Programming Logic and Design, 9th Edition

Chapter 2

Review Questions

1. What does a declaration provide for a variable?

   a. a name

   b. a data type

   c. both of the above

   d. none of the above

2. A variable's data type describes all of the following *except*_____.

   a. what values the variable can hold

   b. the scope of the variable

   c. how the variable is stored in memory

   d. what operations can be performed with the variable

3. The value stored in an uninitialized variable is _____.

   a. null

   b. garbage

   c. compost

   d. its identifier

4. The value 3 is a _____.

   a. numeric variable

   b. string variable

   c. numeric constant

   d. string constant

5. The assignment operator _____.

   a. is a binary operator

   b. has left-to-right associativity

   c. is most often represented by a colon

   d. two of the above

6. Multiplication has a lower precedence than _____.

   a. division

   b. subtraction

   c. assignment

   d. none of the above

7. Which of the following is not a term used as a synonym for *module*?

   a. method

   b. object

   c. procedure

   d. subroutine

8. Modularization _____.

   a. eliminates abstraction

   b. reduces overhead

   c. facilitates reusability

   d. increases the need for correct syntax

9. What is the name for the process of paying attention to important properties while ignoring nonessential details?

   a. abstraction

b. extraction

c. extinction

d. modularization

10. Every module has all of the following *except*_____ .

a. a header

b. local variables

c. a body

d. a `return` statement

11. Programmers say that one module can _____ another, meaning that the first module causes the second module to execute.

a. declare

b. define

c. enact

d. call

12. The more that a module's statements contribute to the same job, the greater the _____ of the module.

a. structure

b. modularity

c. functional cohesion

d. size

13. In most modern programming languages, a variable or constant that is declared in a module is _____ in that module.

a. global

b. invisible

c. in scope

d. undefined

14. Which of the following is *not* a typical housekeeping task?

a. displaying instructions

b. printing summaries

c. opening files

d. displaying report headings

15. Which module in a typical program will execute the most times?

a. the housekeeping module

b. the detail loop

c. the end-of-job module

d. It is different in every program.

16. A hierarchy chart tells you _____.

a. which modules call other modules

b. what tasks are to be performed within each program module

c. when a module executes

d. all of the above

17. What are nonexecuting statements that programmers place within code to explain program statements in English?

a. pseudocode

b. trivia

c. user documentation

d. comments

18. Program comments are _____.

a. required to create a runnable program

b. a form of external documentation

c. both of the above

d. none of the above

19. Which of the following is valid advice for naming variables?

a. To save typing, make most variable names one or two letters.

b. To avoid conflict with names that others are using, use unusual or unpronounceable names.

c. To make names easier to read, separate long names by using underscores or capitalization for each new word.

d. To maintain your independence, shun the conventions of your organization.

20. A message that asks a user for input is a(n) _____.

a. comment

b. echo

c. prompt

d. declaration

Programming Exercises

1. Explain why each of the following names does or does not seem like a good variable name to you.

Answer:

Answers will vary. A possible solution:

a. `stateTaxRate` – This is a good variable name.

b. `txRt` – This is a legal variable name, but cryptic.

c. `t` – This is a legal variable name, but too short to have much meaning.

d. `stateSalesTaxRateValue` – This is a legal variable name and very descriptive, but a little unwieldy.

e. `state tax rate` – This is an illegal variable name because of the embedded spaces.

f. `taxRate` – This is a good variable name.

g. `1TaxRate` – This variable name is illegal because it starts with a digit.

h. `moneyCharged` – This is a good variable name, although more could be added to specify if this is a rate, a dollar amount, or a string perhaps indicating "Y" or "N".

2. If `productCost` and `productPrice` are numeric variables, and `productName` is a string variable, which of the following statements are valid assignments? If a statement is not valid, explain why not.

Answer:

a. `productCost = 100` – Valid

b. `productPrice = productCost` – Valid

c. `productPrice = productName` – Not valid. Variables are different data types.

d. `productPrice = "24.95"` – Not valid. Variables are different data types.

e. `15.67 = productCost` – Not valid. Value on left cannot be a constant.

f. `productCost = $1,35.52` – Not valid. Numeric constant cannot have punctuation.

g. `productCost = productPrice - 10` – Valid

h. `productName = "mouse pad"` – Valid

i. `productCost + 20 = productPrice` – Not valid. Value on left must represent an address.

j. `productName = 3-inch nails` – Not valid. String constant must be placed within quotes.

k. `productName = 43` – Not valid Variable on left is a string but value on right is numeric.

l. `productName = "44"` – Valid.

m. `"99" = productName` – Not valid. Value on left must represent an address.

n. `productName = brush` – Not valid. String constant must appear within quotes.

o. `battery = productName` – Not valid. Value on left is not a declared variable.

p. `productPrice = productPrice` – Valid, but trivial.

q. `productName = productCost` – Valid.

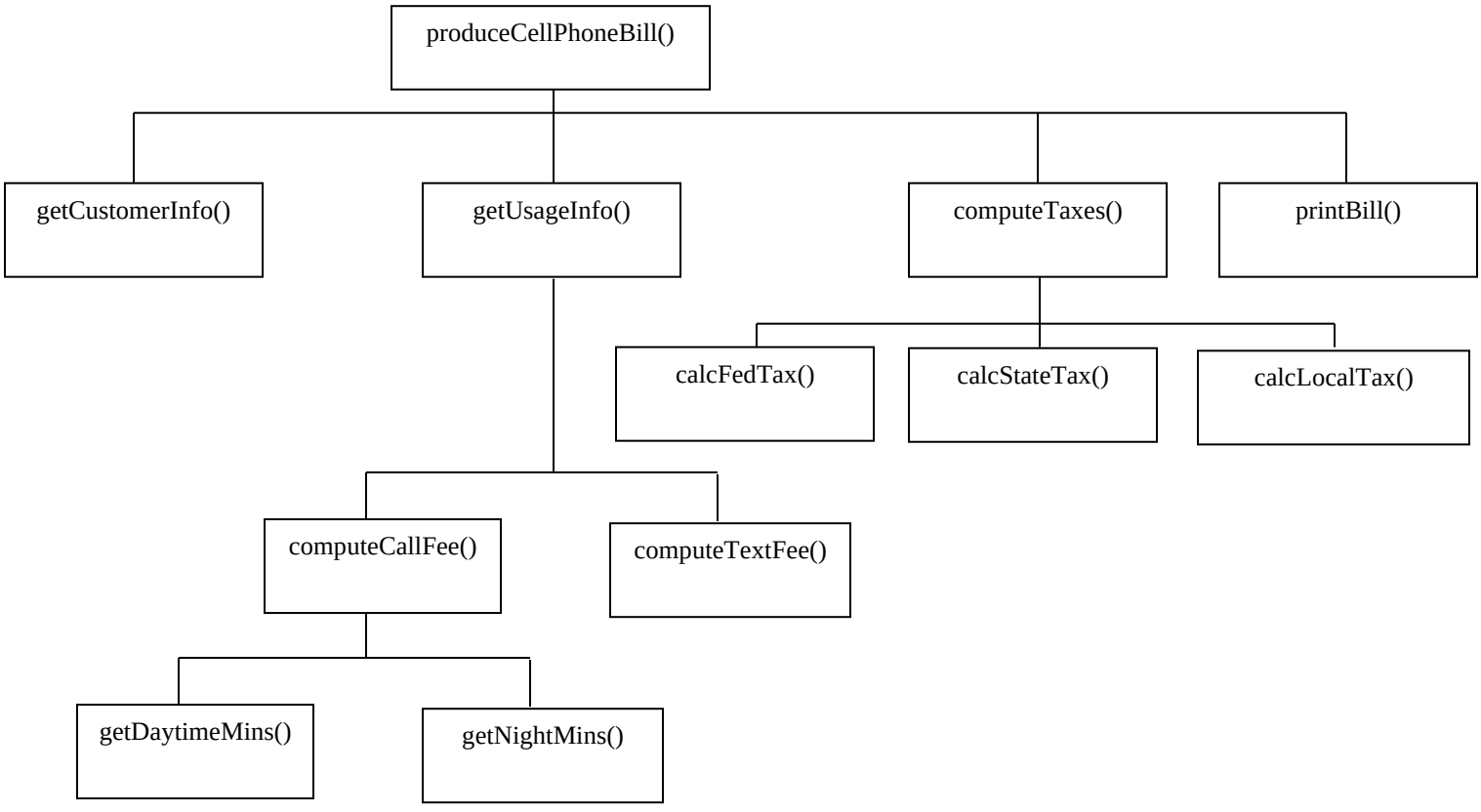
3. Assume that `speed = 10` and `miles = 5`. What is the value of each of the following expressions?

Answer:

a. `speed + 12 - miles * 2` 12
b. `speed + miles * 3` 25
c. `(speed + miles) * 3` 45
d. `speed + speed * miles + miles` 65
e. `(10 - speed) + miles / miles` 1

4. Draw a typical hierarchy chart for a program that produces a monthly bill for a cell phone customer. Try to think of at least 10 separate modules that might be included. For example, one module might calculate the charge for daytime phone minutes used.

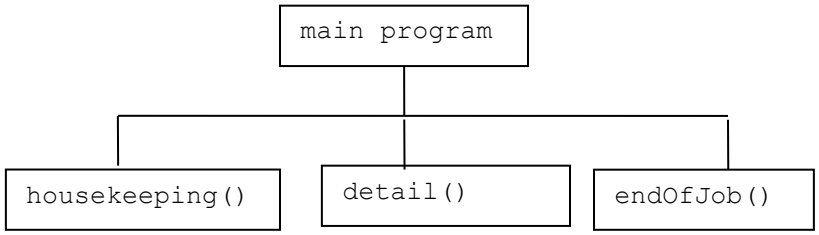
Answer:



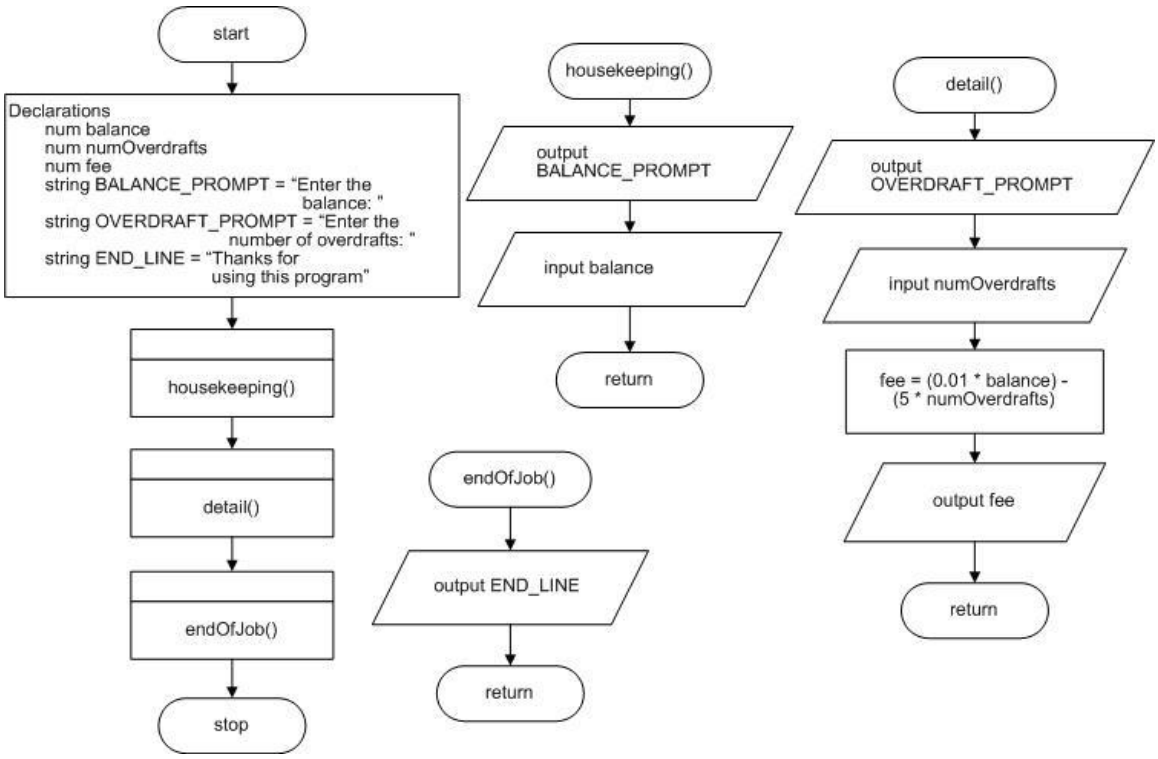
5. a. Draw the hierarchy chart and then plan the logic for a program needed by Hometown Bank. The program determines a monthly checking account fee. Input includes an account balance and the number of times the account was overdrawn. The output is the fee, which is 1 percent of the balance minus 5 dollars for each time the account was overdrawn. Use three modules. The main program declares global variables and calls housekeeping, detail, and end-of-job modules. The housekeeping module prompts for and accepts a balances. The detail module prompts for and accepts the number of overdrafts, computes the fee, and displays the result. The end-of-job module displays the message *Thanks for using this program*.

Answer: A sample solution is as follows:

a. **Hierarchy chart**:



Flowchart:



Pseudocode:

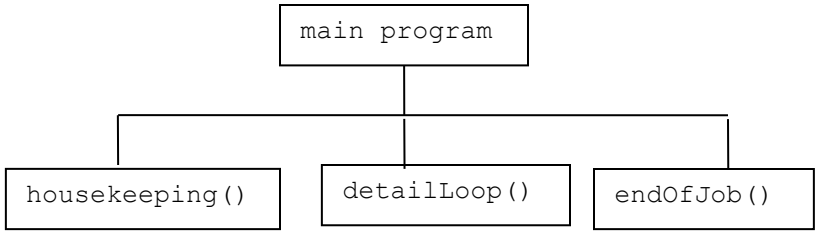
```
start
   Declarations
         num balance
         num numOverdrafts
         num fee
         string BALANCE_PROMPT = "Enter the balance: "
         string OVERDRAFT_PROMPT =
                                  "Enter the number of overdrafts: "
         string END_LINE = "Thanks for using this program"
   housekeeping()
   detail()
   endOfJob()
stop

housekeeping()
   output BALANCE_PROMPT
   input balance
return

detail()
   output OVERDRAFT_PROMPT
   input numOverdrafts
   fee = (0.01 * balance) - (5 * numOverdrafts)
   output fee
return

endOfJob()
   output END_LINE
```

```
return
```

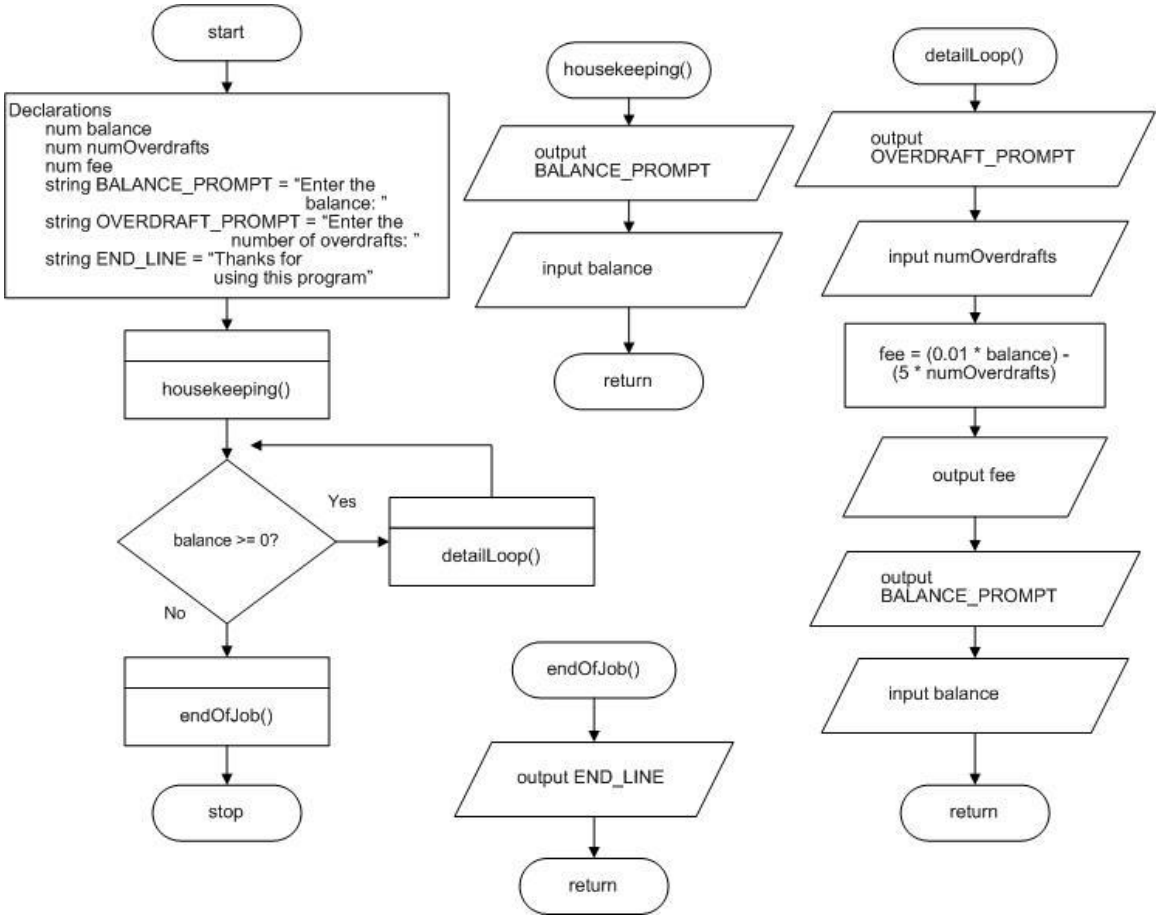
b. Revise the banking program so that it runs continuously for any number of accounts. The detail loop executes continuously while the balance entered is not negative; in addition to calculating the fee, it prompts the user for and gets the balance for the next account. The end-of-job module executes after a number less than 0 is entered for the account balance.

Answer: A sample solution is as follows:

b. **Hierarchy chart**:



Flowchart:



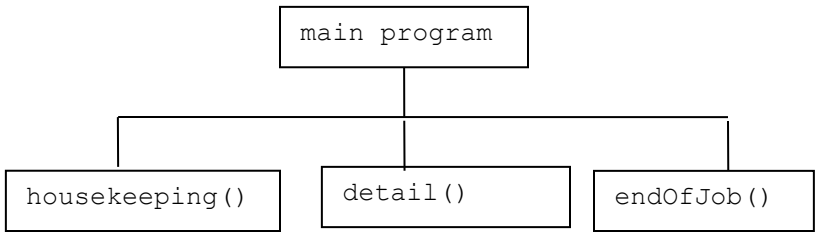
Pseudocode:

```
start
   Declarations
         num balance
         num numOverdrafts
         num fee
         string BALANCE_PROMPT = "Enter the balance: "
         string OVERDRAFT_PROMPT = "Enter the number
                                       of overdrafts: "
         string END_LINE = "Thanks for using this program"
   housekeeping()
   while balance >= 0
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output BALANCE_PROMPT
   input balance
return

detailLoop()
   output OVERDRAFT_PROMPT
   input numOverdrafts
   fee = (0.01 * balance) - (5 * numOverdrafts)
   output fee
   output BALANCE_PROMPT
   input balance
return

endOfJob()
   output END_LINE
return
```

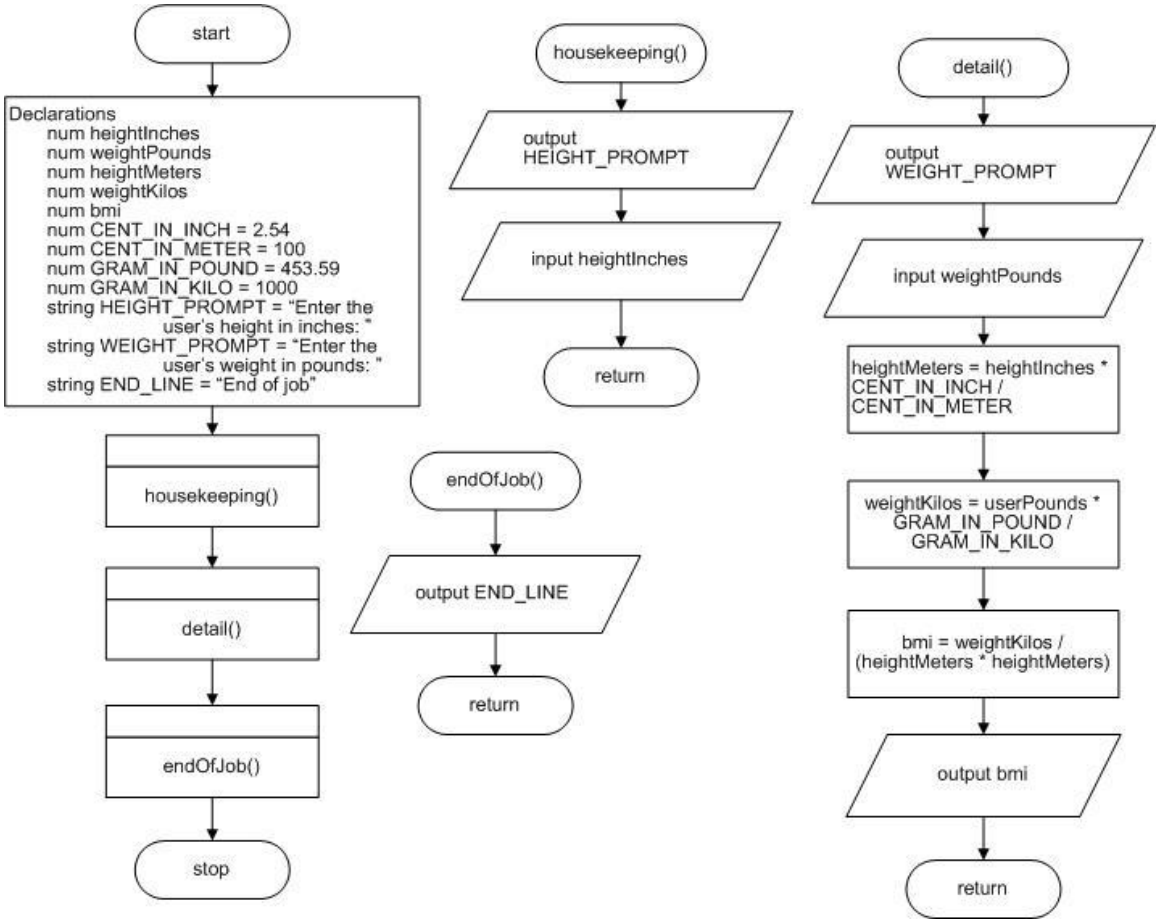
6. a. Draw the hierarchy chart and then plan the logic for a program that calculates a person's body mass index (BMI). BMI is a statistical measure that compares a person's weight and height. The program uses three modules. The first prompts a user for and accepts the user's height in inches. The second module accepts the user's weight in pounds and converts the user's height to meters and weight to kilograms. Then, it calculates BMI as weight in kilograms divided by height in meters squared, and displays the results. There are 2.54 centimeters in an inch, 100 centimeters in a meter, 453.59 grams in a pound, and 1,000 grams in a kilogram. Use named constants whenever you think they are appropriate. The last module displays the message *End of job*.

Answer: A sample solution is as follows:

Hierarchy chart:



Flowchart:



Pseudocode:

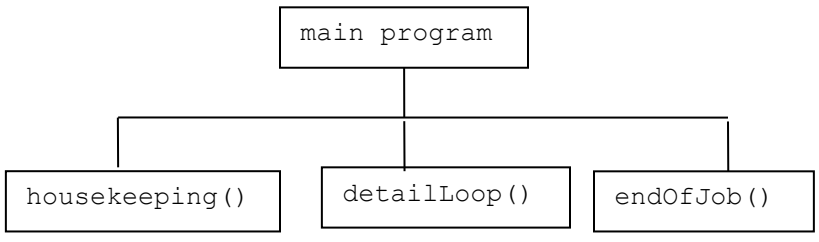
```
start
   Declarations
         num heightInches
         num weightPounds
         num heightMeters
         num weightKilos
         num bmi
```

```
            num CENT_IN_INCH = 2.54
            num CENT_IN_METER = 100
            num GRAM_IN_POUND = 453.59
            num GRAM_IN_KILO = 1000
            string HEIGHT_PROMPT = "Enter the user's height in
                                    inches: "
            string WEIGHT_PROMPT = "Enter the user's weight in
                                    pounds: "
            string END_LINE = "End of job"
      housekeeping()
      detail()
      endOfJob()
   stop

   housekeeping()
      output HEIGHT_PROMPT
      input heightInches
   return

   detail()
      output WEIGHT_PROMPT
      input weightPounds
      heightMeters = heightInches * CENT_IN_INCH / CENT_IN_METER
      weightKilos = userPounds * GRAM_IN_POUND / GRAM_IN_KILO
      bmi = weightKilos / (heightMeters * heightMeters)
      output bmi
   return

   endOfJob()
      output END_LINE
   return
```

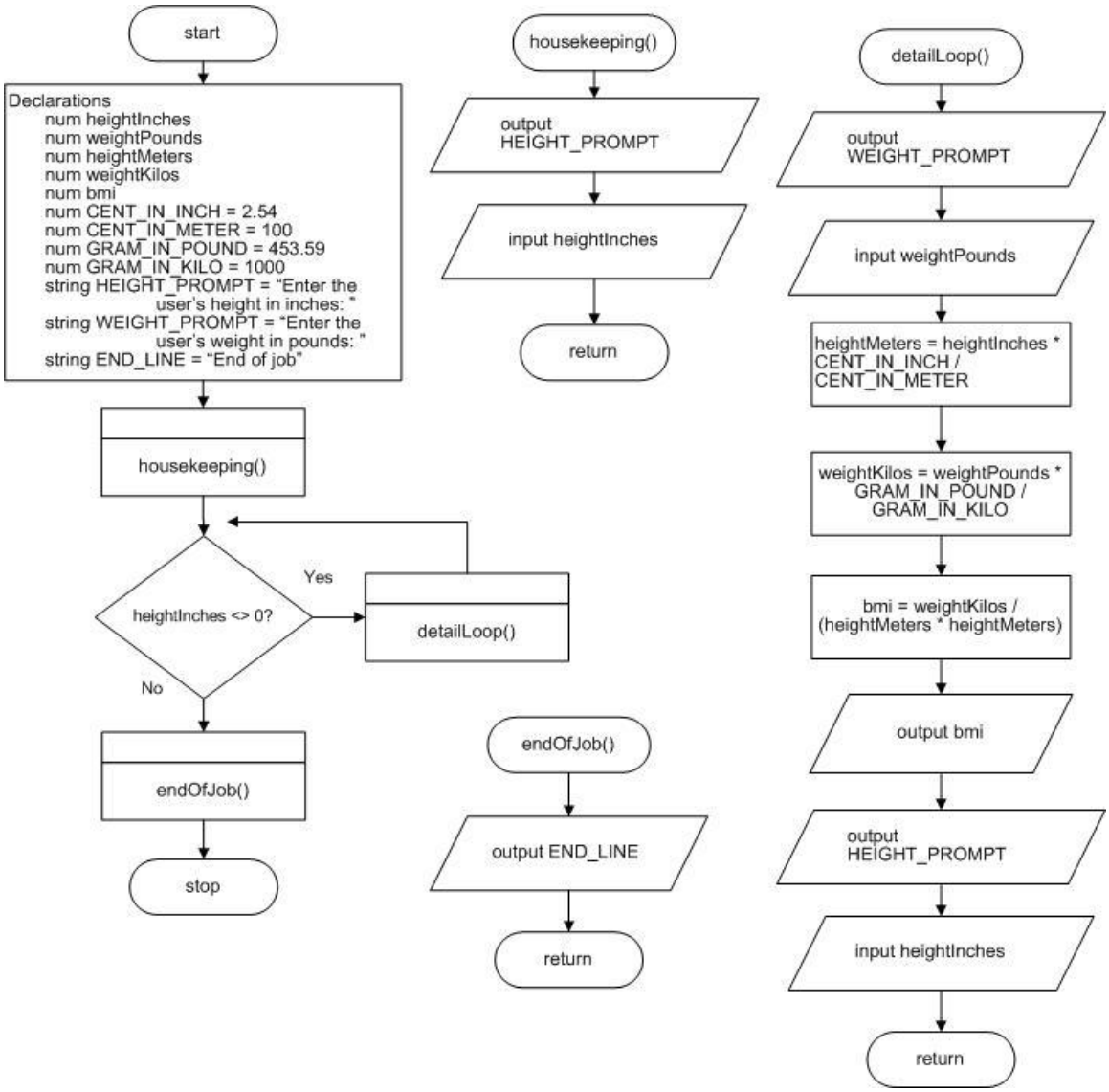
b. Revise the BMI-determining program to execute continuously until the user enters 0 for the height in inches.

Answer: A sample solution is as follows:

b. **Hierarchy chart**:



Flowchart:



Pseudocode:

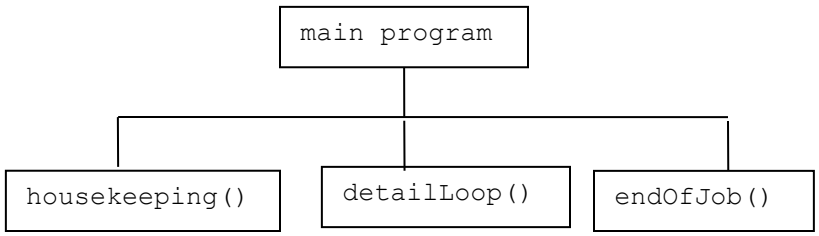
```
start
   Declarations
         num heightInches
         num weightPounds
         num heightMeters
         num weightKilos
         num bmi
         num CENT_IN_INCH = 2.54
         num CENT_IN_METER = 100
         num GRAM_IN_POUND = 453.59
         num GRAM_IN_KILO = 1000
         string HEIGHT_PROMPT = "Enter the user's height in
                                 inches: "
         string WEIGHT_PROMPT = "Enter the user's weight in
                                 pounds: "
         string END_LINE = "End of job"
```

```
   housekeeping()
   while heightInches <> 0
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output HEIGHT_PROMPT
   input heightInches
return

detailLoop()
   output WEIGHT_PROMPT
   input weightPounds
   heightMeters = heightInches * CENT_IN_INCH / CENT_IN_METER
   weightKilos = weightPounds * GRAM_IN_POUND / GRAM_IN_KILO
   bmi = weightKilos / (heightMeters * heightMeters)
   output bmi
   output HEIGHT_PROMPT
   input heightInches
return

endOfJob()
   output END_LINE
return
```

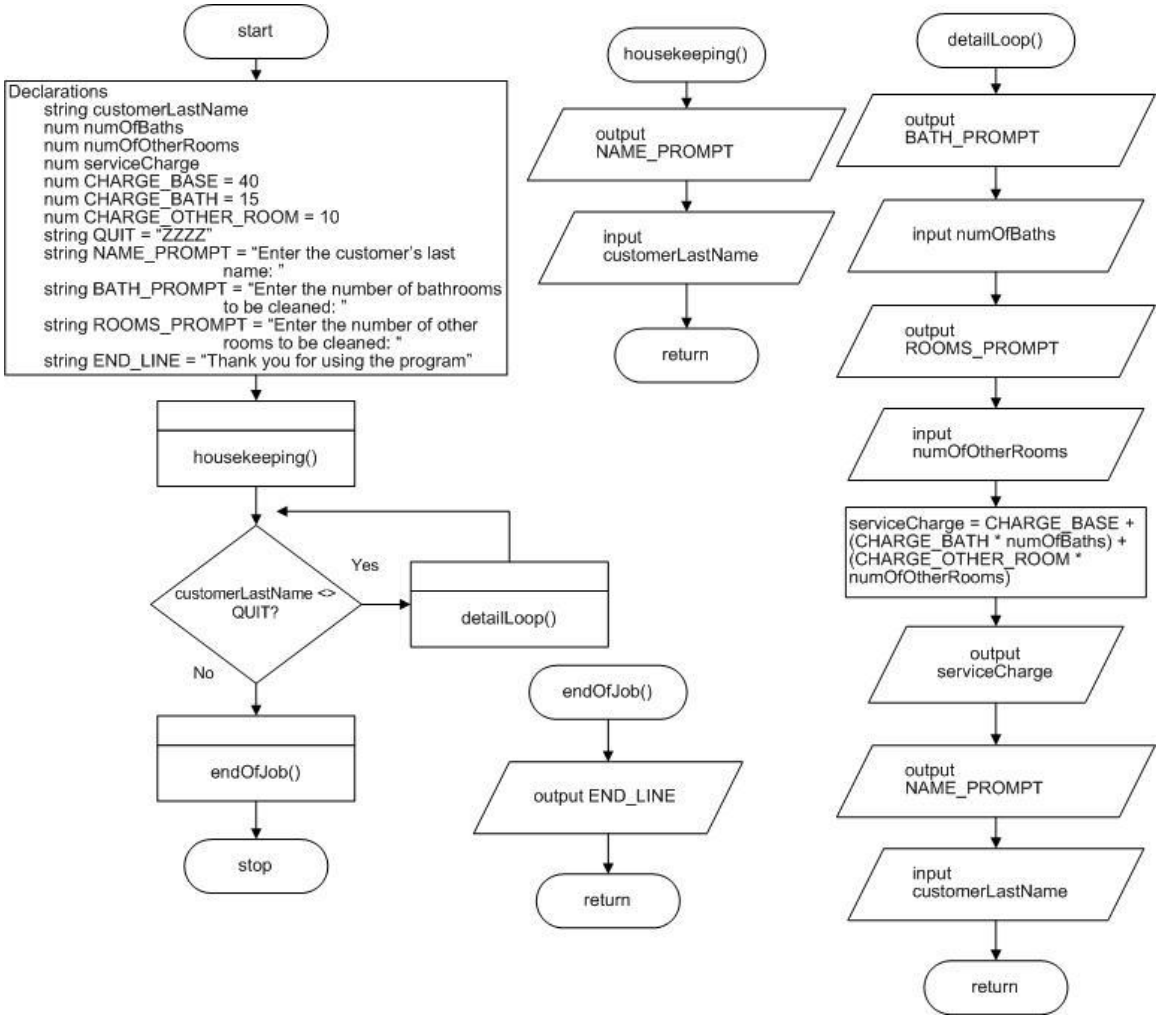
7. Draw the hierarchy chart and design the logic for a program that calculates service charges for Hazel's Housecleaning service. The program contains housekeeping, detail loop, and end-of-job modules. The main program declares any needed global variables and constants and calls the other modules. The housekeeping module displays a prompt for and accepts a customer's last name. While the user does not enter ZZZZ for the name, the detail loop accepts the number of bathrooms and the number of other rooms to be cleaned. The service charge is computed as $40 plus $15 for each bathroom and $10 for each of the other rooms. The detail loop also displays the service charge and then prompts the user for the next customer's name. The end-of-job module, which executes after the user enters the sentinel value for the name, displays a message that indicates the program is complete.

Answer: A sample solution is as follows:

Hierarchy chart:



Flowchart:



Pseudocode:

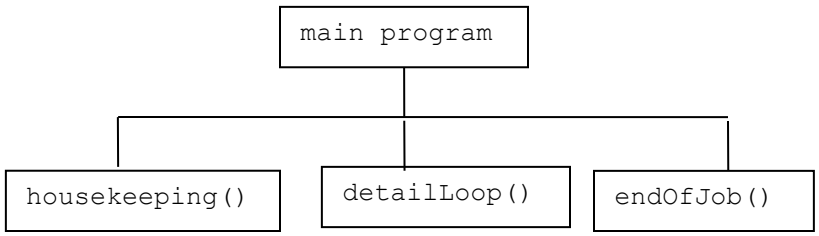
```
start
   Declarations
         string customerLastName
         num numOfBaths
         num numOfOtherRooms
         num serviceCharge
         num CHARGE_BASE = 40
```

```
         num CHARGE_BATH = 15
         num CHARGE_OTHER_ROOM = 10
         string QUIT = "ZZZZ"
         string NAME_PROMPT = "Enter the customer's last name: "
         string BATH_PROMPT = "Enter the number of bathrooms to be
                               cleaned: "
         string ROOMS_PROMPT = "Enter the number of other rooms to
                               be cleaned: "
         string END_LINE = "Thank you for using the program"
   housekeeping()
   while customerLastName <> QUIT
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output NAME_PROMPT
   input customerLastName
return

detailLoop()
   output BATH_PROMPT
   input numOfBaths
   output ROOM_PROMPT
   input numOfOtherRooms
   serviceCharge = CHARGE_BASE + (CHARGE_BATH * numOfBaths) +
                     (CHARGE_ROOM * numOfOtherRooms)
   output serviceCharge
   output NAME_PROMPT
   input customerLastName
return

endOfJob()
   output END_LINE
return
```

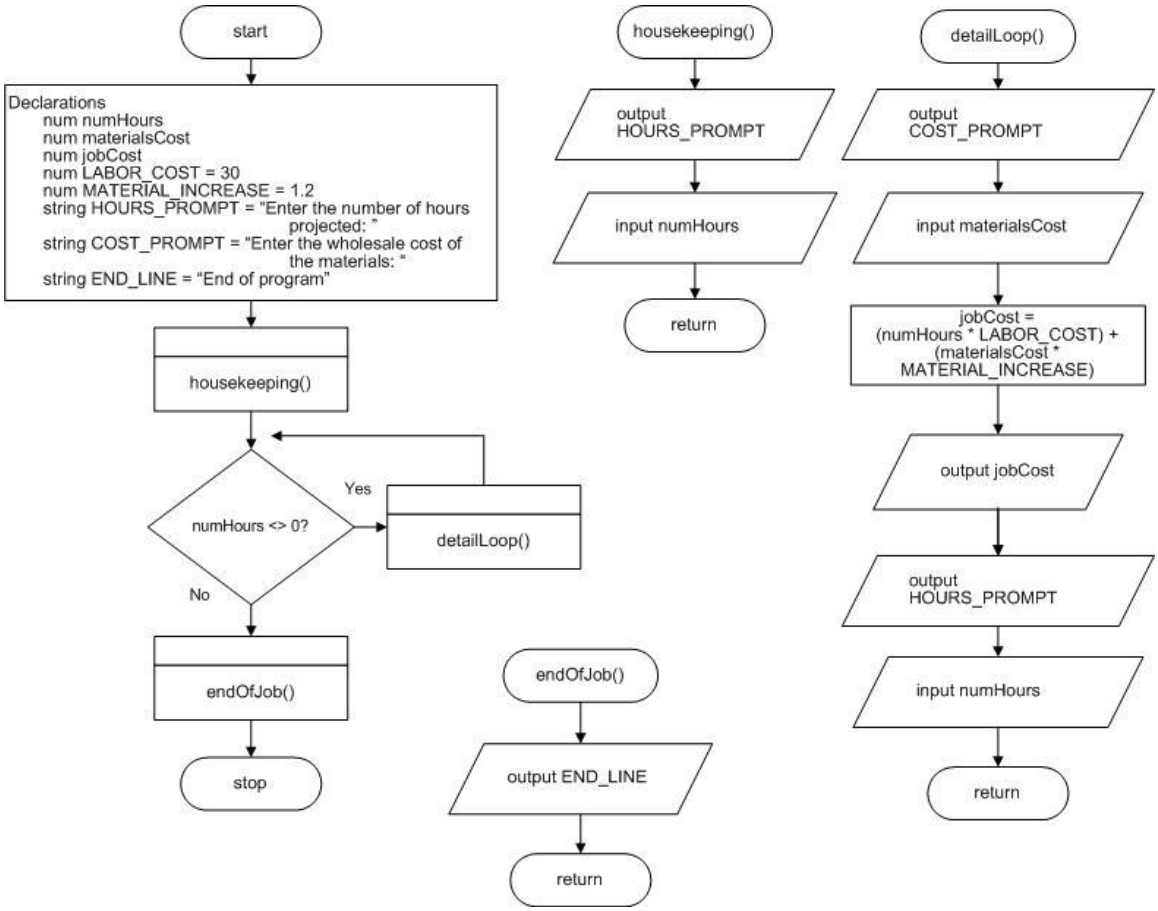
8. Draw the hierarchy chart and design the logic for a program that calculates the projected cost of a remodeling project. Assume that the labor cost is $30 per hour. Design a program that prompts the user for a number hours projected for the job and the wholesale cost of materials. The program computes and displays the cost of the job which is the number of hours times the hourly rate plus the 120% of the wholesale cost of materials. The program accepts data continuously until 0 is entered for the number of hours. Use appropriate modules, including one that displays *End of program* when the program is finished.

Answer: A sample solution is as follows:

Hierarchy chart:



Flowchart:



Pseudocode:

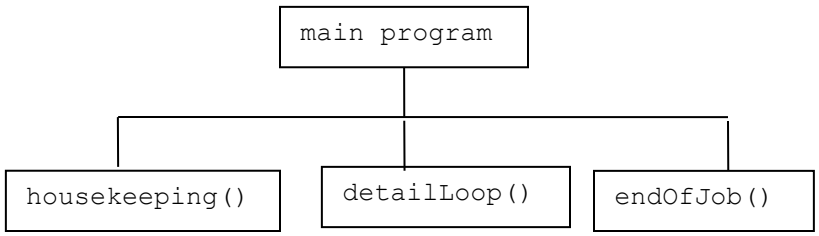
```
start
   Declarations
         string numHours
         num materialsCost
         num jobCost
         num LABOR_COST = 30
         num MATERIAL_INCREASE = 1.2
         string HOURS_PROMPT = "Enter the number of hours
                               projected: "
```

```
         string COST_PROMPT = "Enter the wholesale cost
                              of the materials: "
         string END_LINE = "End of program"
   housekeeping()
   while numHours <> 0
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output HOURS_PROMPT
   input numHours
return

detailLoop()
   output COST_PROMPT
   input materialsCost
   jobCost = (numHours * LABOR_COST) +
               (materialsCost * MATERIAL_INCREASE)
   output jobCost
   output HOURS_PROMPT
   input numHours
return

endOfJob()
   output END_LINE
return
```

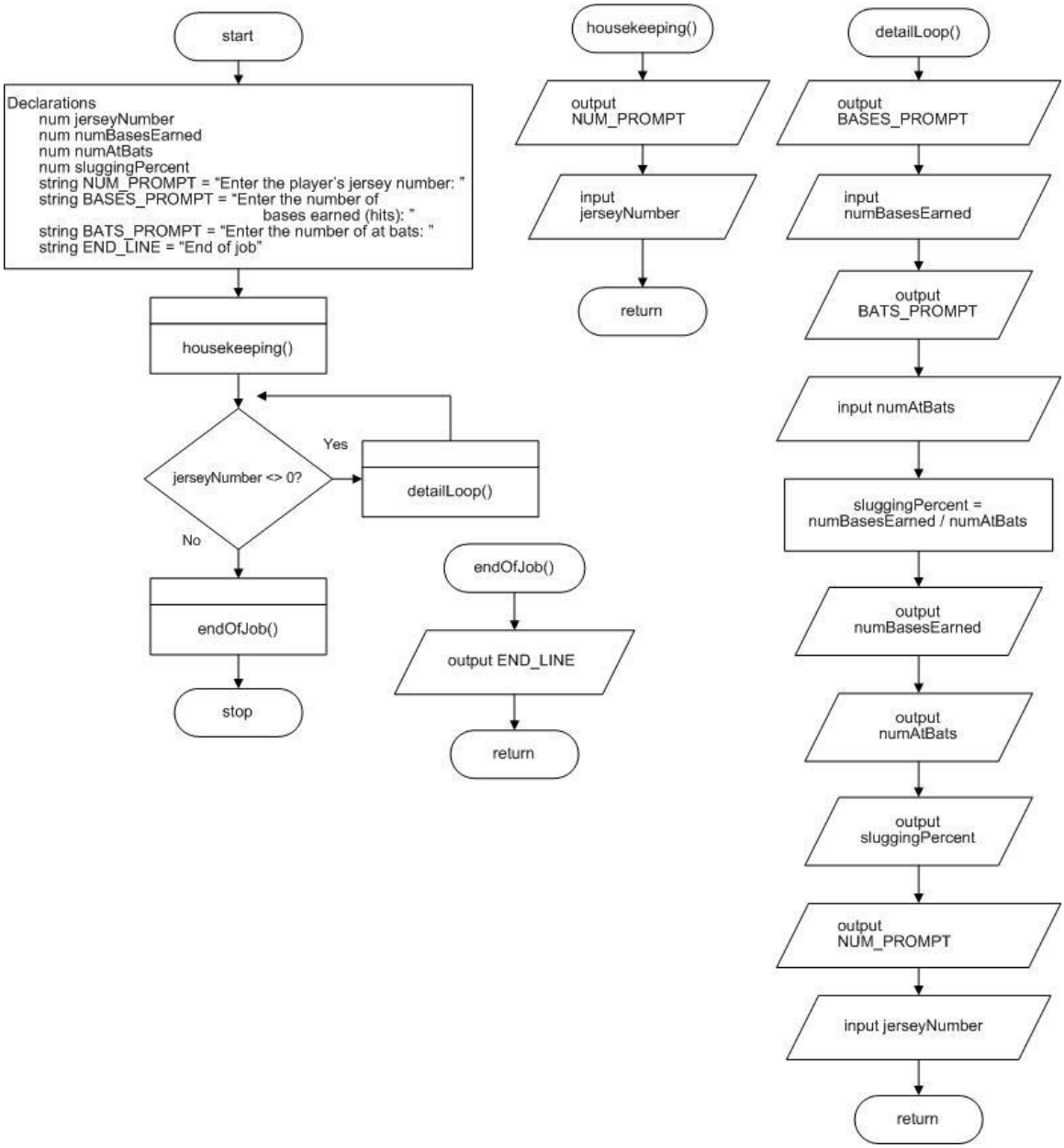
9. a. Draw the hierarchy chart and design the logic for a program needed by the manager of the Stengel County softball team, who wants to compute slugging percentages for his players. A slugging percentage is the total bases earned with base hits divided by the player's number of at-bats. Design a program that prompts the user for a player jersey number, the number of bases earned, and the number of at-bats, and then displays all the data, including the calculated slugging average. The program accepts players continuously until 0 is entered for the jersey number. Use appropriate modules, including one that displays *End of job* after the sentinel is entered for the jersey number.

Answer: A sample solution is as follows:

Hierarchy chart:


main program
housekeeping()
detailLoop()
endOfJob()

Flowchart:


start
Declarations
num jerseyNumber
num numBasesEarned
num numAtBats
num sluggingPercent
string NUM_PROMPT = "Enter the player's jersey number: "
string BASES_PROMPT = "Enter the number of bases earned (hits): "
string BATS_PROMPT = "Enter the number of at bats: "
string END_LINE = "End of job"
housekeeping()
jerseyNumber <> 0?
Yes
detailLoop()
No
endOfJob()
stop
housekeeping()
output NUM_PROMPT
input jerseyNumber
return
endOfJob()
output END_LINE
return
detailLoop()
output BASES_PROMPT
input numBasesEarned
output BATS_PROMPT
input numAtBats
sluggingPercent = numBasesEarned / numAtBats
output numBasesEarned
output numAtBats
output sluggingPercent
output NUM_PROMPT
input jerseyNumber
return

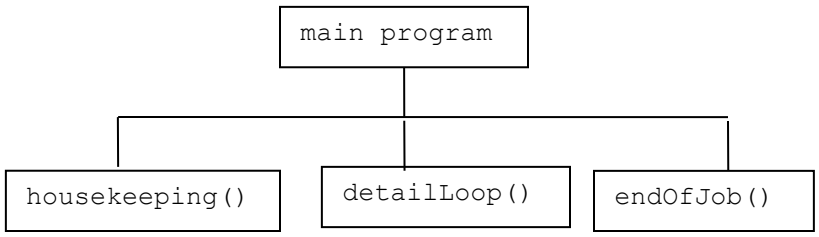
Pseudocode:

```
start
   Declarations
         num jerseyNumber
         num numBasesEarned
         num numAtBats
         num sluggingPercent
         string NUM_PROMPT = "Enter the player's jersey number: "
         string BASES_PROMPT = "Enter the number of bases
                                 earned (hits): "
         string BATS_PROMPT = "Enter the number of at bats: "
         string END_LINE = "End of job"
   housekeeping()
   while jerseyNumber <> 0
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output NUM_PROMPT
   input jerseyNumber
return

detailLoop()
   output BASES_PROMPT
   input numBasesEarned
   output BATS_PROMPT
   input numAtBats
   sluggingPercent = numBasesEarned / numAtBats
   output numBasesEarned
   output numAtBats
   output sluggingPercent
   output NUM_PROMPT
   input jerseyNumber
return

endOfJob()
   output END_LINE
return
```

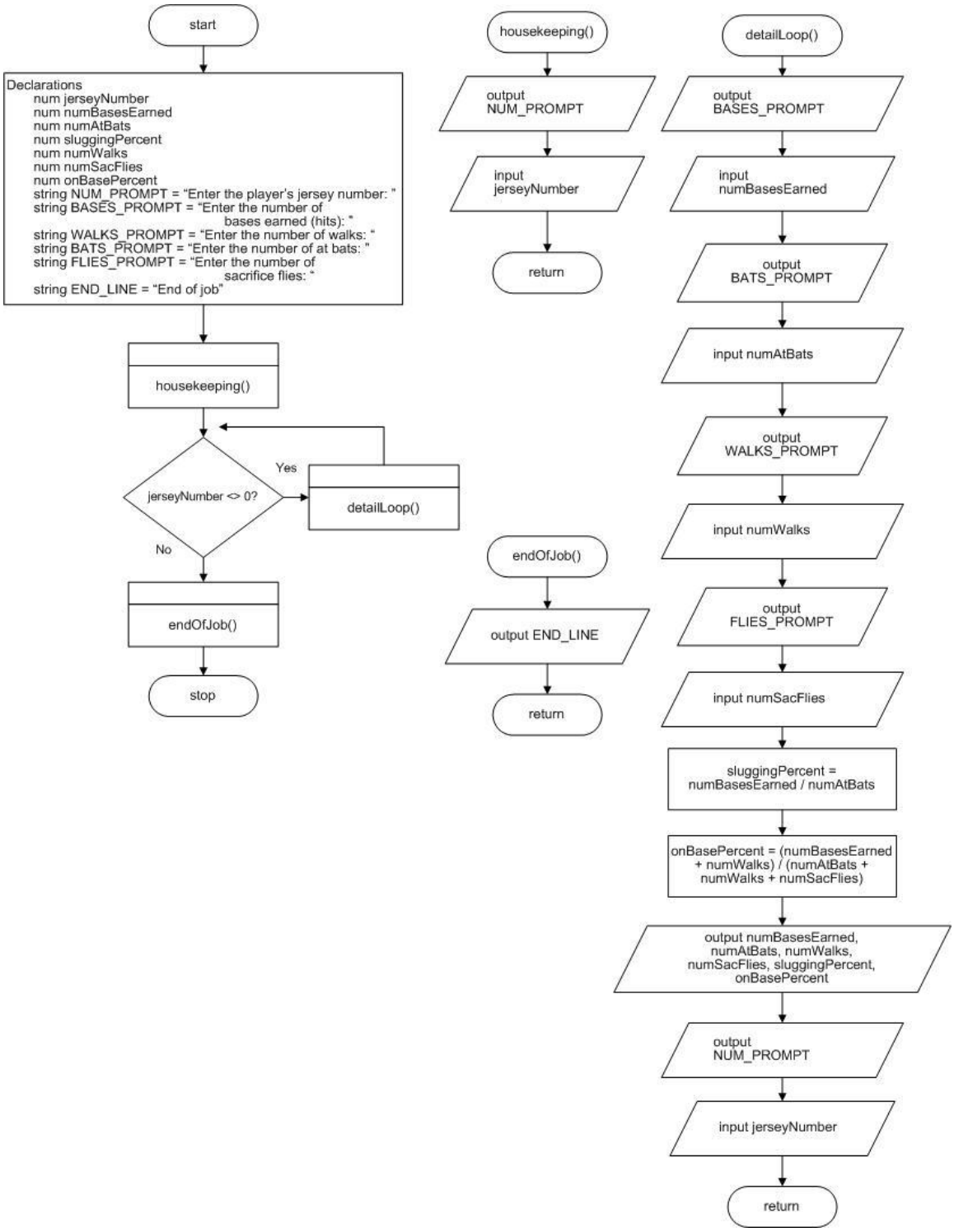
b. Modify the slugging percentage program to also calculate a player's on-base percentage. An on-base percentage is calculated by adding a player's hits and walks, and then dividing by the sum of at-bats, walks, and sacrifice flies. Prompt the user for all the additional data needed, and display all the data for each player.

Answer: A sample solution is as follows:

b. **Hierarchy chart**:



Flowchart:



Pseudocode:

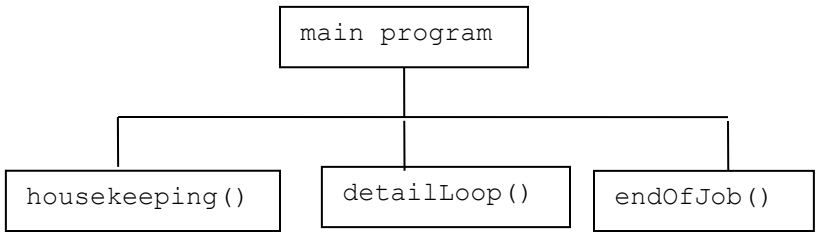
```
start
   Declarations
         num jerseyNumber
         num numBasesEarned
```

```
            num numAtBats
            num sluggingPercent
            num numWalks
            num numSacFlies
            num onBasePercent
            string NUM_PROMPT = "Enter the player's jersey number: "
            string BASES_PROMPT = "Enter the number of bases
                                   earned (hits): "
            string WALKS_PROMPT = "Enter the number of walks: "
            string BATS_PROMPT = "Enter the number of at bats: "
            string FLIES_PROMPT = "Enter the number of sacrifice
                                   flies: "
            string END_LINE = "End of job"
      housekeeping()
      while jerseyNumber <> 0
            detailLoop()
      endwhile
      endOfJob()
stop

housekeeping()
   output NUM_PROMPT
   input jerseyNumber
return

detailLoop()
   output BASES_PROMPT
   input numBasesEarned
   output BATS_PROMPT
   input numAtBats
   output WALKS_PROMPT
   input numWalks
   output FLIES_PROMPT
   input numSacFlies
   sluggingPercent = numBasesEarned / numAtBats
   onBasePercent = (numBasesEarned + numWalks) /
                     (numAtBats + numWalks + numSacFlies)
   output numBasesEarned, numAtBats, numWalks, numSacFlies,
               sluggingPercent, onBasePercent
   output NUM_PROMPT
   input jerseyNumber
return

endOfJob()
   output END_LINE
return
```

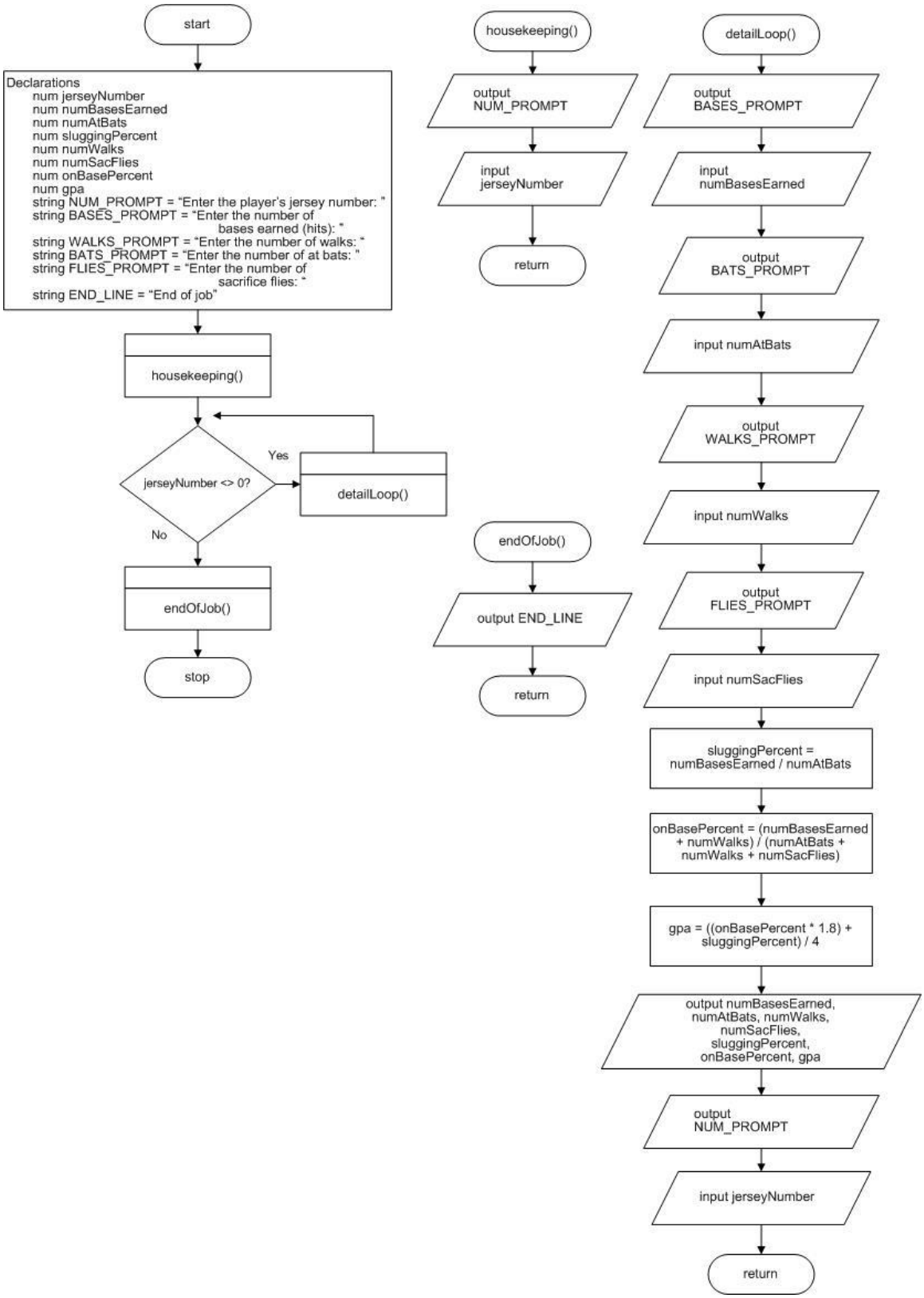
c. Modify the softball program to also compute a gross production average (GPA) for each player. A GPA is calculated by multiplying a player's on-base percentage by 1.8, then adding the player's slugging percentage, and then dividing by four.

Answer: A sample solution is as follows:

c. **Hierarchy chart**:



Flowchart:


start
Declarations
num jerseyNumber
num numBasesEarned
num numAtBats
num sluggingPercent
num numWalks
num numSacFlies
num onBasePercent
num gpa
string NUM_PROMPT = "Enter the player's jersey number: "
string BASES_PROMPT = "Enter the number of bases earned (hits): "
string WALKS_PROMPT = "Enter the number of walks: "
string BATS_PROMPT = "Enter the number of at bats: "
string FLIES_PROMPT = "Enter the number of sacrifice flies: "
string END_LINE = "End of job"
housekeeping()
jerseyNumber <> 0?
Yes
detailLoop()
No
endOfJob()
stop
housekeeping()
output NUM_PROMPT
input jerseyNumber
return
endOfJob()
output END_LINE
return
detailLoop()
output BASES_PROMPT
input numBasesEarned
output BATS_PROMPT
input numAtBats
output WALKS_PROMPT
input numWalks
output FLIES_PROMPT
input numSacFlies
sluggingPercent = numBasesEarned / numAtBats
onBasePercent = (numBasesEarned + numWalks) / (numAtBats + numWalks + numSacFlies)
gpa = ((onBasePercent * 1.8) + sluggingPercent) / 4
output numBasesEarned, numAtBats, numWalks, numSacFlies, sluggingPercent, onBasePercent, gpa
output NUM_PROMPT
input jerseyNumber
return

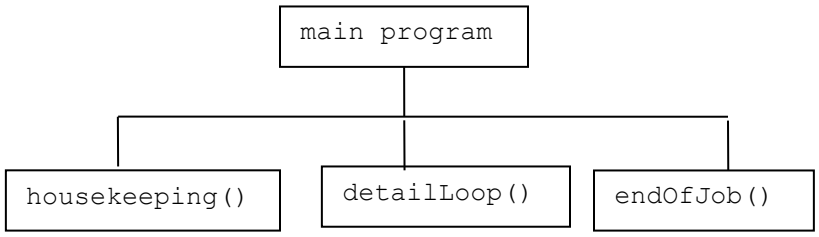
Pseudocode:

```
start
   Declarations
         num jerseyNumber
         num numBasesEarned
         num numAtBats
         num sluggingPercent
         num numWalks
         num numSacFlies
         num onBasePercent
         num gpa
         string NUM_PROMPT = "Enter the player's jersey number: "
         string BASES_PROMPT = "Enter the number of bases
                                  earned (hits): "
         string WALKS_PROMPT = "Enter the number of walks: "
         string BATS_PROMPT = "Enter the number of at bats: "
         string FLIES_PROMPT = "Enter the number of sacrifice
                                  flies: "
         string END_LINE = "End of job"
   housekeeping()
   while jerseyNumber <> 0
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output NUM_PROMPT
   input jerseyNumber
return

detailLoop()
   output BASES_PROMPT
   input numBasesEarned
   output BATS_PROMPT
   input numAtBats
   output WALKS_PROMPT
   input numWalks
   output FLIES_PROMPT
   input numSacFlies
   sluggingPercent = numBasesEarned / numAtBats
   onBasePercent = (numBasesEarned + numWalks) /
                     (numAtBats + numWalks + numSacFlies)
   gpa = ((onBasePercent * 1.8) + sluggingPercent) / 4
   output numBasesEarned, numAtBats, numWalks, numSacFlies,
               sluggingPercent, onBasePercent, gpa
   output NUM_PROMPT
   input jerseyNumber
return

endOfJob()
   output END_LINE
return
```

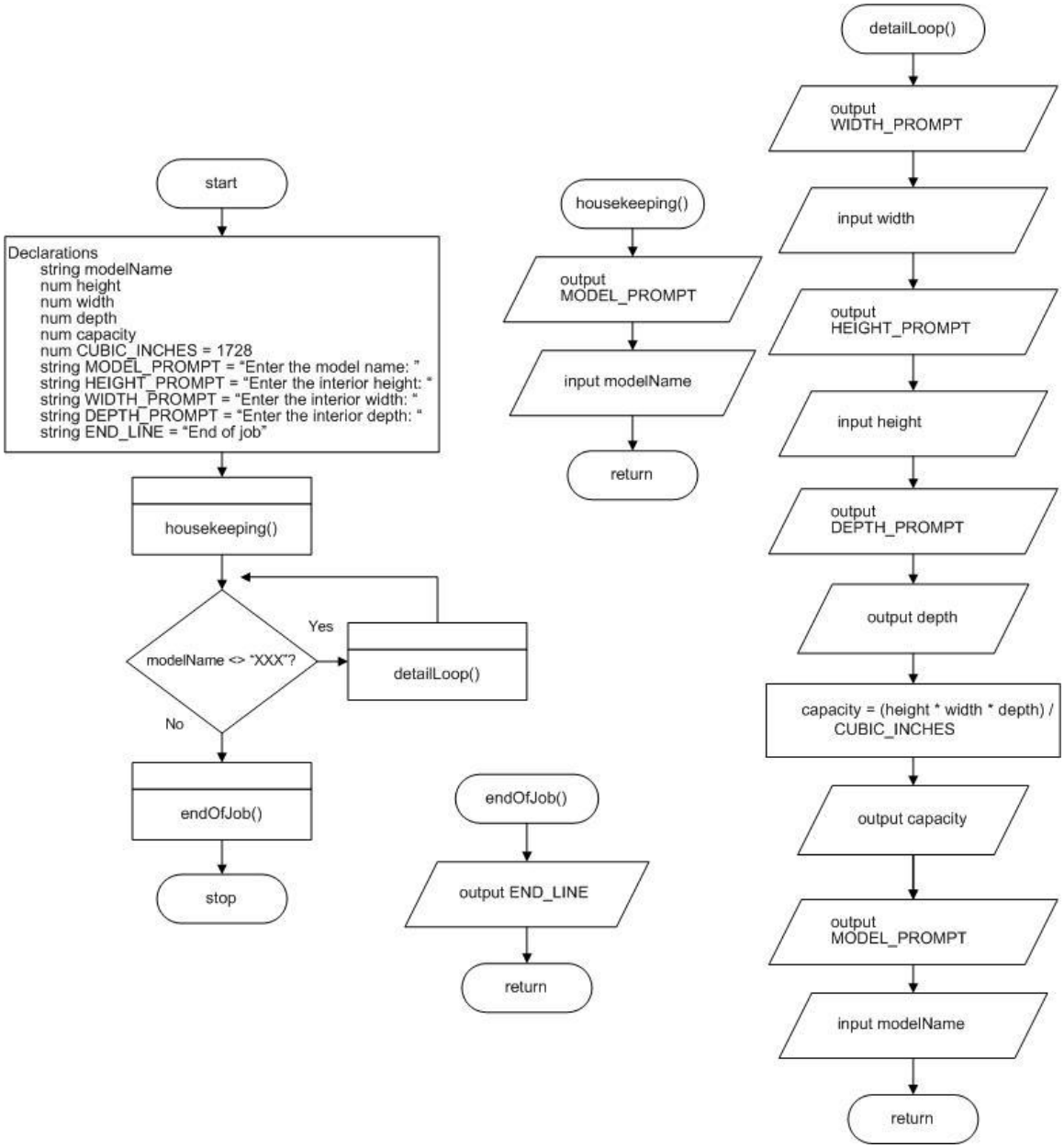
10. Draw the hierarchy chart and design the logic for a program for Arnie's Appliances. Design a program that prompts the user for a refrigerator model name and the interior height, width, and depth in inches. Calculate the refrigerator capacity in cubic feet by first multiplying the height, width, and depth to get cubic inches, and then dividing by 1728 (the number of cubic inches in a cubic foot). The program accepts model names continuously until "XXX" is entered. Use named constants where appropriate. Also use modules, including one that displays *End of job* after the sentinel is entered for the model name.

Answer: A sample solution is as follows:

Hierarchy chart:



Flowchart:



Pseudocode:

```
start
   Declarations
         string modelName
         num height
         num width
         num depth
         num capacity
         num CUBIC_INCHES = 1728
         string MODEL_PROMPT = "Enter the model name: "
         string HEIGHT_PROMPT = "Enter the interior height: "
         string WIDTH_PROMPT = "Enter the interior width: "
         string DEPTH_PROMPT = "Enter the interior depth: "
```

```
         string END_LINE = "End of job"
   housekeeping()
   while modelName <> "XXX"
         detailLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output MODEL_PROMPT
   input modelName
return

detailLoop()
   output WIDTH_PROMPT
   input width
   output HEIGHT_PROMPT
   input height
   output DEPTH_PROMPT
   input depth
   capacity = (height * width * depth) / CUBIC_INCHES
   output capacity
   output MODEL_PROMPT
   input modelNumber
return

endOfJob()
   output END_LINE
return
```

Performing Maintenance

1. A file named MAINTENANCE02-01.txt is included with your downloadable student files. Assume that this program is a working program in your organization and that it needs modifications as described in the comments (lines that begin with two slashes) at the beginning of the file. Your job is to alter the program to meet the new specifications.

Answer:

```
// This program accepts any number of purchase prices
// and computes state sales tax as 6% of the value
// and city sales tax as 2% of the value
// Modify the program so that the user enters
// the two tax rates
// at the start of the program
start
   Declarations
      num price
      num stateTaxRate
      num cityTaxRate
      num totalTax
      num total
   startUp()
```

```
   while price not equal to 0
      mainLoop()
   endwhile
   finishUp()
stop

startUp()
   output "Enter state tax rate"
   input stateTaxRate
   output "Enter city tax rate"
   input cityTaxRate
   output "Enter a price or 0 to quit"
   input price
return

mainLoop()
   totalTax = price * stateTaxRate + price * cityTaxRate
   total = price + totalTax
   output "Price is " , price, " and total tax is ", totalTax
   output "Total is ", total
   output "Enter a price or 0 to quit"
   input price
return

finishUp()
   output "End of program"
return
```

Find the Bugs

1. Your downloadable files for Chapter 2 include DEBUG02-01.txt, DEBUG02-02. txt, and DEBUG02-03.txt. Each file starts with some comments that describe the problem. Comments are lines that begin with two slashes (//). Following the comments, each file contains pseudocode that has one or more bugs you must find and correct.

   Answer:

DEBUG02-01

```
// This pseudocode segment is intended to compute and display
// the average grade of three tests
start
   Declarations
      num test1
      num test2
      num test3
         // test2 was declared twice, and test3 was not declared
      num average
   output "Enter score for test 1 "
      // Closing quote on prompt was missing
```

```
        input test1
        output "Enter score for test 2 "
        input test2
        output "Enter score for test 3 "
        input test3
        average = (test1 + test2 + test3) / 3
           // parentheses are needed; otherwise only test3 is divided
       by 3
        output "Average is ", average
           // variable name is average
     end
```

DEBUG02-02

```
// This pseudocode segment is intended to compute and display
// the average grade of three tests for any number of students.
// The program executes until the user enters a negative value
// for the first test score.
start
   Declarations
      num test1
      num test2
      num test3
      num average
   housekeeping()
   while test1 >= 0
      mainLoop()
   endwhile
   endOfJob()
stop

housekeeping()
   output "Enter score for test 1 or a negative number to quit"
   input test1
      // A value for test1 must be input
return

mainLoop()
   output "Enter score for test 2"
   input test2
   output "Enter score for test 3"
   input test3
      // A value for test3 must be input
   average = (test1 + test2 + test3) / 3
   output "Average is ", average
   output "Enter score for test 1 or a negative number to quit"
   input test1
      // test1 was misspelled
return

endOfJob()
   output "End of program"
```

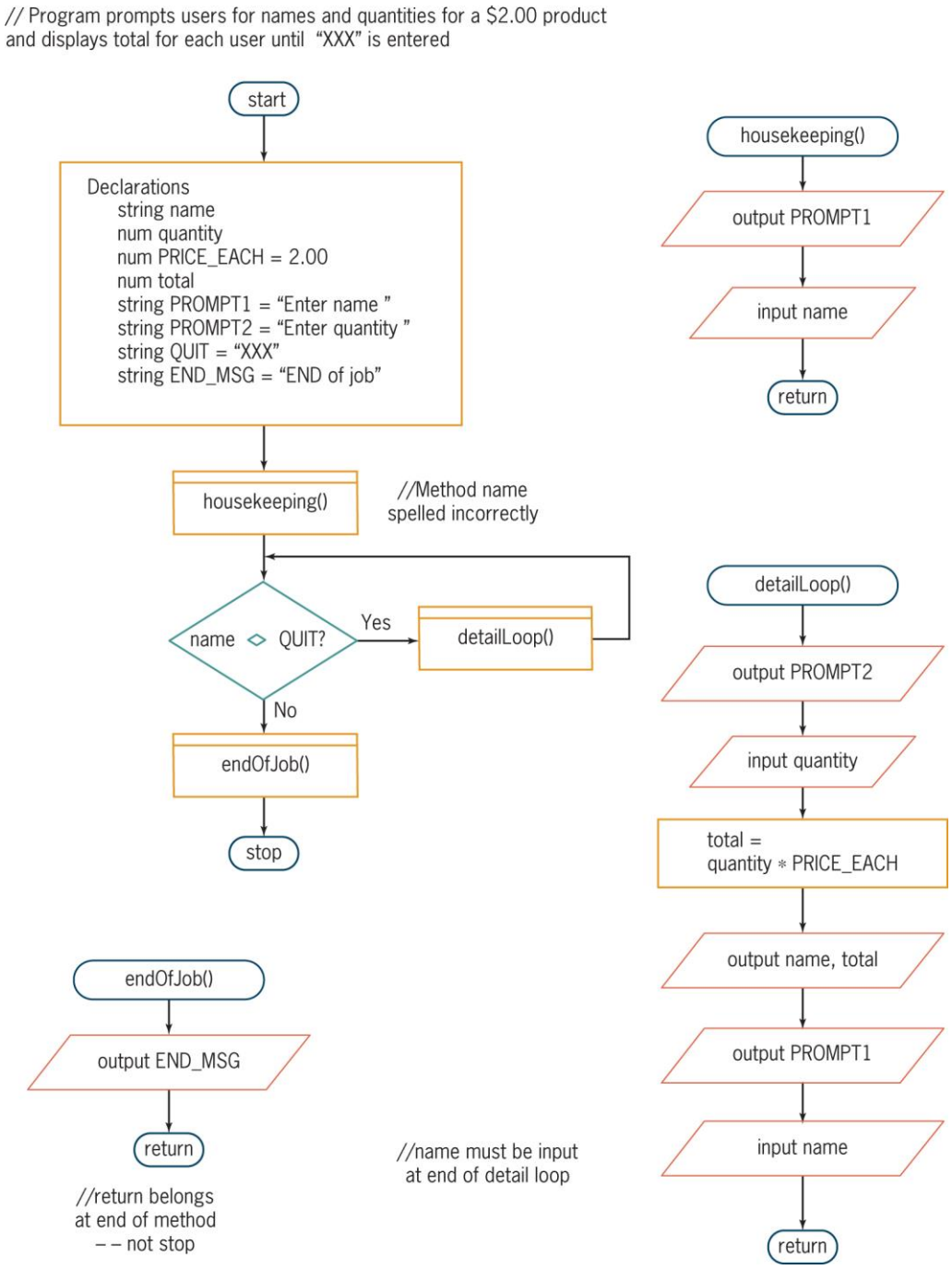
```
return
```

DEBUG02-03

```
// This pseudocode segment is intended to compute and display
// the cost of home ownership for any number of users
// The program ends when a user enters 0 for the mortgage payment
start
   Declarations
      num mortgagePayment
      num utilities
      num taxes
      num upkeep
      num total
   startUp()
      // Method name has uppercase U
   while mortgagePayment not equal to 0
      mainLoop()
         // Method name has lowercase m
   endwhile
   finishUp()
stop

startUp()
   output "Enter your mortgage payment or 0 to quit"
   input mortgagePayment
      // Variable name was misspelled
return

mainLoop()
   output "Enter utilities"
   input utilities
   output "Enter taxes"
   input taxes
   output "Enter amount for upkeep"
   input upkeep
   total = mortgagePayment + utilities + taxes + upkeep
   output "Total is ", total
   output "Enter your mortgage payment or 0 to quit"
   input mortgagePayment
      // The next mortgagePayment must be entered before the mainLoop()
ends
return

finishUp()
   output "End of program"
return
```

2. Your downloadable files for Chapter 2 include a file named DEBUG02-04.jpg that contains a flowchart with syntax and/or logical errors. Examine the flowchart and then find and correct all the bugs.

   Answer:

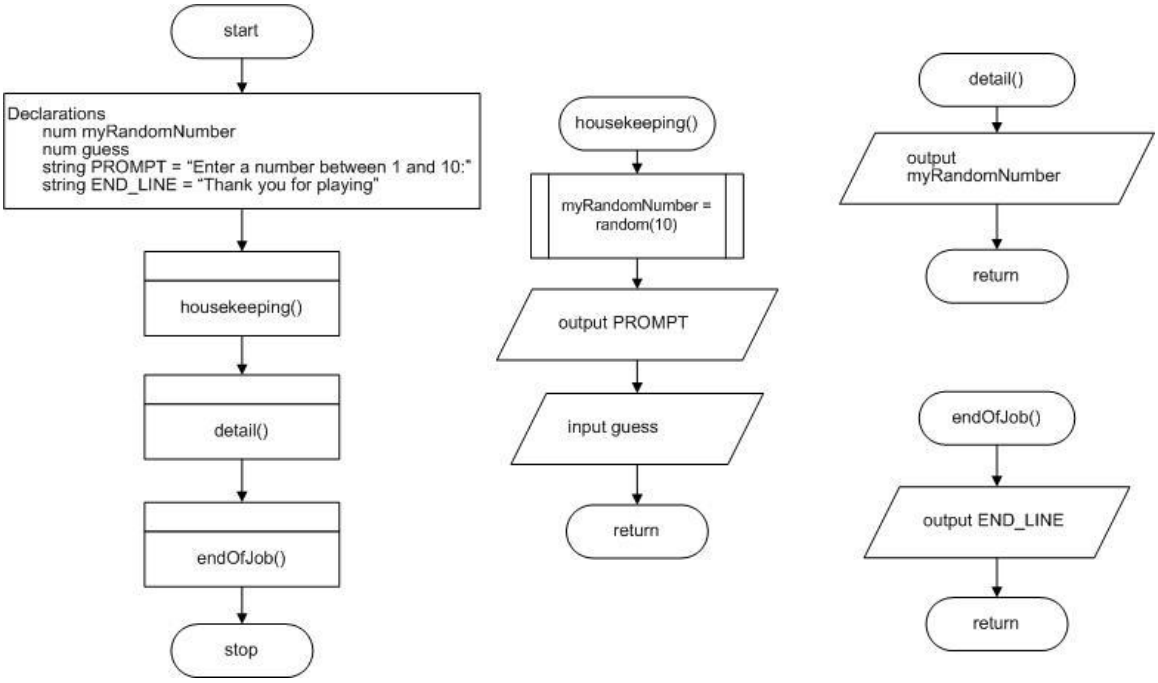


Game Zone

1. Create a flowchart or pseudocode that shows the logic for a program that generates a random number, then asks the user to think of a number between 1 and 10. Then display the randomly generated number so the user can see whether his or her guess was accurate. (In future chapters you will improve this game so that the user can enter a guess and the program can determine whether the user was correct.)

Answer: A sample solution is as follows:

Flowchart:



Pseudocode:

```
start
   Declarations
         num myRandomNumber
         num guess
         string PROMPT = "Enter a number between 1 and 10: "
         string END_LINE = "Thank you for playing"
   housekeeping()
   detail()
   endOfJob()
stop

housekeeping()
   myRandomNumber = random(10)
   output PROMPT
   input guess
return

detail()
   output myRandomNumber
return

endOfJob()
   output END_LINE
return
```