>0 || stocks[2]>0)) // checks if owner is eligible to make any progress
99          {
100             cout<<endl<<"Want to re-enter stocks? (Y to proceed) : "; // prompt owner to re-enter stocks
101             cin>>stockInput;
102         }
103
104     }
105
106     cout<<endl<<"\t\t   Happy Selling!"<<endl;
107     cout<<"************************************************"<<endl<<endl;
108
109
110 }
111
112 void menuNstock()
113 {
114     cout<<endl<<"================================================="<<endl;
115     cout<<"\t\t\tMENU"<<endl<<endl;
116
117
118     cout<<"Chicken Burger   (RM3.50)               Stocks:";
119     if(stocks[1]>0)
120         cout<<stocks[1]<<endl;
121     else
```

```
121     else
122         cout<<"OUT OF STOCK"<<endl;
123 
124     cout<<"Beef Burger      (RM4.50)                Stocks:";
125     if(stocks[2]>0)
126         cout<<stocks[2]<<endl;
127     else
128         cout<<"OUT OF STOCK"<<endl;
129 
130     cout<<"~ Egg add-ons    (RM1.00)                Stocks:";
131         if(stocks[3]>0)
132         cout<<stocks[3]<<endl;
133     else
134         cout<<"OUT OF STOCK"<<endl;
135 
136     cout<<"=================================================="<<endl<<endl;
137 
138     cout<<"\t\t   ----ORDER----"<<endl<<endl;
139 
140     cout<<"1) Special (Egg add-ons)"<<endl;
141     cout<<"2) Standard "<<endl<<endl;
142 }
143 
144 void brgrOptn()
145 {
146     price=0;
147     cout<<"Enter Burger Type (1/2) : "; // prompt customer enter burger type
148     cin>>burgerType;
149     cout<<endl;
150 
151     switch(burgerType)
```

```
151     switch(burgerType)
152     {
153         case 1: {   stocks[3]--;
154                     price++;
155                 }
156 
157         case 2: {
158                     meatSelector();
159                 }
160 
161         break;
162 
163         default:{
164                     cout<<"INVALID BURGER TYPE"<<endl<<endl;
165                     brgrOptn();
166                 }
167     }
168 }
169 
170 void meatSelector()
171 {
172     cout<<"Enter Burger Meat (C/B) : "; // prompt customer enter burger meat
173     cin>>meatOptn;
174 
175     if(meatOptn=='C' || meatOptn=='c')
176     {
177         stocks[0]--;
178         stocks[1]--;
179         price=price+3.5;
180     }
181     else if(meatOptn=='B' || meatOptn=='b')
```

```
    else if(meatOptn=='B' || meatOptn=='b')
    {
        stocks[0]--;
        stocks[2]--;
        price=price+4.5;
    }
    else
    {
        price=0;
        cout<<endl<<"INVALID BURGER MEAT"<<endl<<endl;
        meatSelector();
    }
}

string burgerNaming()
{
    if(burgerType==1)
    {
        if(meatOptn=='C' || meatOptn=='c')
        {
            return "Special Chicken Burger";
        }
        else if(meatOptn=='B' || meatOptn=='b')
        {
            return "Special Beef Burger";
        }
    }
    else if(burgerType==2)
    {
        if(meatOptn=='C' || meatOptn=='c')
        {
```

```
        {
            return "Chicken Burger";
        }
        else if(meatOptn=='B' || meatOptn=='b')
        {
            return "Beef Burger";
        }
    }

}

void priceCalc()
{
    taxAmount=price*tax;
    totalPrice=price+taxAmount;
    totalSales=totalSales+totalPrice;
}

void custBill()
{
    cout<<endl<<"#####################################################"<<endl;
    cout<<"\t\t\tBills"<<endl<<endl;

    cout<<burgerName<<endl<<endl;
    cout<<"Price       :RM "<<price<<endl;
    cout<<"Tax(6%)     :RM "<<taxAmount<<endl;
    cout<<"Total Price :RM "<<totalPrice<<endl<<endl;

    cout<<"#####################################################"<<endl;
}

```

```
241
242 void grossNstock()
243 {
244     cout<<endl<<"-----------------------------------------------------"<<endl;
245     cout<<"\t\t-|Todays Statistic|-"<<endl<<endl<<endl;
246
247     cout<<"Total Sales: RM"<<totalSales<<endl<<endl;
248
249     cout<<"Stocks Left:"<<endl;
250     count=0;
251     for(count ; count<maxItem ; count++)  // displays stocks left
252     {
253         cout<< stockName[count] << stocks[count] << endl;
254     }
255
256     cout<<"-----------------------------------------------------"<<endl;
257 }
```

SAMPLE OUTPUT

```
--------------------------------------------------
               -|Todays Statistic|-


Total Sales: RM20.14

Stocks Left:
Buns = 96
Chicken Meat = 48
Beef Meat = 38
Egg = 57
--------------------------------------------------

------------------------------
Process exited after 44.88 seconds with return value 0
Press any key to continue . . .
```

```
                Good Morning Owner!

Please enter todays stock:

Buns = -5
Chicken Meat = 26
Beef Meat = 9
Egg = -2

INVALID Buns Stock!
INVALID Egg Stocks!

Please enter todays stock:

Buns =
```

invalid outpuut

```
************************************************
               Good Morning Owner!

Please enter todays stock:

Buns = 80
Chicken Meat = 55
Beef Meat = 30
Egg = 42


Want to re-enter stocks? (Y to proceed) : N

                  Happy Selling!
************************************************


================================================
                      MENU

Chicken Burger   (RM3.50)              Stocks:55
Beef Burger      (RM4.50)              Stocks:30
~ Egg add-ons    (RM1.00)              Stocks:42
================================================

                  ----ORDER----

1) Special (Egg add-ons)
2) Standard

Enter Burger Type (1/2) : 1

Enter Burger Meat (C/B) : B

################################################
                      Bills

Special Beef Burger

Price       :RM 5.5
Tax(6%)     :RM 0.33
Total Price :RM 5.83

################################################

Make a new order? (Y to proceed): N

------------------------------------------------
               -|Todays Statistic|-


Total Sales: RM5.83

Stocks Left:
Buns = 79
Chicken Meat = 55
Beef Meat = 29
Egg = 41
------------------------------------------------

------------------------------
Process exited after 21.28 seconds with return value 0
Press any key to continue . . .
```

THANK YOU FOR YOUR ATTENTION

"Thank you for your time and consideration. We appreciate the opportunity to share our passion for burgers with you."