Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

SKU: 9780134801155-SOLUTIONS

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Starting Out with Programming Logic and Design, 5th Edition

Answers to Review Questions

Chapter 2

Multiple Choice

1. C
2. B
3. D
4. B
5. A
6. C
7. C
8. A
9. B
10. D
11. B
12. A
13. C
14. A
15. D
16. B
17. B
18. C
19. D
20. A

True or False

1. False
2. True
3. False
4. True
5. False
6. True
7. True
8. True
9. False
10. False

©2016 Pearson Education, Inc. Hoboken, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Short Answer

1. Interview the customer
2. An informal language that has no syntax rules, and is not meant to be compiled or executed. Instead programmers use pseudocode to create models or “mock-ups” of programs.
3. (1) Input is received. (2) Some process is performed. (3) Output is produced.
4. The term user-friendly is commonly used in the software business to describe programs that are easy to use.
5. The variable’s name and data type.
6. It depends on the language being used. Each language has its own way of handling uninitialized variables. Some languages assign a default value such as 0 to uninitialized variables. In many languages, however, uninitialized variables hold unpredictable values. This is because those languages set aside a place in memory for the variable, but do not alter the contents of that place in memory. As a result, an uninitialized variable holds the value that happens to be stored in its memory location. Programmers typically refer to unpredictable values such this as "garbage."

Algorithm Workbench

1. ```
   Display "Enter your height."
   Input height
   ```

2. ```
   Display "Enter your favorite color."
   Input color
   ```

3. a) `Set b = a + 2`
   b) `Set a = b * 4`
   c) `Set b = a / 3.14`
   d) `Set a = b - 8`

4. a) 12
   b) 4
   c) 2
   d) 6

5. ```
   Declare Real cost
   ```

6. ```
   Declare Integer total = 0
   ```

7. ```
   Set count = 27
   ```

8. ```
   Set total = 10 + 14
   ```

©2016 Pearson Education, Inc. Hoboken, NJ. All Rights Reserved.

9. `Set due = downPayment - total`

10. `Set totalFee = subtotal * 0.15`

11. 11

12. 5

Debugging Exercises

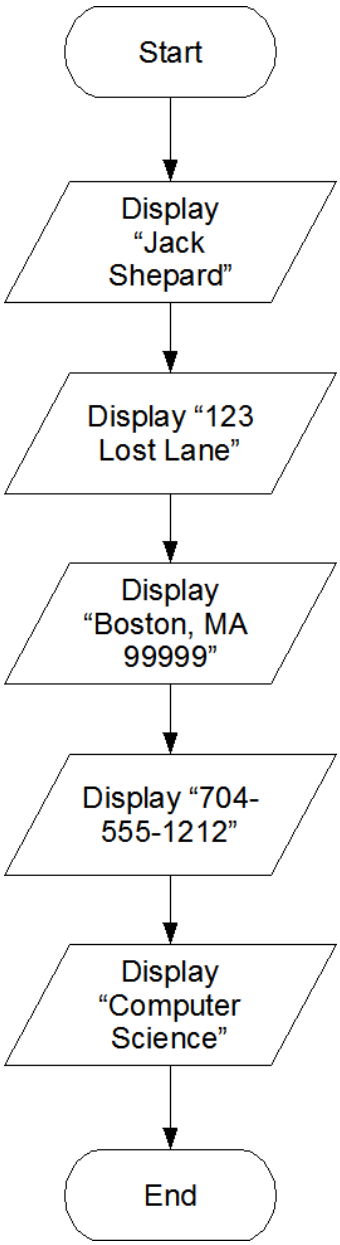
1. The variable name is enclosed in quotes. This is an error because the variable's name will be displayed instead of the variable's value.
2. The first character of the variable name begins with a number. This is an error because most programming languages do not allow variable names to begin with numbers.
3. The expression is missing parentheses. This is an error because, as the order of operations dictates, the division will occur before the addition and the result will be incorrect.
4. The variable is being used before it has been declared. This is an error because most programming languages do not allow variables to be used before they are declared.
5. The variables are being used in a calculation before they have been initialized. This is an error because uninitialized variables often contain unknown values, which will cause the result to be incorrect.
6. The assignment statement is not in the correct format. This is an error because all programming languages require that you write the name of the variable that is receiving the value on the left side of the = operator.
7. A named constant cannot be assigned a value with a `Set` statement. This is an error because the program attempts to change the value of a named constant.

©2016 Pearson Education, Inc. Hoboken, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Programming Exercise 2-1

```
Display "Jack Shepard"
Display "123 Lost Lane"
Display "Boston, MA  99999"
Display "704-555-1212"
Display "Computer Science"
```



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

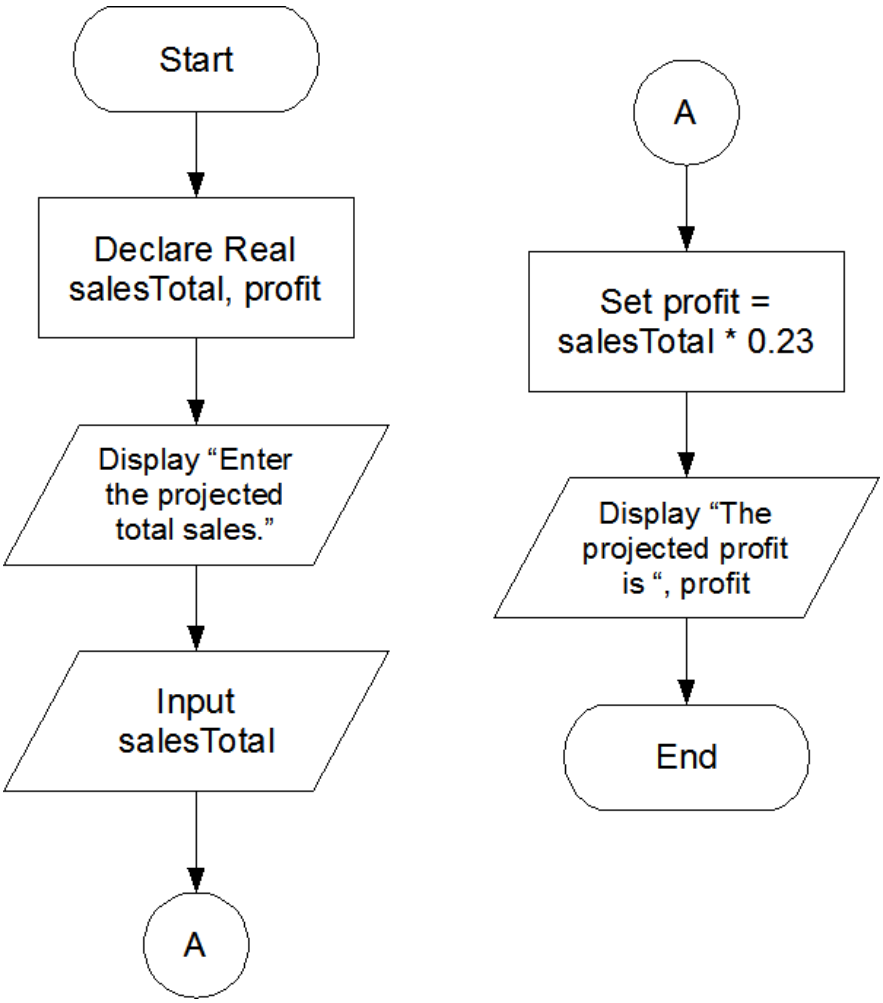
Programming Exercise 2-2

```
// Variables to hold the sales total and the profit
Declare Real salesTotal, profit

// Get the amount of projected sales.
Display "Enter the projected sales."
Input salesTotal

// Calculate the projected profit.
Set profit = salesTotal * 0.23

// Display the projected profit.
Display "The projected profit is ", profit
```



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-3

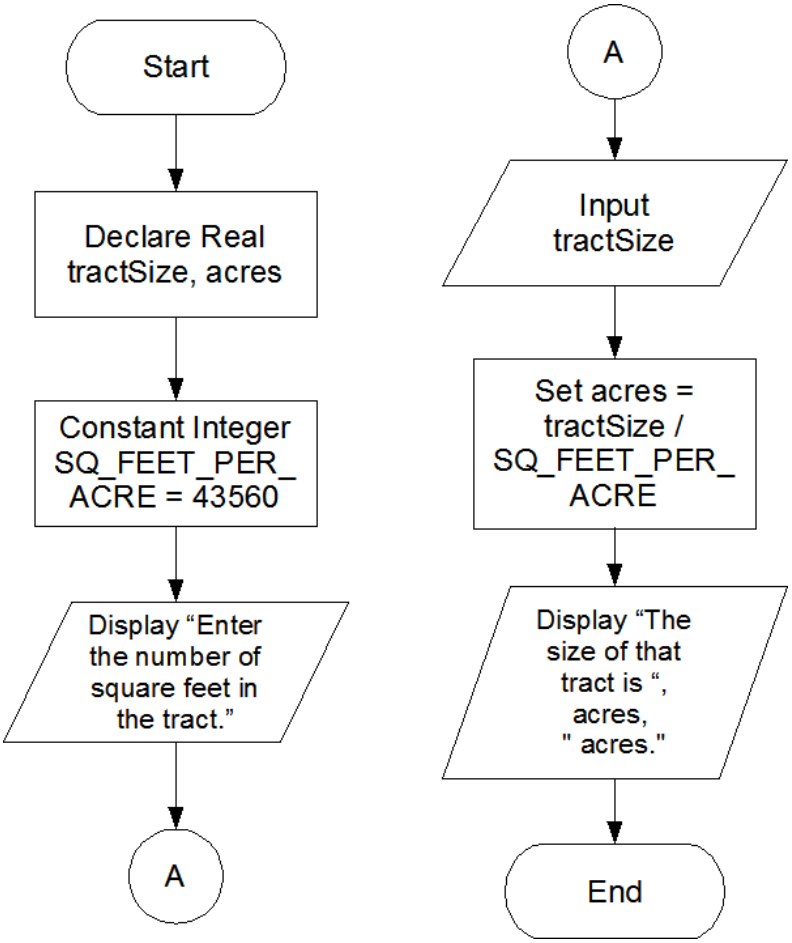
```
// Variables to hold the size of the tract and number of acres.
Declare Real tractSize, acres

// Constant for the number of square feet in an acre.
Constant Integer SQ_FEET_PER_ACRE = 43560

// Get the square feet in the tract.
Display "Enter the number of square feet in the tract."
Input tractSize

// Calculate the acreage.
Set acres = tractSize / SQ_FEET_PER_ACRE

// Display the number of acres.
Display "The size of that tract is ", acres, " acres."
```



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-4

```
// Variables to hold the prices of each item, the subtotal,
// and the total.
Declare Real item1, item2, item3, item4, item5,
             subtotal, tax, total

// Constant for the sales tax rate.
Constant Real TAX_RATE = 0.06

// Get the price of each item.
Display "Enter the price of item #1."
Input item1
Display "Enter the price of item #2."
Input item2
Display "Enter the price of item #3."
Input item3
Display "Enter the price of item #4."
Input item4
Display "Enter the price of item #5."
Input item5

// Calculate the subtotal.
Set subtotal = item1 + item2 + item3 + item4 + item5

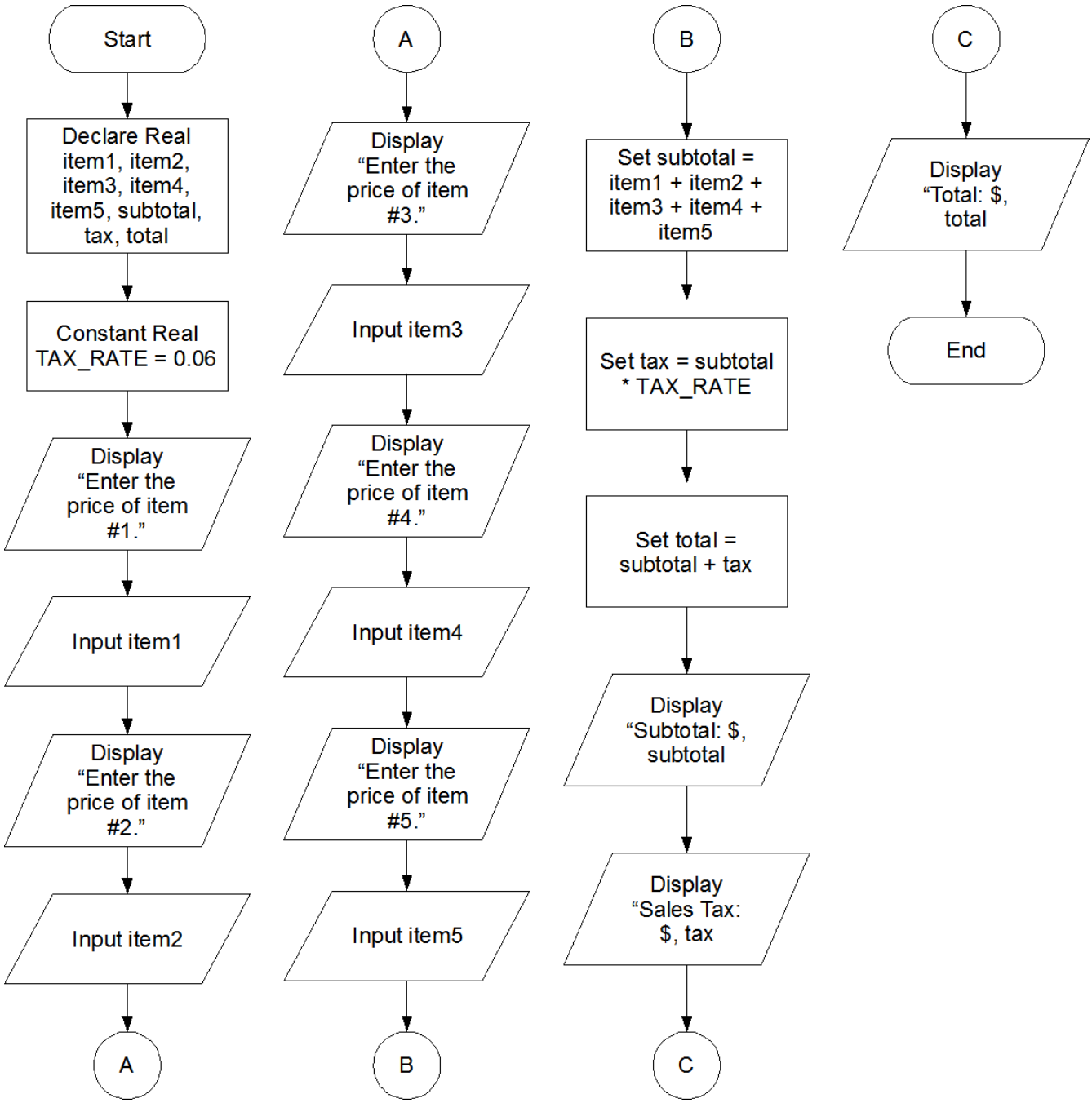
// Calculate the sales tax.
Set tax = subtotal * TAX_RATE

// Calculate the total.
Set total = subtotal + tax

// Display the values.
Display "Subtotal: $", subtotal
Display "Sales Tax: $", tax
Display "Total: $", total
```

©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-5

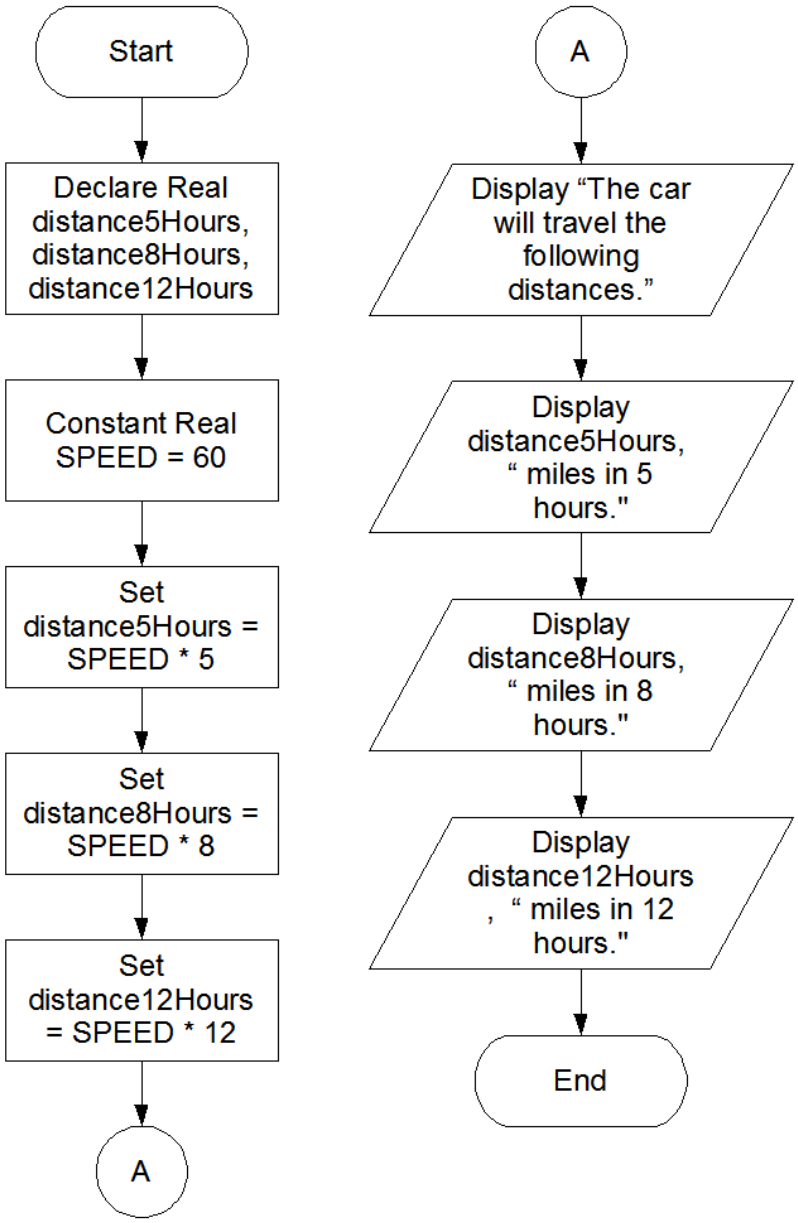
```
// Variables to hold the distances.
Declare Real distance5Hours, distance8Hours, distance12Hours

// Constant for the speed.
Constant Integer SPEED = 60

// Calculate the distance the car will travel in
// 5, 8, and 12 hours.
Set distance5Hours = SPEED * 5
Set distance8Hours = SPEED * 8
Set distance12Hours = SPEED * 12

// Display the results.
Display "The car will travel the following distances:"
Display distance5Hours, " miles in 5 hours."
Display distance8Hours, " miles in 8 hours."
Display distance12Hours, " miles in 12 hours."
```

©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-6

```
// Variable declarations
Declare Real purchase, stateTax, countyTax, totalTax, totalSale

// Constants for the state and county tax rates
Constant Real STATE_TAX_RATE = 0.04
Constant Real COUNTY_TAX_RATE = 0.02

// Get the amount of the purchase.
Display "Enter the amount of the purchase."
Input purchase

// Calculate the state sales tax.
Set stateTax = purchase * STATE_TAX_RATE

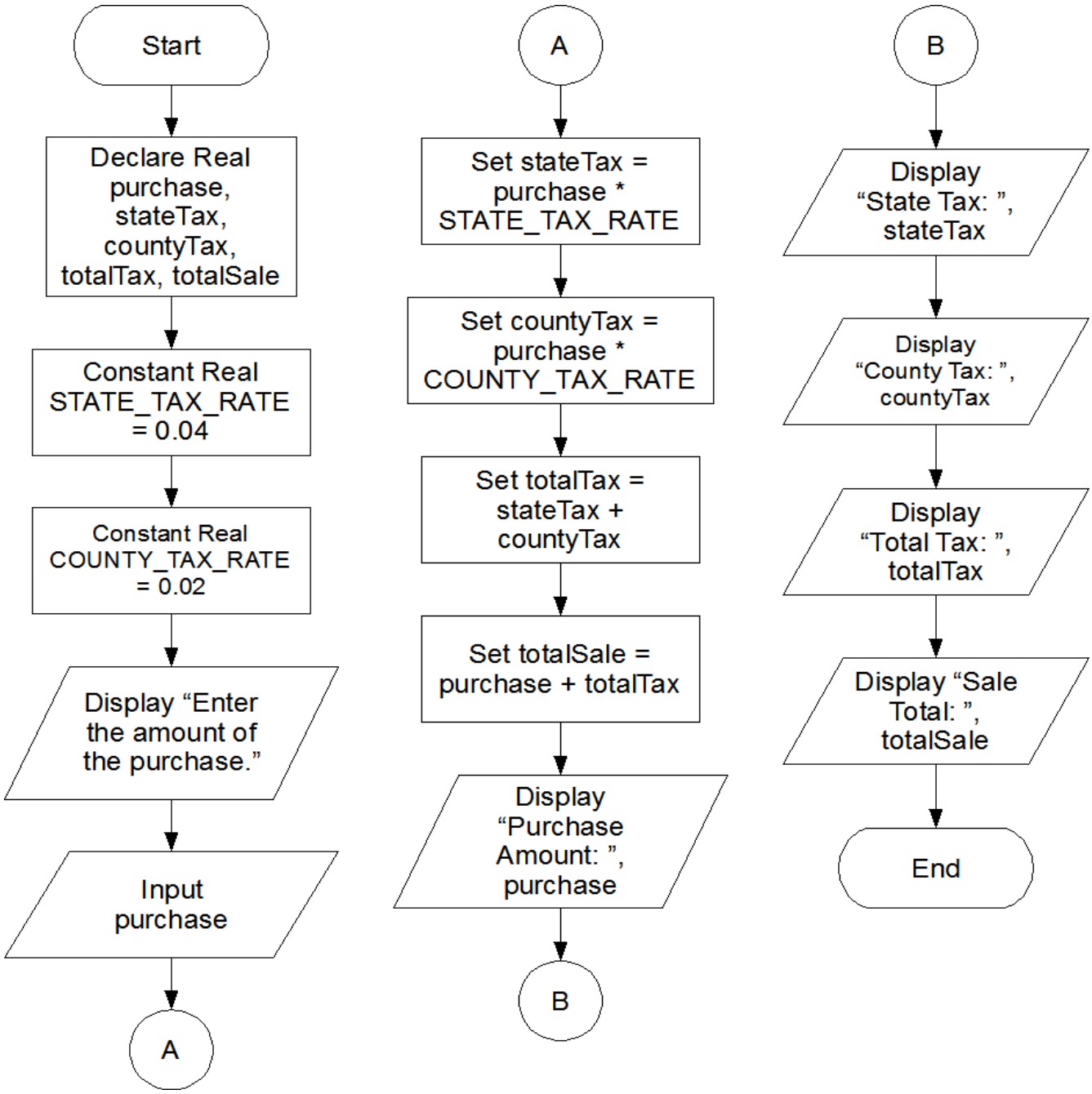
// Calculate the county sales tax.
Set countyTax = purchase * COUNTY_TAX_RATE

// Calculate the total tax.
Set totalTax = stateTax + countyTax

// Calculate the total of the sale.
Set totalSale = purchase + totalTax

// Display information about the sale.
Display "Purchase Amount: $", purchase
Display "State Tax: ", stateTax
Display "County Tax: ", countyTax
Display "Total Tax: ", totalTax
Display "Sale total: ", totalSale
```

©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-7

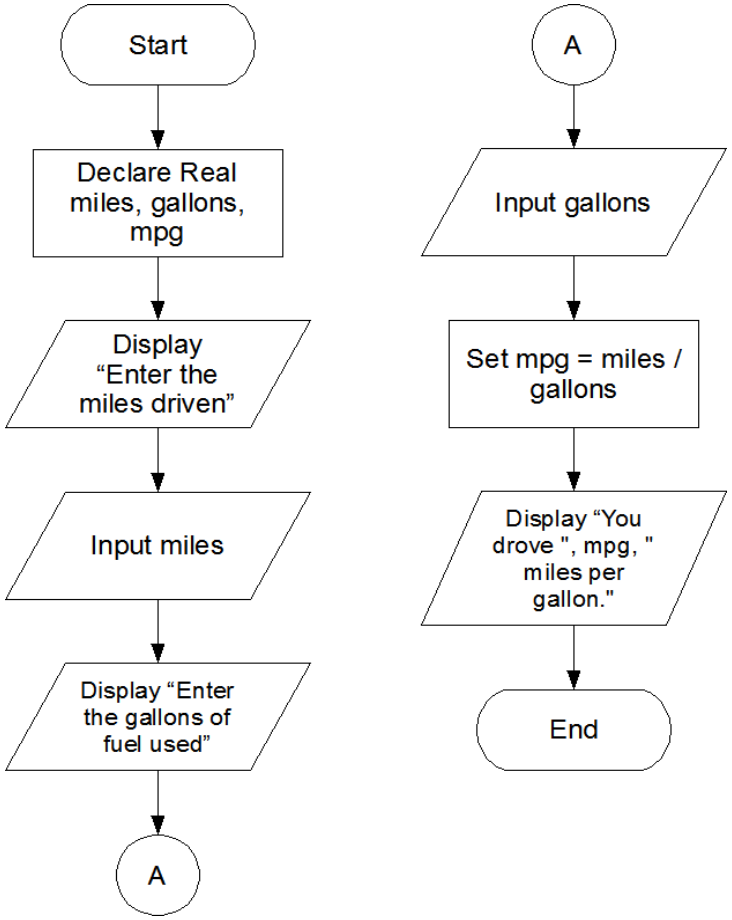
```
// Declare variables to hold miles driven, gallons
// of fuel used, and miles-per-gallon.
Declare Real miles, gallons, mpg

// Get the miles driven.
Display "Enter the miles driven."
Input miles

// Get the gallons of fuel used.
Display "Enter the gallons of fuel used."
Input gallons

// Calculate miles-per-gallon.
Set mpg = miles / gallons

// Display the result.
Display "You drove ", mpg, " miles per gallon."
```



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Programming Exercise 2-8

```
// Declare variables for food charges, tip, tax, and total.
Declare Real food, tip, tax, total

// Constants for the tax rate and tip rate.
Constant Real TAX_RATE = 0.07
Constant Real TIP_RATE = 0.15

// Get the food charges.
Display "Enter the charge for food."
Input food

// Calculate the tip.
Set tip = food * TIP_RATE

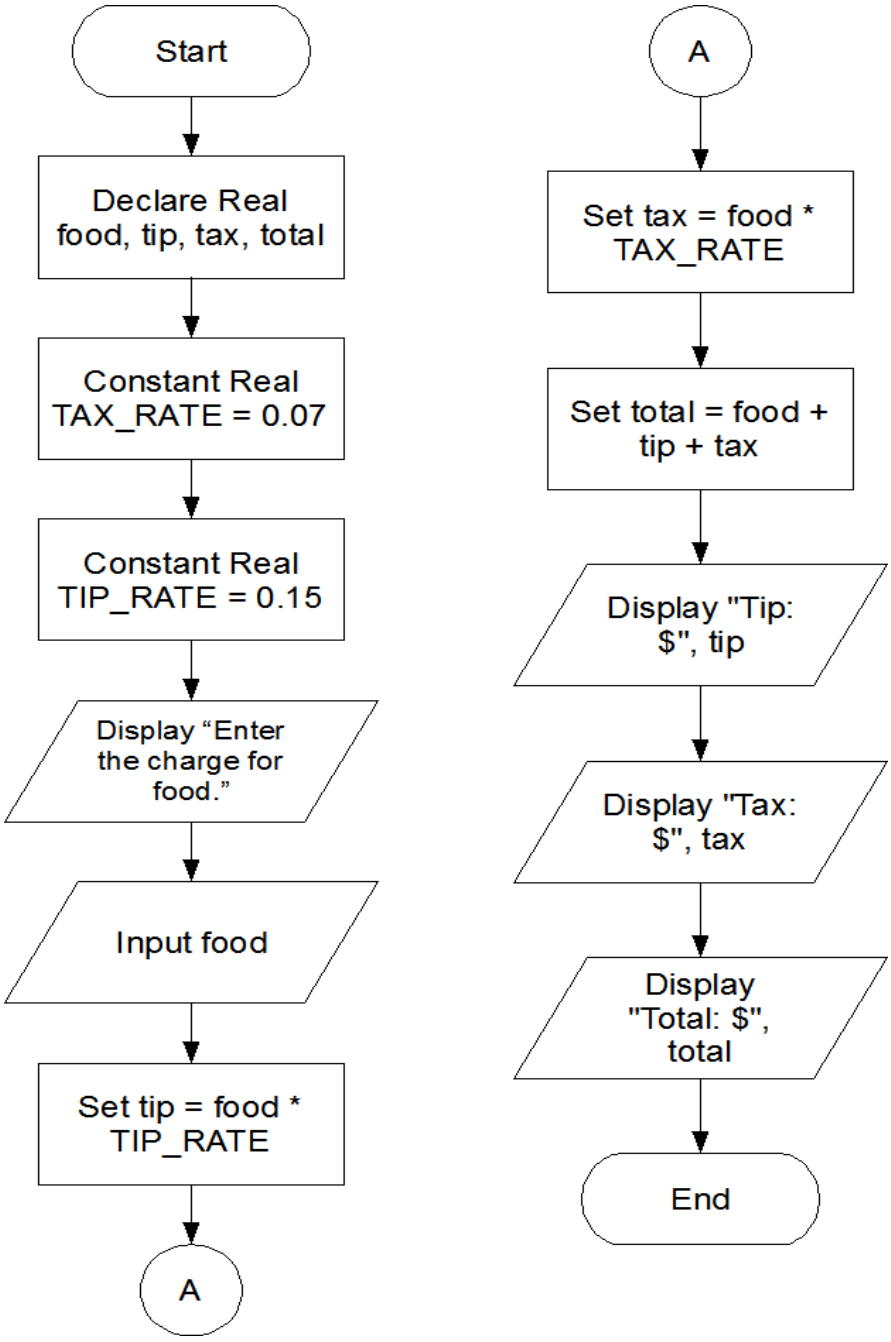
// Calculate the tax.
Set tax = food * TAX_RATE

// Calculate the total.
Set total = food + tip + tax

// Display the tip, tax, and total.
Display "Tip: $", tip
Display "Tax: $", tax
Display "Total: $", total
```

©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-9 (Weight Loss)

```
// Variable for the user's weight
Declare Real weight

// Constant for the monthly weight loss
Constant Real MONTHLY_WEIGHT_LOSS = 4

// Get the user's starting weight.
Display "Enter your starting weight."
Input weight

// Display weight at the end of month 1
Set weight = weight - MONTHLY_WEIGHT_LOSS
Display "Your weight at the end of month 1 is ", weight

// Display weight at the end of month 2
Set weight = weight - MONTHLY_WEIGHT_LOSS
Display "Your weight at the end of month 2 is ", weight

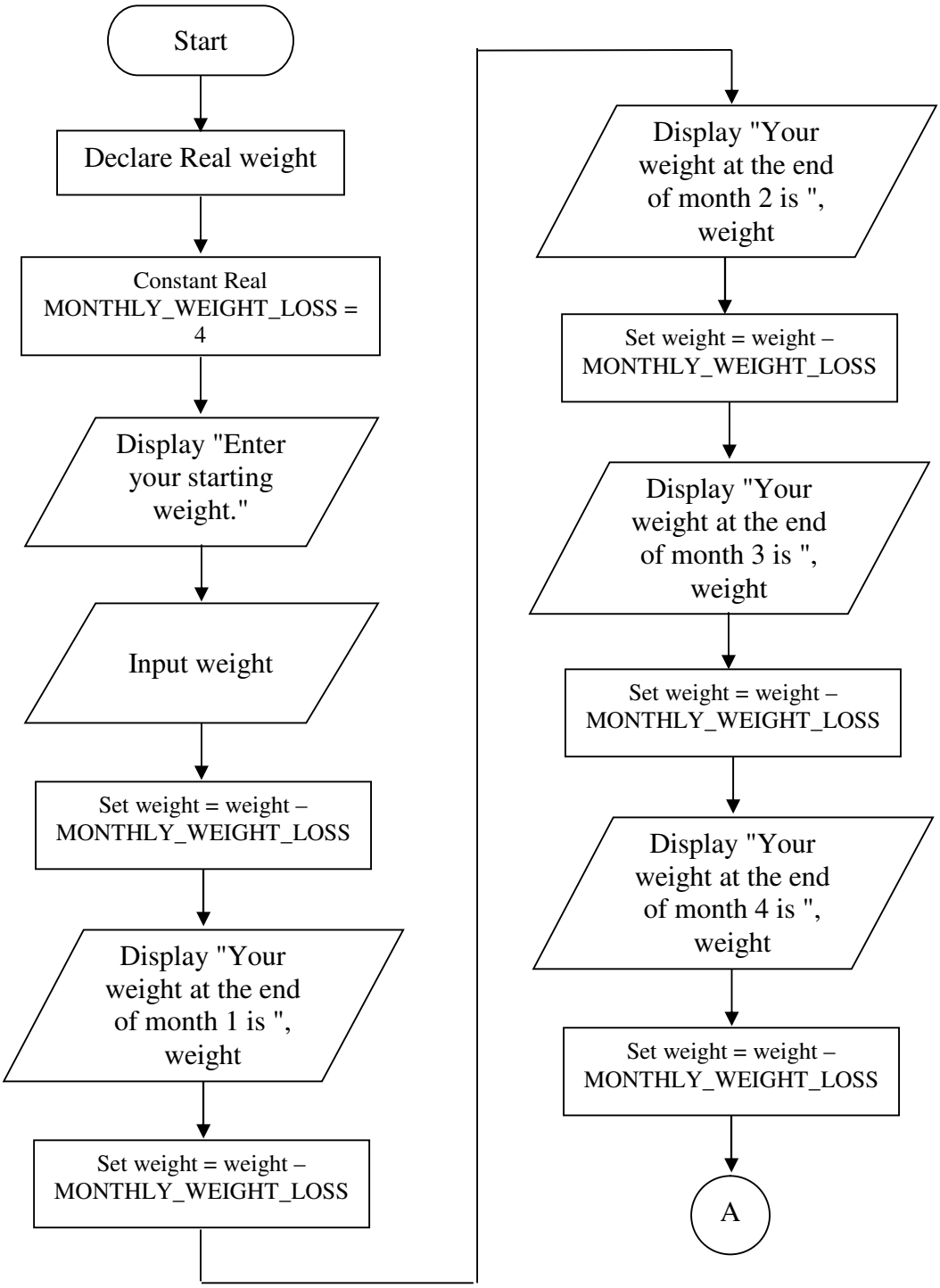
// Display weight at the end of month 3
Set weight = weight - MONTHLY_WEIGHT_LOSS
Display "Your weight at the end of month 3 is ", weight

// Display weight at the end of month 4
Set weight = weight - MONTHLY_WEIGHT_LOSS
Display "Your weight at the end of month 4 is ", weight

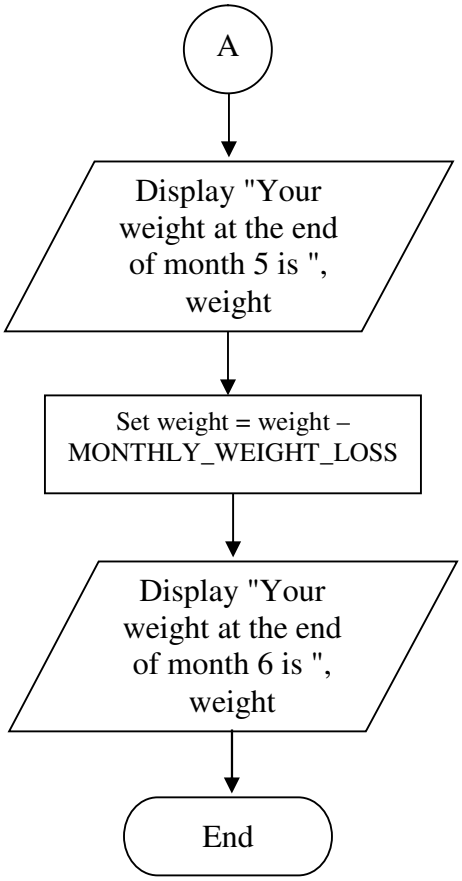
// Display weight at the end of month 5
Set weight = weight - MONTHLY_WEIGHT_LOSS
Display "Your weight at the end of month 5 is ", weight

// Display weight at the end of month 6
Set weight = weight - MONTHLY_WEIGHT_LOSS
Display "Your weight at the end of month 6 is ", weight
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Programming Exercise 2-10 (Amount Paid Over Time)

```
// Declare variables
Declare Real payment              // Monthly payment amount
Declare Real total                // Total payments
Declare Integer months            // Number of months

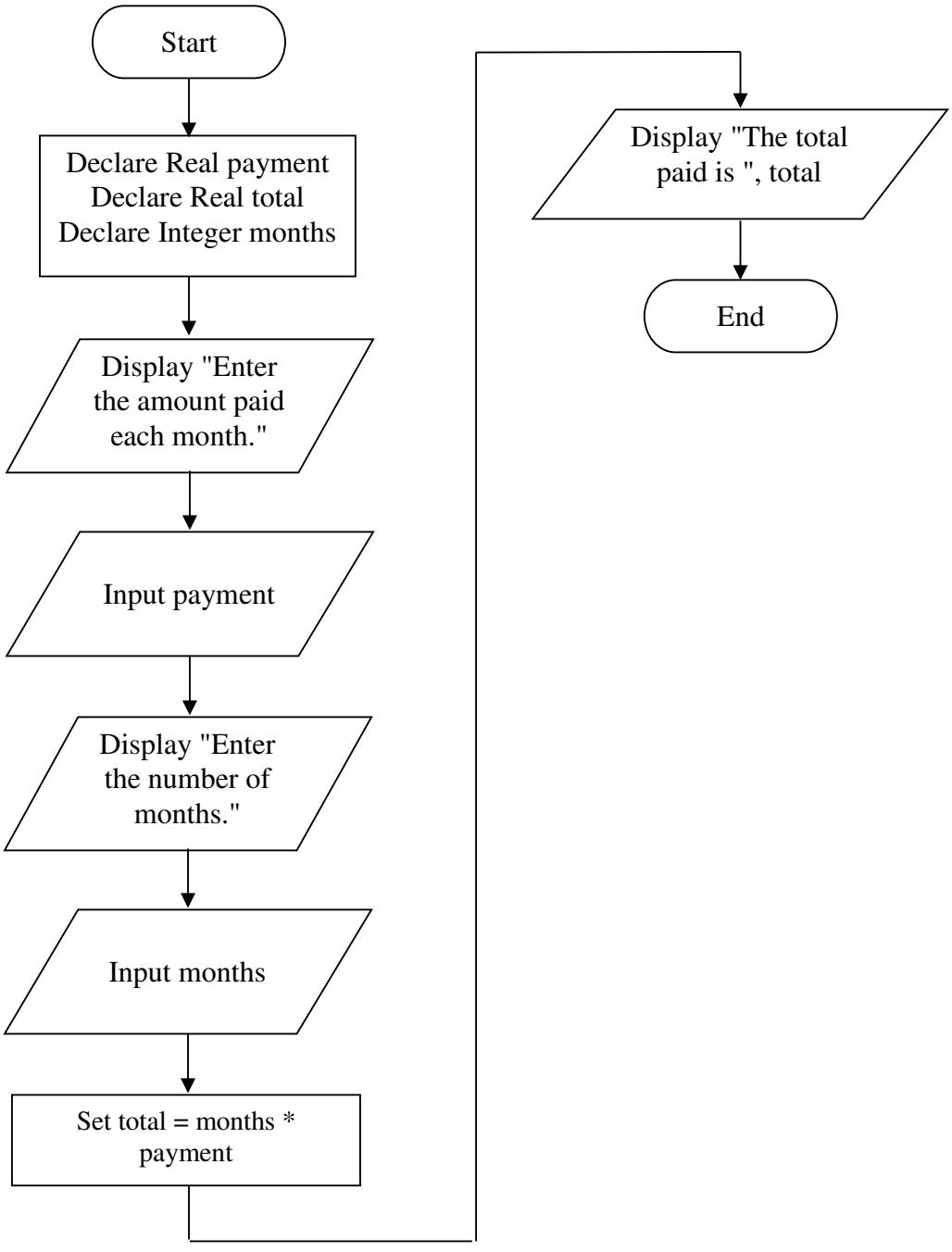
// Get the amount paid each month.
Display "Enter the amount paid each month."
Input payment

// Get the number of months
Display "Enter the number of months."
Input months

// Calculate the total of the payments.
Set total = months * payment

// Display the total.
Display "The total paid is ", total
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Programming Exercise 2-11 (Leftover Pizza)

```
// Declare variables
Declare Integer pizzas            // Number of pizzas
Declare Integer slicesPerPizza    // Slices per pizza
Declare Integer totalSlices       // Total number of slices
Declare Integer people            // Number of people
Declare Integer leftover          // Number of leftover slices

// Number of slices per person
Constant Integer SLICES_PER_PERSON = 3

// Get the number of pizzas.
Display "How many pizzas will you have?"
Input pizzas

// Get the number of slices per pizza.
Display "How many slices per pizza?"
Input slicesPerPizza

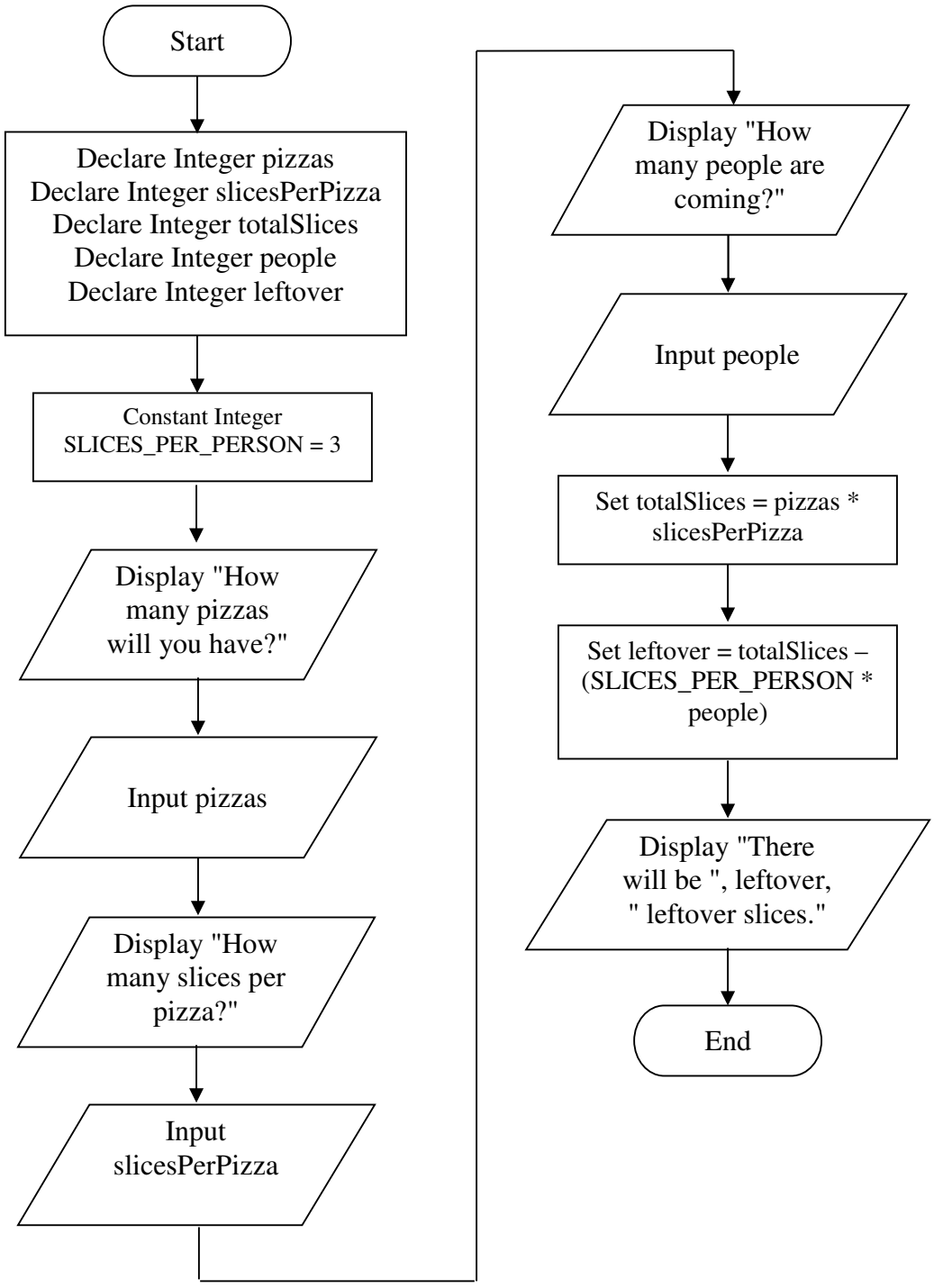
// Get the number of people.
Display "How many people are coming?"
Input people

// Calculate the total number of slices.
Set totalSlices = pizzas * slicesPerPizza

// Calculate the number of leftover slices.
Set leftover = totalSlices - (SLICES_PER_PERSON * people)

// Display the leftover slices
Display "There will be ", leftover, " leftover slices."
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

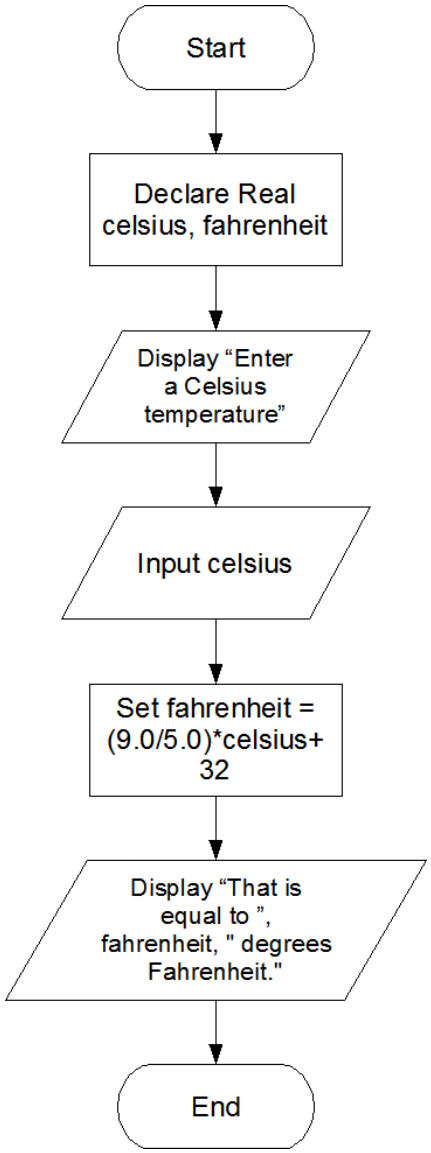
Programming Exercise 2-12 (Celsius to Fahrenheit Temperature Converter)

```
// Declare variables to hold the temperatures.
Declare Real celsius, fahrenheit

// Get the Celsius temperature.
Display "Enter a Celsius temperature."
Input celsius

// Calculate the Fahrenheit equivalent.
Set fahrehneit = (9.0 / 5.0) * celsius + 32

// Display the Fahrenheit temperature.
Display "That is equal to ", fahrenheit, " degrees Fahrenheit."
```



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-13 (Stock Transaction Program)

```
// Named constants
Constant Real COMMISSION_RATE = 0.02
Constant Integer NUM_SHARES = 1000
Constant Real PURCHASE_PRICE = 32.87
Constant Real SELLING_PRICE = 33.92

// Variables
Declare Real amountPaidForStock  // Amount paid for the stock
Declare Real purchaseCommission  // Commission paid to purchase stock
Declare Real totalPaid           // Total amount paid
Declare Real stockSoldFor        // Amount stock sold for
Declare Real sellingCommission   // Commission paid to sell stock
Declare Real totalReceived       // Total amount received
Delcare Real profitOrLoss        // Amount of profit or loss

// Calculate the amount that Joe paid for the stock, not
// including the commission.
Set amountPaidForStock = NUM_SHARES * PURCHASE_PRICE

// Calculate the amount of commission that Joe paid his broker
// when he bought the stock.
Set purchaseCommission = COMMISSION_RATE * amountPaid

// Calculate the total amount that Joe paid, which is the amount
// he paid for the stock plus the commission he paid his broker.
Set totalPaid = amountPaidForStock + purchaseCommission

// Calcualate the amount that Joe sold the stock for.
Set stockSoldFor = NUM_SHARES * SELLING_PRICE

// Calculate the amount of commission that Joe paid his broker
// when he sold the stock.
Set sellingCommission = COMMISSION_RATE * stockSoldFor

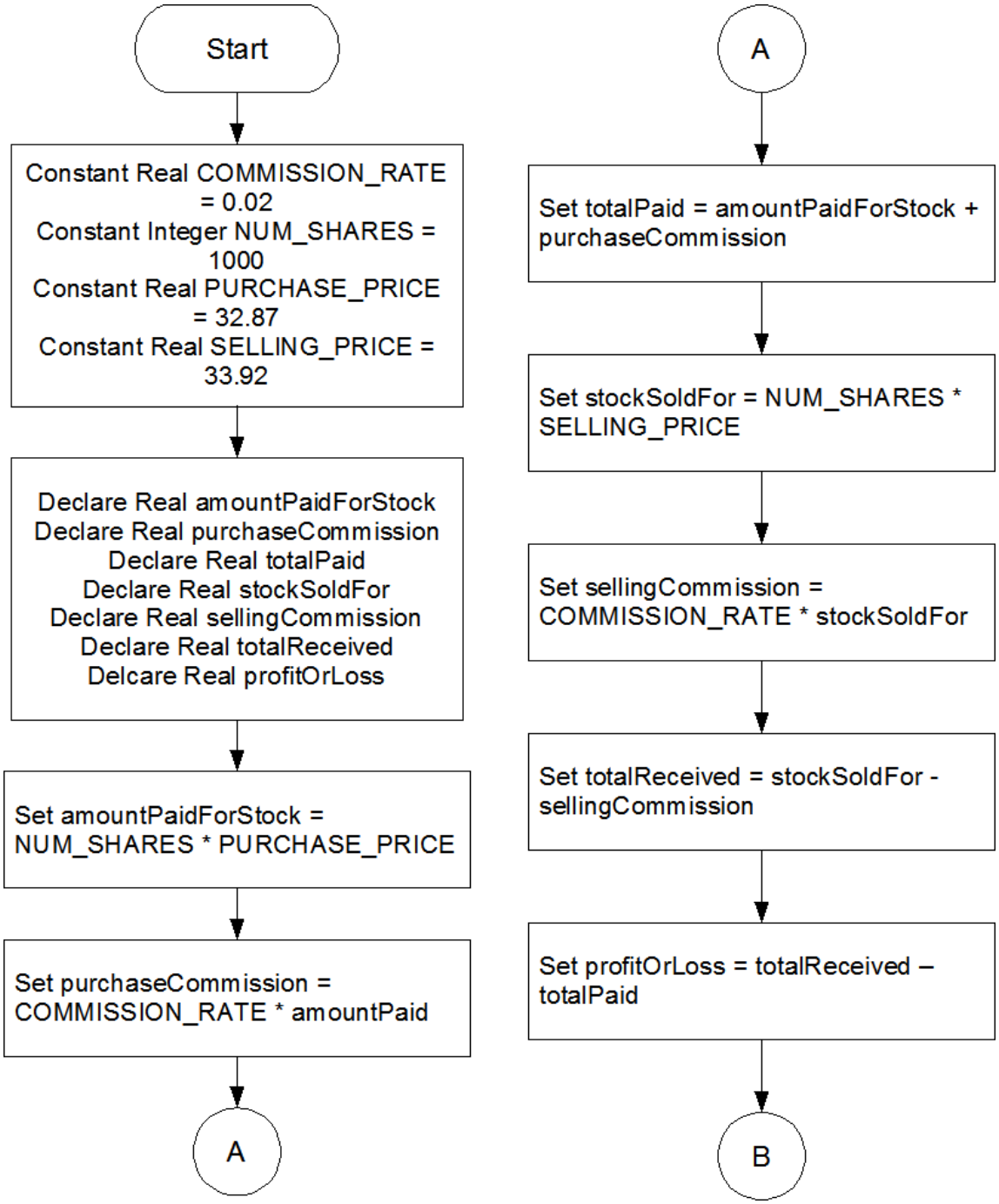
// Calculate the amount of money left over, after Joe paid
// his broker.
Set totalReceived = stockSoldFor - sellingCommission

// Calculate the amount of profit or loss. If this amount is a
// positive number, it is profit. If this is a negative number it
// is a loss.
Set profitOrLoss = totalReceived - totalPaid

// Display the required data.
Display "Amount paid for the stock: $", amountPaidForStock
Display "Commission paid on the purchase: $", purchaseCommission
Display "Amount the stock sold for: $", stockSoldFor
Display "Commission paid on the sale: $", sellingCommission
Display "Profit (or loss if negative): $", profitOrLoss
```

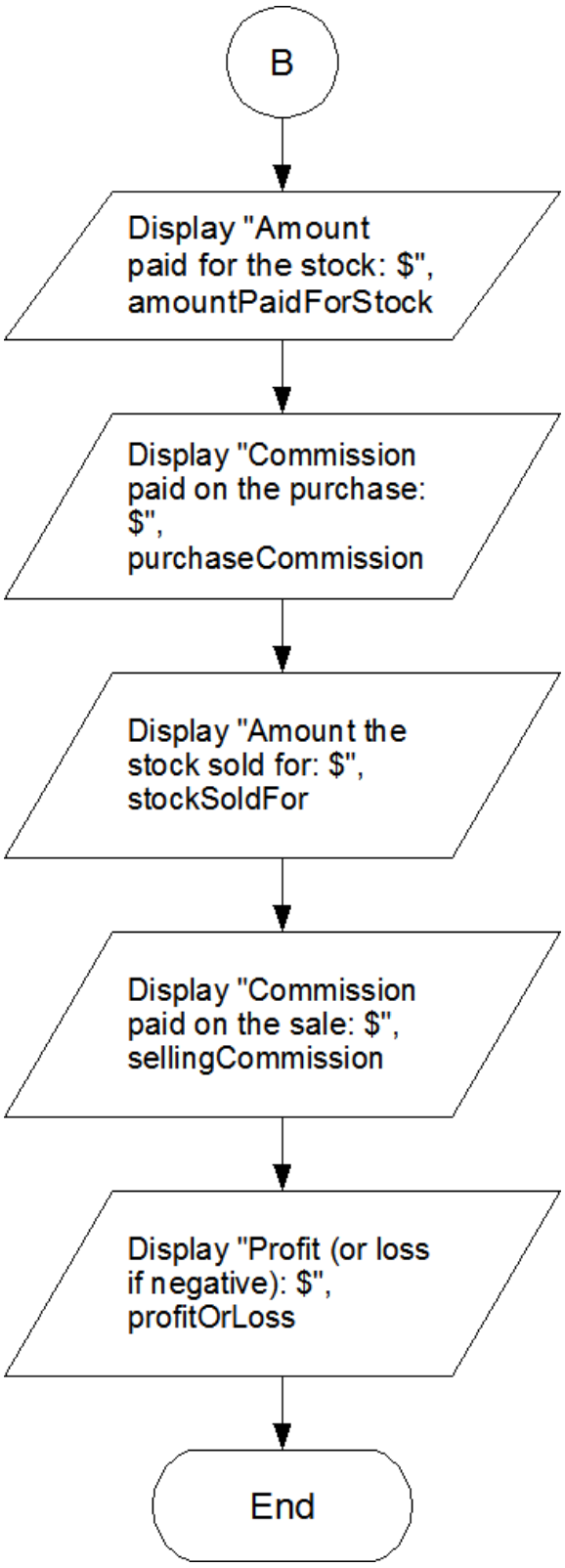
©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



©2013 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Programming Exercise 2-14
(Cookie Calories)

```
// Constant for the number of cookies per bag
Constant Integer COOKIES_PER_BAG = 40

// Constant for the number of servings per bag
Constant Integer SERVINGS_PER_BAG = 10

// Constant for the number of calories per serving
Constant Integer CALORIES_PER_SERVING = 300

// Constant for the number of cookies per serving
Constant Integer COOKIES_PER_SERVING =
                 COOKIES_PER_BAG / SERVINGS_PER_BAG

// Constant for the number of calories per cookie
Constant Integer CALORIES_PER_COOKIE =
                 CALORIES_PER_SERVING / COOKIES_PER_SERVING

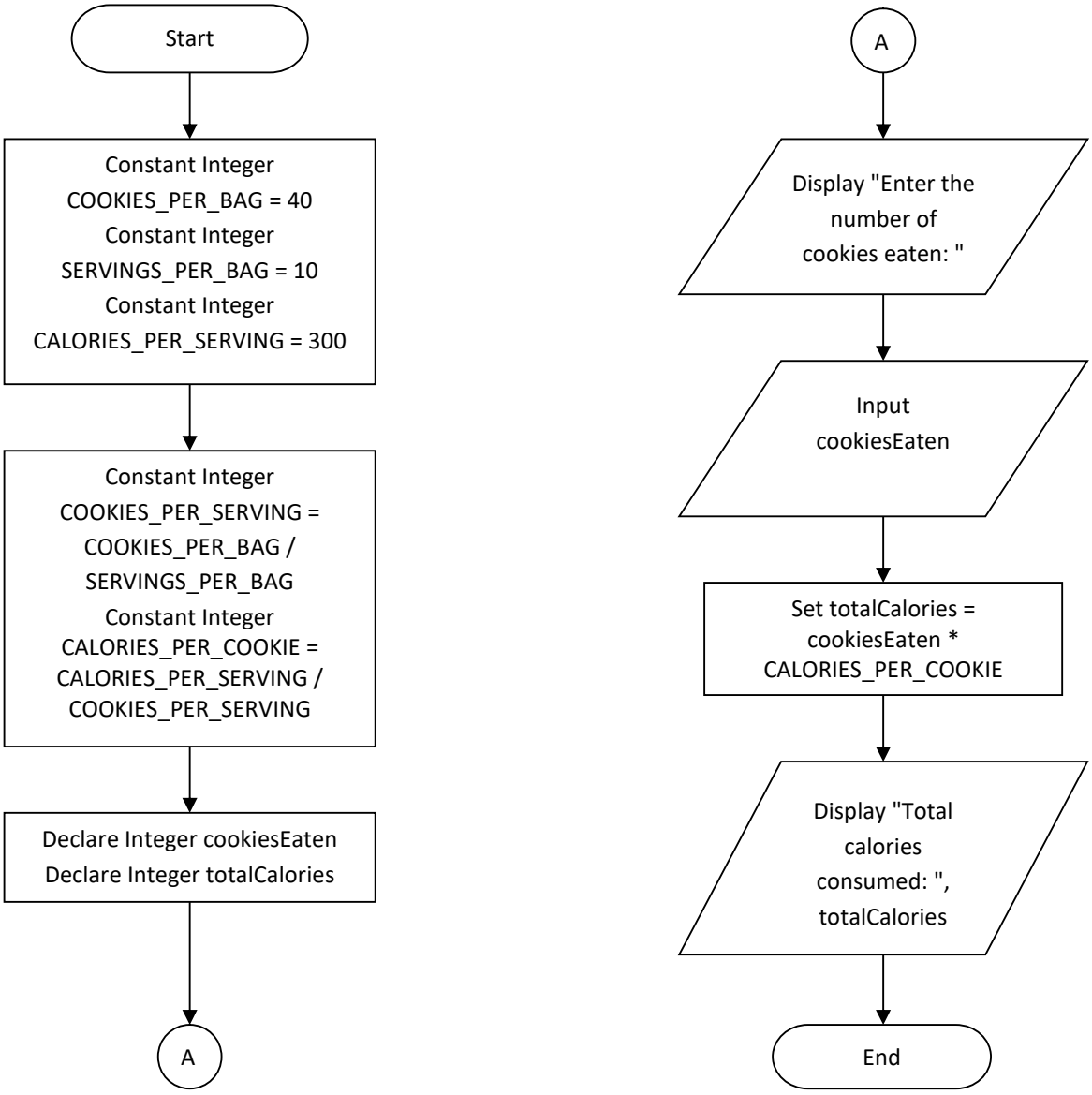
// Variables
Declare Integer cookiesEaten       // Cookies eaten
Declare Integer totalCalories      // Calories consumed

// Get the number of cookies eaten.
Display "Enter the number of cookies eaten: "
Input cookiesEaten

// Calculate the number of total calories consumed.
Set totalCalories = cookiesEaten * CALORIES_PER_COOKIE

// Display the number of total calories consumed.
Display "Total calories consumed: ", totalCalories
```

©2016 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.



©2016 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Programming Exercise 2-15 (Male and Female Percentages)

```
// Variables
Declare Integer male          // Number of male students
Declare Integer female        // Number of female students
Declare Real total            // Total number of students
Declare Real percentMale      // Percentage of male students
Declare Real percentFemale    // Percentage of female students

// Get the number of male students.
Display "Enter the number of male students: "
Input male

// Get the number of female students.
Display "Enter the number of female students: "
Input female

// Calculate the total number of students.
Set total = male + female

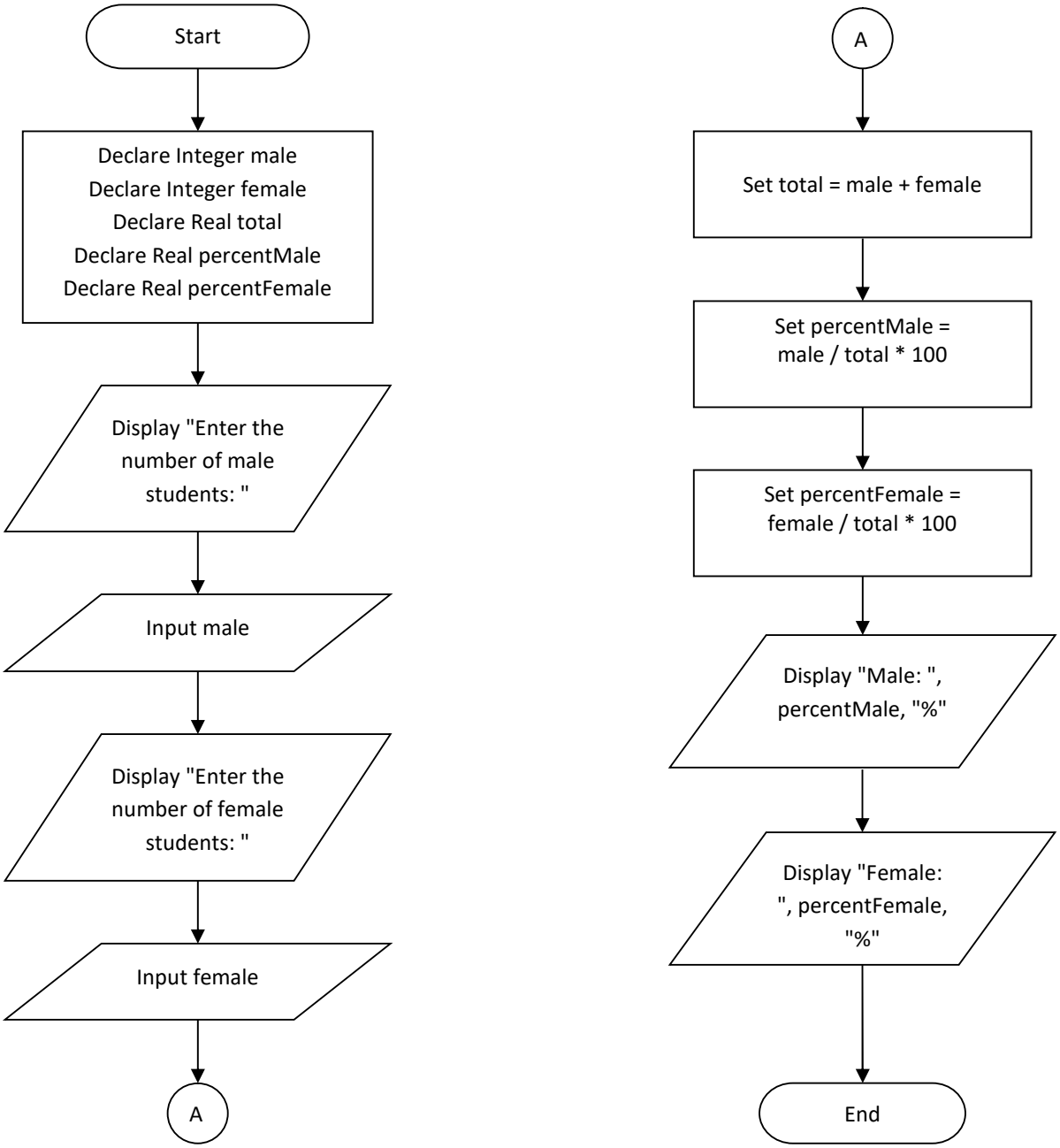
// Calculate the percentage of male students.
Set percentMale = male / total * 100

// Calculate the percentage of female students.
Set percentFemale = female / total * 100

// Display the percentage of male students.
Display "Male: ", percentMale, "%"

// Display the percentage of female students.
Display "Female: ", percentFemale, "%");
```

©2016 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.



©2016 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Programming Exercise 2-16 (Ingredient Adjuster)

```
// Named constants for the original recipe
Constant Integer COOKIES_RECIPE = 48    // Number of cookies
Constant Real SUGAR_RECIPE  = 1.5       // Cups of sugar
Constant Real BUTTER_RECIPE = 1.0       // Cups of butter
Constant Real FLOUR_RECIPE  = 2.75      // Cups of flour

// Variables for the adjusted recipe
Declare Integer cookies       // Adjusted number of cookies
Declare Real sugar            // Adjusted cups of sugar
Declare Real butter           // Adjusted cups of butter
Declare Real flour            // Adjusted cups of flour

// Get the adjusted number of cookies.
Display "Enter the number of cookies: "
Input cookies

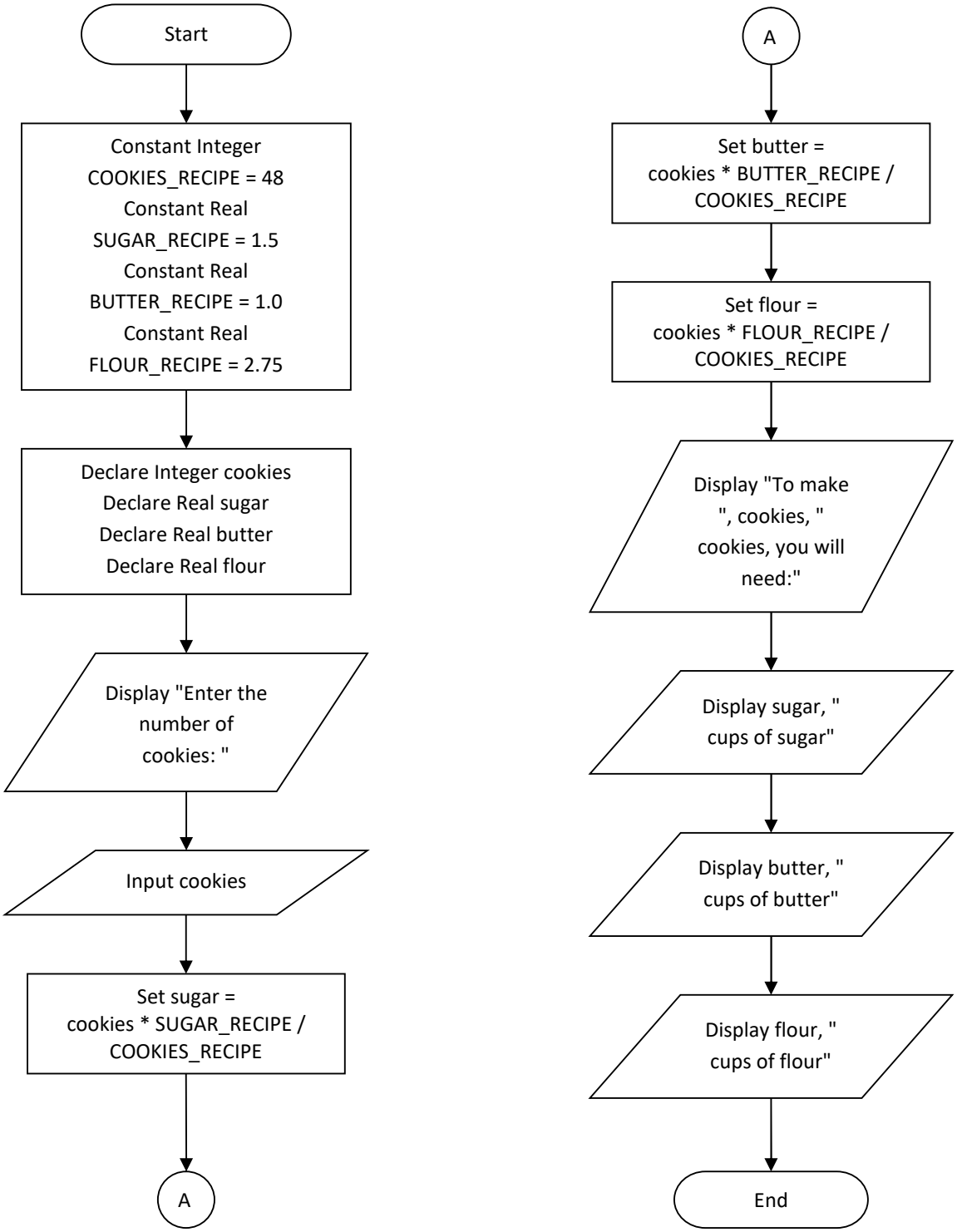
// Calculate the adjusted cups of sugar.
Set sugar = cookies * SUGAR_RECIPE / COOKIES_RECIPE

// Calculate the adjusted cups of butter.
Set butter = cookies * BUTTER_RECIPE / COOKIES_RECIPE

// Calculate the adjusted cups of flour.
Set flour = cookies * FLOUR_RECIPE / COOKIES_RECIPE

// Display the adjusted recipe amounts.
Display "To make ", cookies, " cookies, you will need:"
Display sugar, " cups of sugar"
Display butter, " cups of butter"
Display flour, " cups of flour"
```

©2016 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.



©2016 Pearson Education, Inc. Upper Saddle River, NJ. All Rights Reserved.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Solutions
Lab 2: Modules

Note to Instructor: This lab accompanies Chapter 3 of *Starting Out with Programming Logic & Design.* Material in the chapter should have been covered prior to lab assignment. In addition, students should have had instruction on using a flowcharting application such as Raptor and instruction on using the IDLE environment for Python.

Evaluation: The instructor should be present to answer any questions and observe the student performing the lab. The student should turn in a hard copy (paper) or soft copy (email) of their work. To minimize the number of attachments or individual files created for this lab, space is set aside in the lab for students to insert completed exercises. Directions are provided to students on copying and pasting.

Learning Objectives for this lab include:

1. Understand how modules work.
2. Know how to define and call a module.
3. Learn how to declare local variables.
4. Learn how to pass arguments to modules.

Labs 2.1, 2.2, 2.3, and 2.7 use the following programming problem.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

Labs 2.4 and 2.5 walk students through steps so they can become familiar with Python code using modules and variables.

Lab 2.6 uses the following programming problem.

Write a program that will calculate a 20% tip and a 6% tax on a meal price. The user will enter the meal price and the program will calculate tip, tax, and the total. The total is the meal price plus the tip plus the tax.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.1 – Algorithms

Step 2: Given a total sales of $27,097, answer the following:
What is the calculated state tax? 1083.88
What is the calculated county tax? 541.94
What is the calculated total tax? 1625.82

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.2 –Pseudocode and Modules

Step 1: This program is most easily solved using just four variables. Declare the variables that you will need in the program, using the proper data type and documenting the purpose.

Variable Name	Purpose
`Declare Real totalSales`	Stores total sales the user inputs
`Declare Real countyTax`	Stores the calculated county tax
`Declare Real stateTax`	Stores the calculated state tax
`Declare Real totalTax`	Stores the calculated total tax

Step 2: Given the major task involved in this program, what modules might you consider including? Describe the purpose of each module. (Reference: Defining and Calling a Module, page 106).

Module Name	Purpose
`Module inputData()`	Allows the user to enter required user input
`Module calcCounty()`	Calculates the county tax at 2% of the monthly sales
`Module calcState()`	Calculates the state tax at 4% of the monthly sales
`Module calcTotal()`	Calculates the total tax as state plus county
`Module printData()`	Prints the necessary information

Step 3: Complete the pseudocode by writing the missing lines. (Reference: Defining and Calling a Module, page 106). Also, when writing your modules and making calls, be sure to pass necessary variables as arguments and accept them as reference parameters if they need to be modified in the module. (Reference: Passing Arguments by Value and by Reference, page 127).

```
Module main ()
   // Declare local variables
   Declare Real totalSales
   Declare Real countyTax
   Declare Real stateTax
   Declare Real totalTax

   // Function calls
   Call inputData(totalSales)
   Call calcCounty(totalSales)
   Call calcState(totalSales)
   Call calcTotal(countyTax, stateTax, totalTax)
   Call printData(countyTax, stateTax, totalTax)
End Module

// this module takes in the required user input
Module inputData(Real Ref totalSales)
   Display "Enter the total sales for the month."
   Input totalSales
End Module
```

```
// this module calculates county tax
// totalSales can be a value parameter because it is not
// changed in the module.
// countyTax must be a reference parameter because it is
// changed in the module
Module calcCounty(Real totalSales, Real Ref countyTax)
   countyTax = totalSales * .02
End Module

// this module calculates state tax
Module calcState(Real totalSales, Real Ref stateTax)
   stateTAx = totalSales * .04
End Module

// this module calculates total tax
Module calcTotal(Real countyTax, Real stateTax, Real Ref totalTax)
   totalTax = countyTax + stateTax
End Module

// this module prints the requirements of the program
Module printData(Real countyTax, Real stateTax, Real totalTax)
   Display "The county tax is ", countyTax
   Display "The state tax is ", stateTax
   Display "The total tax is ", totalTax
End Module
```

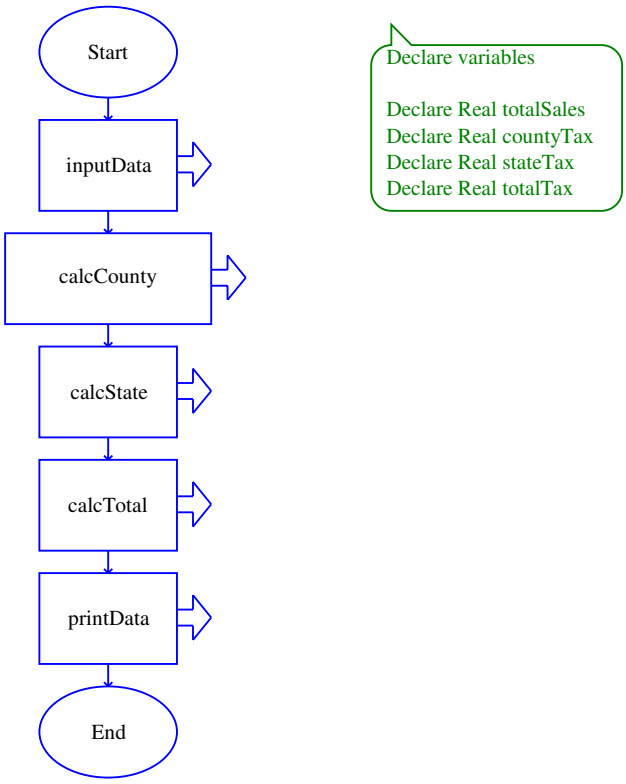
Lab 2.3 –Flowcharts

Note to Instructor:
Steps 1 – 6 walk students through the process of using Raptor and designing a flowchart. Step 7 requires students to show their completed flowchart. Variable names and statements will vary from student to student. Also, the lab can be completed using a different flowcharting application such as Visio. If so, the sample solution below will look a little different, but should still have the same fundamental steps.

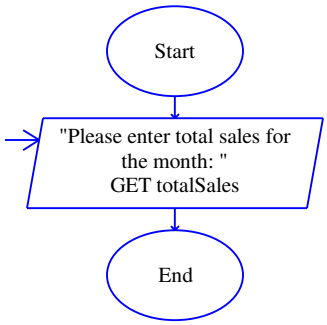
Instructors should grade on clear variable names, proper flow of program, the correct calculation, and the right output. Below is a sample solution to the programming problem.

Step 7:

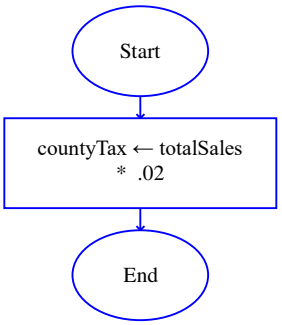
Main Module



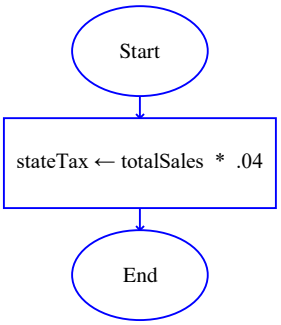
inputData Module



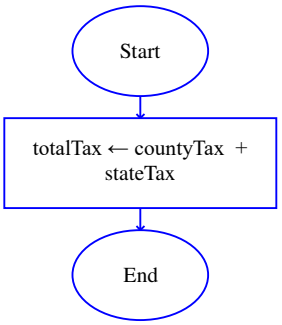
calcCounty Module



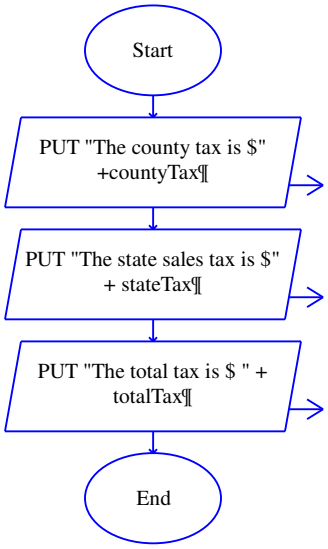
calcState Module



calcTotal Module



printData Module



Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.4 – Python Code and Functions

Note to Instructor:
Steps 1 – 7 in this lab require students to write Python code to create a program. Their final program is required in Step 8.

Step 8:

```
# Student Name
# Date
# Description:  This program tests out the different ways
#               to use functions

# The main function
def main():
    welcome_message()  # causes welcome_message to execute
    goodbye_message()  # causes goodbye_message to execute

# This function is to welcome people to my program
def welcome_message():
    print('Welcome to my program using functions')
    print('My name is Joe Student')

def goodbye_message():
    print('Goodbye...thank you for using my program')

# This is the main function that starts the program in motion
main()   # calls main
```

Lab 2.5 – Python Code and Variables

Note to Instructor:
Steps 1 – 7 in this lab require students to write Python code to create a program. Their final program is required in Step 8.

Step 8:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the variable program')
    print()   #prints a blank line

    name = inputName()
    age = inputAge()
    print('Hello', name)
    print('Your age is', age)

# this function inputs name
def inputName():
    name = input('Enter your name: ')
    return name

# this function inputs age
def inputAge():
    age = int(input('Enter your age: '))
    return age

# calls main
main()
```

Lab 2.6 – Writing a Complete Program

Note to Instructor:
Steps 1 – 15 in this lab walk students through an entire program. Step 16 requires them to paste their entire program.

Step 16:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the tip and tax calculator program')
    print()   # prints a blank line
    mealprice = input_meal()
    tip = calc_tip(mealprice)
    tax = calc_tax(mealprice)
    total = calc_total(mealprice, tip, tax)
    print_info(mealprice, tip, tax, total)


# this function will input meal price
def input_meal():
    mealprice = input('Enter the meal price $')
    mealprice = float(mealprice)
    return mealprice


# this function will calculate tip at 20%
def calc_tip(mealprice):
    tip = mealprice * .20
    return tip

# this function will calculate tax at 6%
def calc_tax(mealprice):
    tax = mealprice * .06
    return tax

# this function will calculate tip, tax, and the total cost
def calc_total(mealprice, tip, tax):
    total = mealprice + tip + tax
    return total

# this function will print tip, tax, the mealprice, and the
total
def print_info(mealprice, tip, tax, total):
    print('The meal price is $', mealprice)
```

```
    print('The tip is $', tip)
    print('The tax is $', tax)
    print('The total is $', total)

# calls main
main()
```

Lab 2.7 – Programming Challenge 1 – Retail Tax

Note to Instructor:
This lab requires students to translate their work in the pseudocode and flowchart from Lab 2.2 and Lab 2.3 to actual code using Python. The final program may look as follows:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the total tax calculator program')
    print()   # prints a blank line
    totalsales = inputData()
    countytax = calcCounty(totalsales)
    statetax = calcState(totalsales)
    totaltax = calcTotal(countytax, statetax)
    printData(countytax, statetax, totaltax)

# this function will input meal price
def inputData():
    totalsales = input('Enter the total sales for the month $')
    totalsales = float(totalsales)
    return totalsales

# this function will calculate tip at 20%
def calcCounty(totalsales):
    countytax = totalsales * .02
    return countytax

# this function will calculate tax at 6%
def calcState(totalsales):
    statetax = totalsales * .04
    return statetax

# this function will calculate tip, tax, and the total cost
def calcTotal(countytax, statetax):
    totaltax = countytax + statetax
    return totaltax

# this function will print tip, tax, the mealprice, and the total
def printData(countytax, statetax, totaltax):
    print('The county tax is $', countytax)
    print('The state tax is $', statetax)
    print('The total tax is $', totaltax)

# calls main
main()
```

Lab 2: Modules

This lab accompanies Chapter 3 of *Starting Out with Programming Logic & Design*.

Name: _____

Lab 2.1 – Algorithms

This lab requires you to think about the steps that take place in a program by writing algorithms. Read the following program prior to completing the lab.

> A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:
>
> - The amount of county sales tax
> - The amount of state sales tax
> - The total sales tax (county plus state)

Step 1: Examine the following algorithm.

1. Get the total sales for the month.
2. Multiply the total sales by .04 to calculate the state sales tax.
3. Multiply the total sales by .02 to calculate the county sales tax.
4. Add the state tax and county tax to calculate the total sales tax.
5. Display the calculated county tax, state tax, and total sales tax.

Step 2: Given a total sales of $27,097, answer the following:

What is the calculated state tax? _____

What is the calculated county tax? _____

What is the calculated total tax? _____

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.2 – Pseudocode and Modules

Critical Review

A Module is a group of statements that exists within a program for the purpose of performing a specific task.

Modules are commonly called procedures, subroutines, subprograms, methods, and functions.

The code for a module is known as a module definition. To execute the module, you write a statement that calls it.

The format for a module definition is as follows:

```
Module name()
   Statement
   Statement
   Etc.
End Module
```

Calling a module is normally done from the `main()` module such as:

```
Call name()
```

Generally, local variables should be used and arguments should be passed by reference when the value of the variable is changed in the module and needs to be retained. For example:

```
Module main()
   Real Integer number
   Call inputData(number)
   Call printData(number)
End Module

// accepts number as a reference so the changed value
// will be retained
Module inputData(Real Ref number)
   Number = 20
End Module

// number does not to be sent as a reference because
// number is not going to be modified
Module printData(number)
   Display "The number is ", number
End Module
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

This lab requires you to think about the steps that take place in a program by writing pseudocode. Read the following program prior to completing the lab.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

Step 1: This program is most easily solved using just four variables. Declare the variables that you will need in the program, using the proper data type and documenting the purpose.

Variable Name	**Purpose**
Declare Real totalSales	Stores total sales the user inputs

Step 2: Given the major task involved in this program, what modules might you consider including? Describe the purpose of each module. (Reference: Defining and Calling a Module, page 106).

Module Name	**Purpose**
Module inputData ()	Allows the user to enter required user input

Step 3: Complete the pseudocode by writing the missing lines. (Reference: Defining and Calling a Module, page 106). Also, when writing your modules and making calls, be sure to pass necessary variables as arguments and accept them as reference parameters if they need to be modified in the module. (Reference: Passing Arguments by Value and by Reference, page 127).

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

```
Module main ()
   // Declare local variables
   Declare Real totalSales
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________

   // Function calls
   Call inputData(totalSales)
   Call calcCounty(totalSales, countyTax)
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
End Module

// this module takes in the required user input
Module inputData(Real Ref totalSales)
   Display "Enter the total sales for the month."
   Input totalSales
End Module

// this module calculates county tax
// totalSales can be a value parameter because it is not
// changed in the module.
// countyTax must be a reference parameter because it is
// changed in the module
Module calcCounty(Real totalSales, Real Ref countyTax)
   countyTax = totalSales * .02
End Module

// this module calculates state tax
Module _____________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
End Module

// this module calculates total tax
Module _____________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
End Module

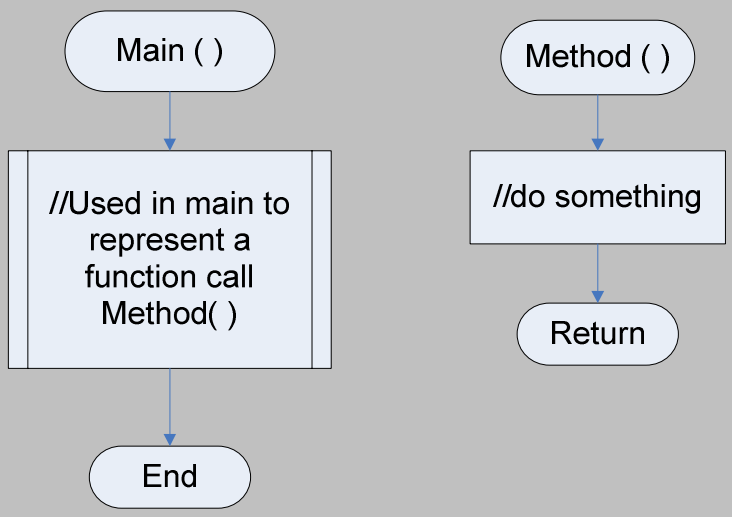
// this module prints the total, county, and state tax
Module _____________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
   _____________________________________________________________
End Module
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.3 – Flowcharts

Critical Review

The flowchart symbol used for a function call is a rectangle with vertical bars on each side:



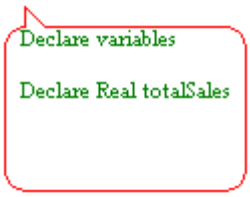
This lab requires you to think about the steps that take place in a program by designing a flowchart. Use an application such as Raptor or Visio. Read the following program prior to completing the lab.

```
A retail company must file a monthly sales tax report
listing the total sales for the month and the amount of
state and county sales tax collected. The state sales tax
rate is 4 percent and the county sales tax rate is 2
percent. Write a program that asks the user to enter the
total sales for the month. The application should calculate
and display the following:
```

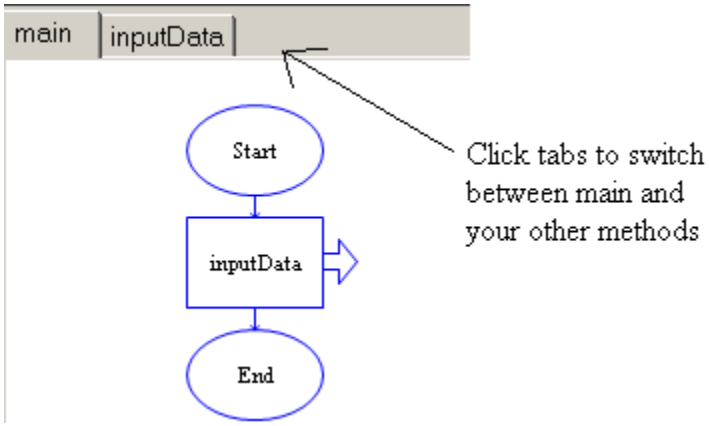
- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

Step 1: Start Raptor and save your document as *Lab 2-3*. The *.rap* file extension will be added automatically. Start by adding a Comment box that declares your variables. Here is a start to how your Comment box should look.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

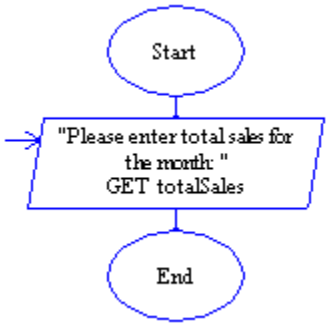


Step 2: The next step in your flowchart should be to call your methods. Click the Call Symbol on the Left and Drag and Drop to the flow lines between Start and Stop. Double click on the Call Symbol and type the name of your first method. For example, type *inputData* in the Enter Call box. Do not put the () when using Raptor. Click the Done button. A new box will pop up that will ask you to create a new tab. Click Yes. A new tab will be created for your new method. Notice the new Tab called *inputData.*

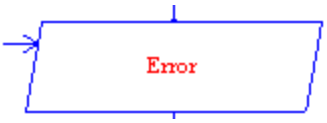


Step 3: Continue this process to add your additional methods, which are calcCounty() calcState(), calcTotal() and printData().

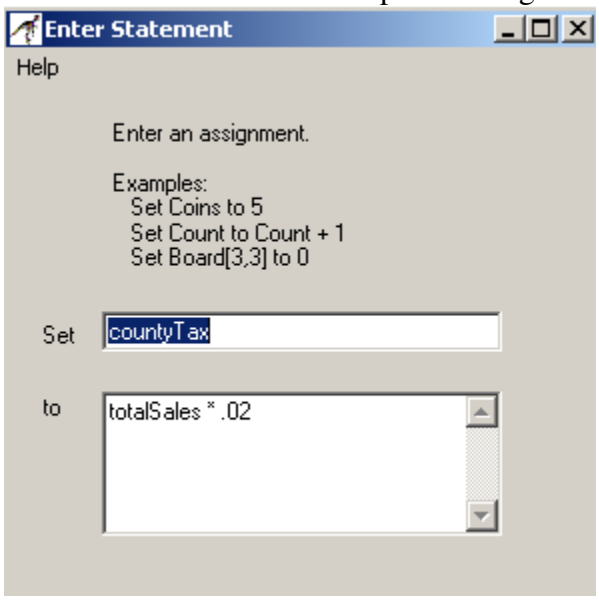
Step 4: Click on the inputData tab and add the necessary code as identified in your pseudocode in *Lab 2.2.* In Raptor, there is no need to pass variables as References as in pseudocode. Your inputData method might look like the following:



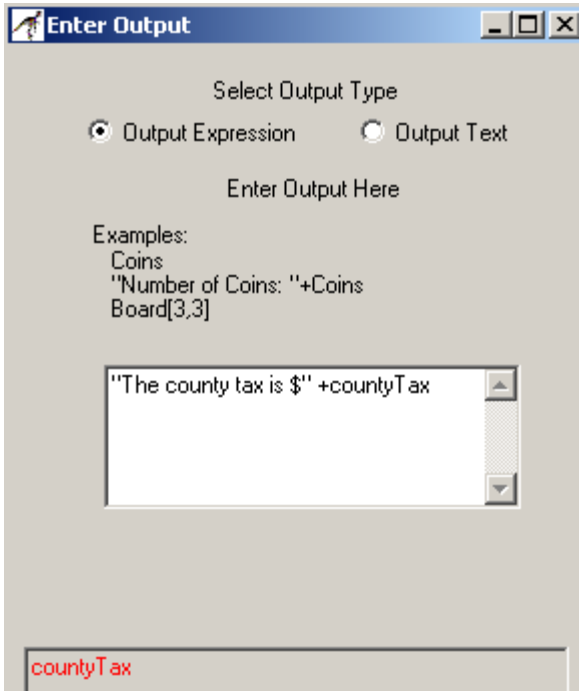
Step 5: Continue to code the remaining methods, which are calcCounty() calcState(), calcTotal() and printData(). If you happened to execute your program without completing your program, an error will occur such as:



Your calculations methods input box might look like the following:



Your output data methods box might look like the following:



Step 6: After your program is complete, click Run, then Execute to Finish. For your input, enter 67854 as your total monthly sales. If your program is coded correctly, the output should be as follows:

The county tax is $1357.0800
The state sales tax is $2714.1600
The total tax is $ 4071.2400
----Run finished----

Step 7: The final step is to insert your finished flowchart in the space below. Inside Raptor, select File and the Print to Clipboard from the menu. Inside Word in the space below, select Edit and Paste. You will have to do this for each module you created.

PASTE FLOWCHART HERE

Lab 2.4 – Python Code and Functions

Critical Review

The code for a function is known as a function definition. To execute the function, write a statement that calls it.

To create a function, write its definition. The keyword *def* is used before a function name, followed by parentheses and a colon. Here is the general format of a function definition in Python:

```
def function_name():
    statement
    statement
    etc.
```

Calling a function is done in order to make the module execute. The general format is:

```
function_name()
```

Function names must be flushed to the left.

Statements within a module must be aligned evenly in order to avoid syntax errors.

Step 1: Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab2-4.py*. Be sure to include the .py extension.

Step 2: Document the first few lines of your program to include your name, the date, and a brief description of what the program does. Description of the program should be:

```
# This program will demonstrate various ways to
# use functions in Python.
```

Step 3: After your documentation, add the following function definition and function call.

```
# This function is to welcome people to my program
def welcome_message():
    print('Welcome to my program using functions')
    print('My name is Joe Student')

# This is a function call
welcome_message()
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Step 4: Click Run, then Run Module to see your output. It should look like the following:

```
>>> ==================== RESTART ====================
>>>
Welcome to my program using functions
My name is Joe Student
>>>
```

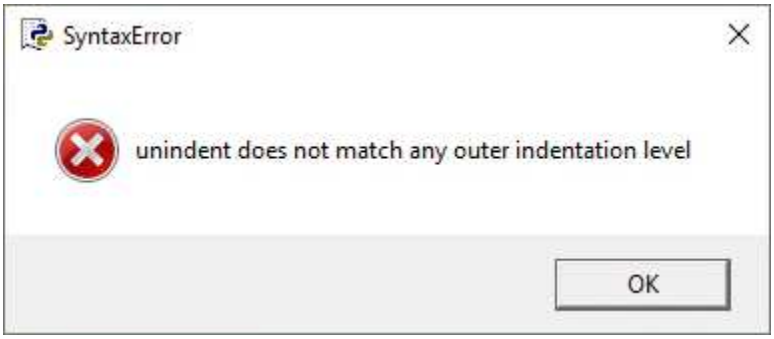
Step 5: Change your program so that the function call is tabbed over, such as:

```
# This function is to welcome people to my program
def welcome_message():
    print('Welcome to my program using functions')
    print('My name is Joe Student')

# This is a function call
    welcome_message()    # tab this line over
```

Step 6: Click Run and Run Module again. You'll notice that nothing is printed. This is because in Python, each line in a block must be indented and aligned. Function calls must be flushed to the left, and each line within a module must be aligned evenly. The following will cause a syntax error.

```
def my_function():
    print('And now for')
  print('something completely')
   print('different.')
```



Step 7: Since programs normally center around a `main` function, modify your program so that it looks as follows:

```
# The main function
def main():
    welcome_message()  # causes welcome_message to run

# This function is to welcome people to my program
def welcome_message():
    print('Welcome to my program using functions')
    print('My name is Joe Student')

# This is the main function that starts the program in
# motion
main()   # calls main
```

Step 8: Add an additional function to your program that is called `goodbye_message()`. The contents of this function should print a goodbye message. Execute your program so that it works and paste the final code below

PASTE CODE HERE

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.5 – Python Code and Variables

Critical Review

Variables can either be local or global in scope.

A local variable is created inside a function and cannot be accessed by statements that are outside a function, unless they are passed.

A local variable that needs to be used in multiple functions should be passed to the necessary functions.

An argument is any piece of data that is passed into a function when the function is called. A parameter is a variable that receives an argument that is passed into a function.

A global variable can be accessed by any function within the program, but should be avoided if at all possible.

Step 1: Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab2-5.py*. Be sure to include the .py extension.

Step 2: Document the first few lines of your program to include your name, the date, and a brief description of what the program does. Description of the program should be:

```
# This program demonstrates how to use variables and
# functions.
```

Step 3: Add a function called `main()` and a function call to `main`. Your code might look like this:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the tip calculator program')
    print()   #prints a blank line

#calls main
main()
```

Step 4: Add a function called `inputName()` under the `def main():` function. Your code might look as follows:

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

```
#this function inputs name
def inputName()
```

Step 5: Under your function definition, write a statement that allows the user to enter their name. Inside of the `main` function, call `inputName()` and write a `print` statement that displays the name. Your code might look as follows:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the variable program')
    print()   # prints a blank line
    inputName()
    print('Hello', name)

#this function inputs name
def inputName():
    name = input('Enter your name: ')

#calls main
main()
```

Step 6: Compile and run your program. Notice that when the program attempts to display the name, a syntax error occurs. This is because name is declared as a local variable within the `inputName()` function and `main` cannot access it.

Step 7: There are a couple of ways to fix this error. Examine the following code:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the variable program')
    print()   # prints a blank line

    name = inputName()
    print('Hello', name)

# this function inputs data
def inputName():
    name = input('Enter your name: ')
    return name

# calls main
main()
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

The local variable name is declared in `main` and set equal to whatever the `inputName()` function returns. Notice the `return name` statement at the end of the `inputName()` function. This passes the value that was taken in back to `main`.

Step 8: Add an additional function to your program that is called `inputAge()`. The contents of this function should be structured similar to the `inputName()` function excepts that it asks the user to enter their age. Additionally, make a call to this new function such as `age = inputAge()`. You should also display the value of age after the name is displayed. Execute your program so that it works and paste the final code below

PASTE CODE HERE

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.6 – Writing a Complete Program

Step 1: Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab2-6.py*. Be sure to include the .py extension.

Step 2: Document the first few lines of your program to include your name, the date, and a brief description of what the program does. Description of the program should be:

```
Write a program that will calculate a 20% tip and a 6%
tax on a meal price.  The user will enter the meal
price and the program will calculate tip, tax, and the
total.  The total is the meal price plus the tip plus
the tax.
```

Step 3: Add a function called `main()` and a function call to `main`.

Step 4: Add the function definition for `input_meal()`, `calc_tip()`, `calc_tax()`, `calc_total()`, and `print_info()`. Your code might look like the following:

```
# This program uses functions and variables

# the main function
def main():
    print('Welcome to the meal calculator program')
    print()   #prints a blank line

# this function will input meal price
def input_meal():

# this function will calculate tip at 20%
def calc_tip():

# this function will calculate tax at 6%
def calc_tax():

# this function will calculate tip, tax, and the total
# cost
def calc_total():

# this function will print tip, tax, the mealprice,
# and the total
def print_info():

# calls main
main()
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Step 5: Inside of `main()` under the `print() # prints a blank line` statement, create a local variable named `mealprice` that is set to the `input_meal()` function. This should look like the following:

```
mealprice = input_meal()
```

Step 6: Add the following lines of code inside of `input_meal()` function. This should look like the following:

```
mealprice = input('Enter the meal price $')
mealprice = float(mealprice)
return mealprice
```

The first line asks the user to enter their meal price. The second line converts the value to a `float`, since it will likely be a decimal value. This must be done with all potential decimal values that the user enters. The third line returns the input value of `mealprice` to the place where it was called (in Step 5).

Step 7: Inside of `main()` under the `meal = input_meal()` statement, create a local variable named `tip` that is set to the `calc_tip()` function. In this case, you must pass mealprice to the function, so it must be placed between the parentheses. This should look like the following:

```
tip = calc_tip(mealprice)
```

Step 8: Add the following lines of code inside of `calc_tip(mealprice)` function. The entire function should look like the following:

```
def calc_tip(mealprice):
    tip = mealprice * .20
    return tip
```

The first line is the function definition. It accepts `mealprice` as a parameter. The second line is to calculate tip as 20% of the mealprice. The third line returns the calculated tip to the place where it is called.

Step 9: Inside of `main()` under the `tip = calc_tip(mealprice)` statement, create a local variable named `tax` that is set to the `calc_tax()` function. In this case, you must pass `mealprice` to the function, so it must be placed between the parentheses. This should look like the following:

```
tax = calc_tax(mealprice)
```

Step 10: Add the following lines of code inside of `calc_tax(mealprice)` function. The entire function should look like the following:

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

```
def calc_tax(mealprice):
    tax = mealprice * .06
    return tax
```

The first line is the function definition. It accepts `mealprice` as a parameter. The second line is to calculate tax as 6% of the `mealprice`. The third line returns the calculated tax to the place where it is called.

Step 11: Inside of `main()` under the `tax = calc_tax(mealprice)` statement, create a local variable named `total` that is set to the `calc_total()` function. In this case, you must pass `mealprice`, `tip`, and `tax` to the function, so they must be placed between the parentheses. This should look like the following:

```
total = calc_total(mealprice, tip, tax)
```

Step 12: Add the following lines of code inside of `calc_total(mealprice, tip, tax)` function. The entire function should look like the following:

```
def calc_total(mealprice, tip, tax):
    total = mealprice + tip + tax
    return total
```

The first line is the function definition. It accepts `mealprice`, `tip`, and `tax` as parameters. The second line is to calculate the total of all three values added together. The third line returns the calculated total to the place where it is called.

Step 13: Inside of `main()` under the `total = calc_total(mealprice, tip, tax)` statement, call the `print_info ()` function. In this case, you must pass `mealprice`, `tip`, `tax`, and `total` to the function, so they must be placed between the parentheses. This should look like the following:

```
print_info(mealprice, tip, tax, total)
```

Step 14: Add the following lines of code inside of `print_info(mealprice, tip, tax, total)` function. The entire function should look like the following:

```
def print_info(mealprice, tip, tax, total):
    print 'The meal price is $', mealprice
    print 'The tip is $', tip
    print 'The tax is $', tax
    print 'The total is $', total
```

The first line is the function definition. It accepts `mealprice`, `tip`, `tax`, and `total` as parameters. The following lines print the `mealprice`, the calculated tip, the calculated tax, and the calculated total.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Step 15: Run your module and fix any errors you may have. The most common errors may be that you have misspelled something when typing, or that your indentations are not aligned properly. When running your program, enter 24.50 as the meal price. Your output should look as follows:

```
Welcome to the tip and tax calculator program

Enter the meal price $24.50
The meal price is $ 24.5
The tip is $ 4.9
The tax is $ 1.47
The total is $ 30.87
```

Step 16: When your program is completed and you have tested your output in Step 15, paste your completed program below.

PASTE CODE HERE

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

Lab 2.7 – Programming Challenge 1 – Retail Tax

This lab requires you to translate your work in the pseudocode and flowchart from Lab 2.2 and Lab 2.3 to actual code using Python. Read the following program prior to completing the lab.

A retail company must file a monthly sales tax report listing the total sales for the month and the amount of state and county sales tax collected. The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. Write a program that asks the user to enter the total sales for the month. The application should calculate and display the following:

- The amount of county sales tax
- The amount of state sales tax
- The total sales tax (county plus state)

Consider the following functions for your program:

- `main` that calls your other functions
- `inputData` that will ask for the monthly sales
- `calcCounty` that will calculate the county tax
- `calcState` that will calculate the state tax
- `calcTotal` that will calculate the total tax
- `printData` that will display the county tax, the state tax, and the total tax

If your program is correct, sample output might look as follows:

```
Welcome to the total tax calculator program

Enter the total sales for the month $12567
The county tax is $ 251.34
The state tax is $ 502.68
The total tax is $ 754.02
```

The Python Code

PASTE COMPLETED CODE HERE

Python Instructor Manual
Lab 2: Modules

Divide and Conquer

Lab 2 focuses on breaking up processes into many different modules so students become familiar with the concept of *divide and conquer*. Large tasks divided into many smaller tasks will help them understand how functions and Python work together.

Naming Functions

Python requires that you follow the same rules that you follow when naming variables, which are recapped here:

- You cannot use one of Python's key words as a function name.
- A function name cannot contain spaces.
- The first character must be one of the letters a through z, A through Z, or an underscore character (_).
- After the first character, you may use the letters a through z or A through Z, the digits 0 through 9, or underscores.
- Uppercase and lowercase characters are distinct.

Defining and Calling a Function

To create a function, you write its *definition*. Here is the general format of a function definition in Python:

```
def function_name():
    statement
    statement
    etc.
```

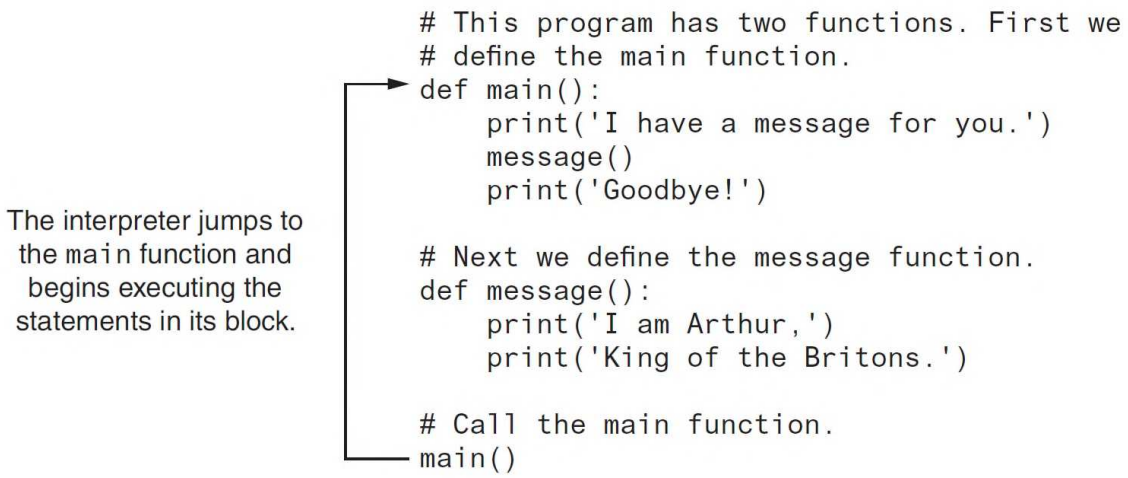
A call is then made by a simple statement such as:

```
function_name()
```

Using a Main Function

To reinforce the concept of everything centering on a main module, students should be encouraged to always use a `main()`. The following figure demonstrates the general setup with a `main` function definition, and a call to `main`. The `main` function should control the calls to all other functions within the program.

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual



Indentation in Python

In Python, each line in a block must be indented. Indentations that are not aligned evenly will cause compiler errors.

Local Variables and Passing Arguments

A variable's *scope* is the part of a program in which the variable may be accessed. A variable is visible only to statements in the variable's scope. A local variable's scope is the function in which the variable is created.

The following code will print that the value is 10 because the `changevalue()` cannot see the local value variable.

```
#the main function
def main():
    value = 10
    changevalue()
    print('The value is ', value)

def changevalue():
    value = 50

#calls main
main()
```

One way to fix this issue is to pass the value to the function and set it equal to the value. Additionally, a return value statement is added to the function. This is demonstrated in the following code:

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

```
# the main function
def main():
    value = 10
    value = changevalue(value)
    print('The value is ', value)

def changevalue(value):
    value = 50
    return value

# calls main
main()
```

Of course, additional methods could be used to fix this problem. Another solution is to print the value inside of the `changevalue()` function. However, this will not suffice when the value needs to be passed to other functions later in the program.

Multiple arguments can also be passed to functions simply by separating the values with commas. The arguments and the parameter list must be listed sequentially.

General Flow of Programs for Lab 2

While there are various ways to structure programs, Lab 2 encourages students to create local variables in main and set the function to return the modified value. For example:

```
# the main function
def main():
    totalsales = inputData()
    countytax = calcCounty(totalsales)
    statetax = calcState(totalsales)
    totaltax = calcTotal(countytax, statetax)
    printData(countytax, statetax, totaltax)

# this function will input meal price
def inputData():
    totalsales = float(input('Enter the total sales for the month $'))
    return totalsales

# this function will calculate tip at 20%
def calcCounty(totalsales):
    countytax = totalsales * .02
    return countytax

# this function will calculate tax at 6%
def calcState(totalsales):
    statetax = totalsales * .04
    return statetax

# this function will calculate tip, tax, and the total cost
def calcTotal(countytax, statetax):
    totaltax = countytax + statetax
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual

```
    return totaltax

# this function will print tip, tax, the mealprice, and the total
def printData(countytax, statetax, totaltax):
    print('The county tax is $', countytax)
    print('The state tax is $', statetax)
    print('The total tax is $', totaltax)

# calls main
main()
```

Starting Out with Programming Logic and Design 5th edition Gaddis Solutions Manual