**Gaddis Chapter 5 – Part 2 - Practice Problems  
*Repetition Structures – Part 2***

Critical Review

A repetition structure causes a statement or set of statements to execute repeatedly. Repetition structures are used to perform the same task over and over. Repetition structures are commonly called loops.

There are two major types of loops:

1. A **pre-test** loop is a loop in which the test or condition that controls it appears before the loop body. The test is evaluated before the loop executes to determine whether or not to execute the loop. If the test is initially false, the loop will never execute.
2. A **post-test** loop is a loop in which the test or condition appears after the loop body. The test is evaluated after the loop executes to determine whether or not to execute the loop again. This type of loop always executes at least once.

A loop control variable is a variable that determines whether the loop will continue. When creating a loop, the programmer must write code to do all 3 of the following:

* Initialize the loop control variable.
* Make a comparison using the loop control variable to determine whether or not to execute the loop.
* Alter the value of the loop control variable within the loop.

The general structure of a Do While loop with a condition-controlled statement is:

//Declare loop control variable

Do

Statement

Statement

Etc.

//Ask Question that changes the loop control variable

While Condition

A count-controlled loop iterates a specific number of times. Although you can write this with a While or a Do-While loop, most programming languages provide a loop known as the For loop. This loop is specifically designed as a count-controlled loop.

The process of the For loop is:

* The loop keeps a count of the number of times that it iterates, and when the count reaches a specified amount, the loop stops.
* A count-controlled loop uses a variable known as a counter variable to store the number of iterations that it has performed.
* Using the counter, the following three actions take place: Initialization, Test, and Increment.

The general structure of a For statement is:

For counterVariable = startingValue To maxValue

Statement

Statement

Statement

Etc.

End For

|  |  |  |
| --- | --- | --- |
|  | **What is output by the following pseudocode segment?**  Declare Integer num  Constant Integer MAX = 4  For num = 0 To MAX  Display num  End For |  |
|  | **What is output by the following pseudocode segment?**  Declare Integer num  Constant Integer MAX = 5  Constant Integer START = 2  For num = START To MAX  Display num \* MAX  End For |  |
|  | **What is output by the following pseudocode segment?**  Declare Integer num  Declare Integer sum = 0  Constant Integer MAX = 4  For num = 0 To MAX  Set sum = sum + num  End For  Display "Sum is: ", sum |  |
|  | **Convert the following While loop to an equivalent For loop.**  Declare Integer num = 0  While num <= 3  Display "school"  Set num = num + 1  End While  Display "End" |  |
|  | **What is output by the following pseudocode segment? Assume that the user will enter 3, 5, and then -1.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 |  |
|  | **What is output by the following pseudocode segment? Assume that the user will enter 3, 5, and then -1.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 |  |
|  | **What is output by the following pseudocode segment? Assume that the user will enter 3, -1, and then 5.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 |  |
|  | **What is output by the following pseudocode segment? Assume that the user will enter -1, 3, and then 5.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 |  |
|  | **What is output by the following pseudocode segment?**  Declare Integer product = 1  Declare Integer i = 6  Do  Set product = product \* i  Set i = i + 1  While product < 9  Display "product is ", product, " and i is ", i |  |
|  | **What is output by the following pseudocode segment?**  numeric x  Declare Integer y  Declare Integer x = 1    While x < 5  Display x  Set y = 10  While y >= 8  Display y  Set y = y – 1  End While  Set x = x + 1  End While |  |
|  | **What is output by the following pseudocode segment?** Declare Integer y  Declare Integer x  Declare Integer value    For x = 3 To 5  Display x  For y = 1 To 4  Set value = x \* y  Display value  End For  End For |  |

[Click here to view the answers to #1 above.](#answer1)

1. This problem requires you to complete pseudocode for a Do-While loop. Loops are commonly used to call modules multiple times. The best design is to use a loop around the module calls in main(). Assume that pseudocode exists for getSales(), getIncrease(), storeBonus(), empBonus(), and printBonus().

**Step 1:** In the space provided below, declare a loop control variable named keepGoing of the data type Character. Initialize this variable to 'y'.

**Step 2:** In the space provided below, write a Do-While statement.

Module main ()

//Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Method calls

Do

Call getSales(monthlySales)

Call getIncrease(salesIncrease)

Call storeBonus(monthlySales, storeAmount)

Call empBonus(salesIncrease, empAmount)

Call printBonus(storeAmount, empAmount)

Display "Do you want to run the program again? (Enter y for yes)."

Input \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

While \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

End Module

[Click here to view the answer.](#answer2)

1. This problem requires you to complete pseudocode for a Do-While loop.

**Step 1:**  Write a for loop that will print 60 minutes to the screen. Complete the missing lines of code.

Constant Integer MAX\_MINUTES = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Declare Integer minutes

For \_\_\_\_\_\_\_\_\_\_\_\_\_ = 1 To \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Display \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

End For

**Step 2:** Write a for loop that will print 60 seconds to the screen. Complete the missing lines of code.

Constant Integer MAX\_SECONDS = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Declare Integer seconds

For \_\_\_\_\_\_\_\_\_\_\_\_\_ = 1 To \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Display \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

End For

**Step 3:** For loops can also be used when the user controls the number of iterations. Examine the following code:

Declare Integer numStudents

Declare Integer counter

Display "Enter the number of students in class"

Input numStudents

For counter = 1 To numStudents

Display "Student #", counter

End For

**Step 4:** Specify what will be displayed to the screen by the pseudocode in Step 3, assuming the user enters the value 3.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Step 5:** Write the missing lines for a program that will allow the user to enter how many ages they want to enter and then find the average.

Declare Integer counter

Declare Integer totalAge = 0

Declare Real averageAge = 0

Declare Integer age

Declare Integer number

Display "How many ages do you want to enter: "

Input \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For counter = 1 To number

Display "Enter age: "

Input \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Set totalAge = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ + \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

End For

Set averageAge = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Display "The average age is ", \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Click here to view the answer.](#answer3)

1. Do Debugging Exercise 3 on page 222 of the textbook.

[Click here to view the answer.](#answer4)

1. Do Algorithm Workbench problem 8 on page 221 of the textbook.

[Click here to view the answer.](#answer5)

1. The No Interest Credit Company provides zero-interest loans to customers. (They make a profit by selling advertising space in their monthly statements and selling their customer lists.) Here are the class diagram and pseudocode for the current customer class:

Customer

-accountNum: String

-name: String

-balanceDue: Real

+Customer()

+Customer(newAccountNum: String, newName: String, newBalanceDue: Real)

+getAccountNum(): String

+setAccountNum(newAccountNum: String)

+getName(): String

+setName(newName: String)

+getBalanceDue(): Real

+setBalanceDue(newBalanceDue: Real)

Class Customer

Private String accountNum

Private String name

Private Real balanceDue

Public Module Customer()

Set accountNum = " "

Set name = " "

Set balanceDue = 0.0

End Module

Public Module Customer (String newAccountNum, String newName,   
 Real newBalanceDue)

Set accountNum = newAccountNum

Set name = newName

Call setBalanceDue(newBalanceDue)

End Module

Public Function String getAccountNum ()

Return accountNum

End Module

Public Module setAccountNum (String newAccountNum)

Set accountNum = newAccountNum

End Module

Public Function String getName ()

Return name

End Module

Public Module setName (String newName)

Set name = newName

End Module

Public Function Real getBalanceDue ()

Return balanceDue

End Module

Public Module setBalanceDue (Real newBalanceDue)

If newBalanceDue >= 0 Then

Set balanceDue = newBalanceDue

Else

Set balanceDue = 0

Display newBalanceDue, " is not a valid balance due amount."

End If

End Module

End Class

* 1. Add a method to the Customer class that inputs a Customer from a file, where the file is passed as a parameter. The file contains, on each line, the account number, then the customer name, then the balance due.
  2. Create a class diagram and pseudocode for a PayoffReport application that contains two methods: the main() method and the printSchedule() method. The main() method reads Customer account records from a file named "customer.txt" and sends them, one at a time, to the printSchedule() method. The printSchedule() method prints a payoff schedule for a customer, which shows the account number and name; then the customer’s projected balance each month for the next 10 months. Assume that there is no finance charge on this account, that the customer makes no new purchases, and that the customer pays off the balance with equal monthly payments, which are 10 percent of the original bill.

[Click here to view the answer.](#answer6)



|  |  |  |
| --- | --- | --- |
|  | **What is output by the following pseudocode segment?**  Declare Integer num  Constant Integer MAX = 4  For num = 0 To MAX  Display num  End For | **0**  **1**  **2**  **3**  **4** |
|  | **What is output by the following pseudocode segment?**  Declare Integer num  Constant Integer MAX = 5  Constant Integer START = 2  For num = START To MAX  Display num \* MAX  End For | **10**  **15**  **20**  **25** |
|  | **What is output by the following pseudocode segment?**  Declare Integer num  Declare Integer sum = 0  Constant Integer MAX = 4  For num = 0 To MAX  Set sum = sum + num  End For  Display "Sum is: ", sum | **Sum is: 10** |
|  | **Convert the following While loop to an equivalent For loop.**  Declare Integer num = 0  While num <= 3  Display "school"  Set num = num + 1  End While  Display "End" | **Declare Integer num = 0**  **For num = 0 to 3**  **Display "school"**  **End For**  **Display "End"** |
|  | **What is output by the following pseudocode segment? Assume that the user will enter 3, 5, and then -1.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 | **Enter an integer**  **Enter an integer Enter an integer** |
|  | **What is output by the following pseudocode segment? Assume that the user will enter 3, -1, and then 5.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 | **Enter an integer**  **Enter an integer** |
|  | **What is output by the following pseudocode segment? Assume that the user will enter -1, 3, and then 5.**  Declare Integer num  Do  Display "Enter an integer"  Input num  While num != -1 | **Enter an integer** |
|  | **What is output by the following pseudocode segment?**  Declare Integer product  Declare Integer i = 6  Do  Set product = product \* i  Set i = i + 1  While product < 9 **Note: The comparison is for product, not i**  Display "product is ", product, " and i is ", i | **product is 42 and i is 8** |
|  | **What is output by the following pseudocode segment?**  numeric x  Declare Integer y  Declare Integer x = 1    While x < 5  Display x  Set y = 10  While y >= 8  Display y  Set y = y – 1  End While  Set x = x + 1  End While | **1**  **10**  **9**  **8**  **2**  **10**  **9**  **8**  **3**  **10**  **9**  **8**  **4**  **10**  **9**  **8** |
|  | **What is output by the following pseudocode segment?** Declare Integer y  Declare Integer x  Declare Integer value    For x = 3 To 5  Display x  For y = 1 To 4  Set value = x \* y  Display value  End For  End For | **3**  **3**  **6**  **9**  **12**  **4**  **4**  **8**  **12**  **16**  **5**  **5**  **10**  **15**  **20** |

1. This problem requires you to complete pseudocode for a Do-While loop. Loops are commonly used to call modules multiple times. The best design is to use a loop around the module calls in main(). Assume that pseudocode exists for getSales(), getIncrease(), storeBonus(), empBonus(), and printBonus().

**Step 1:** In the space provided below, declare a loop control variable named keepGoing of the data type Character. Initialize this variable to 'y'.

**Step 2:** In the space provided below, write a Do-While statement.

Module main ()

//Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

**Declare Character keepGoing = 'y'**

//Function calls

Do

Call getSales(monthlySales)

Call getIncrease(salesIncrease)

Call storeBonus(monthlySales, storeAmount)

Call empBonus(salesIncrease, empAmount)

Call printBonus(storeAmount, empAmount)

Display "Do you want to run the program again? (Enter y for yes)."

Input **keepGoing**

While **keepGoing == 'y'**

End Module

1. This problem requires you to complete pseudocode for a Do-While loop.

**Step 1:**  Write a for loop that will print 60 minutes to the screen. Complete the missing lines of code.

Constant Integer MAX\_MINUTES = **60**

Declare Integer minutes

For **minutes** = 1 To **MAX\_MINUTES**

Display **minutes**

End For

**Step 2:** Write a for loop that will print 60 seconds to the screen. Complete the missing lines of code.

Constant Integer MAX\_SECONDS = **60**

Declare Integer seconds

For **seconds** = 1 To **MAX\_SECONDS**

Display **seconds**

End For

**Step 3:** For loops can also be used when the user controls the number of iterations. Examine the following code:

Declare Integer numStudents

Declare Integer counter

Display "Enter the number of students in class"

Input numStudents

For counter = 1 To numStudents

Display "Student #", counter

End For

**Step 4:** Specify what will be displayed to the screen by the pseudocode in Step 3, assuming the user enters the value 3.

**Student #1**

**Student #2**

**Student #3**

**Step 5:** Write the missing lines for a program that will allow the user to enter how many ages they want to enter and then find the average.

Declare Integer counter

Declare Integer totalAge = 0

Declare Real averageAge = 0

Declare Integer age

Declare Integer number

Display "How many ages do you want to enter: "

Input **number**

For counter = 1 To number

Display "Enter age: "

Input **age**

Set totalAge = **totalage** + **age**

End For

Set averageAge = **totalage** / **number**

Display "The average age is ", **averageAge**

1. Do Debugging Exercise 3 on page 222 of the textbook.

**The ranges of the inner loop and outer loop are mismatched. The outer loop is set to iterate the number of times of the inner loop, and the inner loop is set to iterate the number of times of the outer loop. The values should be switched to correct this.**

1. Do Algorithm Workbench problem 8 on page 221 of the textbook.

**Declare String sure = "n"**

**While sure != "Y" AND sure != "y"**

**Display "Are you sure you want to quit?"**

**Input sure**

**End While**

1. The No Interest Credit Company provides zero-interest loans to customers. (They make a profit by selling advertising space in their monthly statements and selling their customer lists.) Here are the class diagram and pseudocode for the current customer class:

Customer

-accountNum: String

-name: String

-balanceDue: Real

+Customer()

+Customer(newAccountNum: String, newName: String, newBalanceDue: Real)

+getAccountNum(): String

+setAccountNum(newAccountNum: String)

+getName(): String

+setName(newName: String)

+getBalanceDue(): Real

+setBalanceDue(newBalanceDue: Real)

Class Customer

Private String accountNum

Private String name

Private Real balanceDue

Public Module Customer()

Set accountNum = " "

Set name = " "

Set balanceDue = 0.0

End Module

Public Module Customer (String newAccountNum, String newName,   
 Real newBalanceDue)

Set accountNum = newAccountNum

Set name = newName

Call setBalanceDue(newBalanceDue)

End Module

Public Function String getAccountNum ()

Return accountNum

End Module

Public Module setAccountNum (String newAccountNum)

Set accountNum = newAccountNum

End Module

Public Function String getName ()

Return name

End Module

Public Module setName (String newName)

Set name = newName

End Module

Public Function Real getBalanceDue ()

Return balanceDue

End Module

Public Module setBalanceDue (Real newBalanceDue)

If newBalanceDue >= 0 Then

Set balanceDue = newBalanceDue

Else

Set balanceDue = 0

Display newBalanceDue, " is not a valid balance due amount."

End If

End Module

End Class

* 1. Add a method to the Customer class that inputs a Customer from a file, where the file is passed as a parameter. The file contains, on each line, the account number, then the customer name, then the balance due.

**Public Module inputData (InputFile Ref inFile)**

**Read inFile accountNum, name, balanceDue**

**End Module**

* 1. Create a class diagram and pseudocode for a PayoffReport application that contains two methods: the main() method and the printSchedule() method. The main() method reads Customer account records from a file named "customer.txt" and sends them, one at a time, to the printSchedule() method. The printSchedule() method prints a payoff schedule for a customer, which shows the account number and name; then the customer’s projected balance each month for the next 10 months. Assume that there is no finance charge on this account, that the customer makes no new purchases, and that the customer pays off the balance with equal monthly payments, which are 10 percent of the original bill.

|  |
| --- |
|  |
|  |
| **+main()**  **+printSchedule(oneCustomer: Customer)** |

**Module main ()**

**Declare Customer oneCustomer**

**Set oneCustomer = New Customer()**

**Declare InputFile inFile**

**Open inFile "customer.txt"**

**Call oneCustomer.inputData(inFile)**

**While NOT eof(inFile)**

**Call printSchedule(oneCustomer)**

**Call oneCustomer.inputData(inFile)**

**End While**

**Close inFile**

**End Module**

**Module printSchedule (Customer oneCustomer)**

**Declare Real paymentAmt**

**Declare Integer NUM\_PAYMENTS = 10**

**Declare Integer paymentNum = 1**

**Declare Real currBal = oneCustomer.getBalanceDue()**

**Display "Payoff Schedule for: ", oneCustomer.getName()**

**Display "Account number: ", oneCustomer.getAccountNum()**

**Set paymentAmt = currBal / NUM\_PAYMENTS**

**While paymentNum <= NUM\_PAYMENTS**

**Set currBal = currBal - paymentAmt**

**Display "After payment # ", paymentNum, " balance due is ", currBal**

**Set paymentNum = paymentNum + 1**

**End While**

**End Module**