Lab 5: Repetition Structures

This lab accompanies Chapter 5 of *Starting Out with Programming Logic & Design*.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Lab 5.1 –Repetition Structures Pseudocode: Condition Controlled Loops

|  |
| --- |
| 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    A condition-controlled loop uses a true/false condition to control the number of times that it repeats.    The general structure of a While loop with a condition-controlled statement is:    //Declare loop control variable  While condition  Statement  Statement  Etc.  //Ask Question that changes the loop control variable End While    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 |

This lab requires you to implement a condition controlled loop.

**Step 1:** Examine the following main module from Lab 4.2. Loops are commonly used to call modules multiple times. The best design is to use a loop around the module calls in Main.

Module main ()

//Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

//Function calls

Call getSales(monthlySales)

Call getIncrease(salesIncrease)

Call storeBonus(monthlySales, storeAmount)

Call empBonus(salesIncrease, empAmount)

Call printBonus(storeAmount, empAmount)

End Module

**Step 2:** In the space provided, create a loop control variable named keepGoing of the data type String. Initialize this variable to "y". (Reference: Modularizing the Code in the Body of a Loop, page 226).

**Step 3:** In the space provided, write a While statement.

Module main ()

// Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

Declare String keepGoing

// Function calls

While keepGoing = y

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

End While

End Module

**Step 4:** In the space provided, create a loop control variable named keepGoing of the data type string. Initialize this variable to "y". (Reference: Writing a Do-While Loop in Pseudocode, page 229).

**Step 5:** In the space provided, write a Do-While statement.

Module main ()

// Declare local variables

Declare Real monthlySales

Declare Real storeAmount

Declare Real empAmount

Declare Real salesIncrease

Declare String keepGoing

// 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 -

End Module

# Lab 5.2 –Repetition Structures Pseudocode: Counter Controlled Loops

|  |
| --- |
| Critical Review    A count-controlled loop repeats a specific number of times.    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 variable, known as a counter variable, is used to store the number of iterations that it has performed.    The three actions that take place are initialization, test, and increment.   * Initialization: Before the loop begins, the counter variable is initialized to a starting value. * Test: The loop tests the counter variable by comparing it to a maximum value. * Increment: To increment a variable means to increase its value. This is done by adding one to the loop control variable.     Any loop can be used with a count-controlled loop.    A running total is a sum of numbers that accumulates with each iteration of a loop. The variable used to keep the running total is called an accumulator. |

This lab requires you to write a complete program using a condition controlled loop, a counter controlled loop, and an accumulator. The program is as follows:

Write a program that will allow a grocery store to keep track of the total number of bottles collected for seven days. The program should allow the user to enter the total number of bottles returned for seven days. The program will calculate the total number of bottles returned for the week and the amount paid out (the total returned times .10 cents). The output of the program should include the total number of bottles returned and the total paid out.

**Step 1:** In the pseudocode below, declare the following variables under the documentation for Step 1.

* A variable called totalBottles that is initialized to 0 o This variable will store the accumulated bottle values
* A variable called counter and that is initialized to 1 o This variable will control the loop
* A variable called todayBottles that is initialized to 0 o This variable will store the number of bottles returned on a day
* A variable called totalPayout that is initialized to 0 o This variable will store the calculated value of totalBottles times .10
* A variable called keepGoing that is initialized to "y" o This variable will be used to run the program again

**Step 2:** In the pseudocode below, make calls to the following functions under the documentation for Step 2.

* A function call to getBottles that passes totalBottles, todayBottles, and counter.
* A function called calcPayout that passes totalPayout and totalBottles.
* A function called printInfo that passes totalBottles and totalPayout

**Step 3:** In the pseudocode below, write a condition controlled while loop around your function calls using the keepGoing variable under the documentation for Step 3.

**Complete Steps 1-3 below:**

Module main ()

**// Step 1: Declare variables below**

Declare Real totalBottles

Declare int counter

Declare Real todayBottles

Declare Real totalPayout

Declare String keepGoing

**// Step 3: Loop to run program again**

While keepGoing = y

**// Step 2: Call functions**

Call getBottles(totalBottles, todayBottles, counter)

Call calcPayout(totalPayout, totalBottles)

Call printInfo(totalBottles, totalPayout)

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

Input keepGoing

End While

End Module

**Step 4:** In the pseudocode below, write the missing lines, including:

1. The missing parameter list
2. The missing condition (Hint: should run seven iterations)
3. The missing input variable
4. The missing accumulator
5. The increment statement for the counter

// getBottles module

Module getBottles(Real Ref totalBottles, Real ref todayBottles, int Ref counter)

While counter <= 7

Display "Enter number of bottles returned for the day:"

Input **c.** todayBottles

**d.** totalBottles = todayBottles + totalBottles

**e.** counter + 1

End While

End Module

**Step 5:** In the pseudocode below, write the missing lines, including:

1. The missing parameter list
2. The missing calculation

// calcPayout module

Module calcPayout(**a.** Real ref totalPayout, Real totalBottles) totalPayout = 0 // resets to 0 for multiple runs

**b.** totalPayout = totalBottles \*.10

End Module

**Step 6:** In the pseudocode below, write the missing lines, including:

1. The missing parameter list
2. The missing display statement
3. The missing display statement

//printInfo module

Module printInfo(Real totalBottles, Real totalPayout)

* 1. Display “The number of bottles you have returned is: “, totalBottles
  2. Display “You have earned $”, totalPayout

End Module

# Lab 5.3 – Flowcharts

Critical Review

In a While loop, the question is asked first. After the statements process, the

control goes back above the condition.

Condition

Statements

False

True

In a Do

-

While loop, the question is asked last. The statements always process

at least one time.

Condition

Statements

False

True

In Raptor, you can place the modules before or after the condition depending on

whether you

want to use a Do

-

While or a while loop.

This lab requires you to convert your pseudocode in Lab 5.2 to a flowchart. Use an application such Visio or Draw.io.

**Insert** your finished flowchart in the space below for later reference. You will also upload this flowchart in Blackboard. If you are using Draw.io, please convert your flowchart to a pdf file before uploading it.



# Lab 5.4 – Python Code

|  |
| --- |
| Critical Review    In Python, you use the while statement to write a condition-controlled loop. The loop has two parts: (1) a condition that is tested for a true or false value, and (2) a statement or set of statements that is repeated as long as the condition is true.    A while loop can also be used for count-controlled loops.    Here is the general format of the while loop in Python:    while *condition*:  *statement*  *statement*  *etc.*    Since the while loop is a pre-test, it is important to initialize your loop control variable to a starting value so that the first iteration will be true.    As with all loops, be sure to change the loop control variable either by incrementing or asking a question. |

The goal of this lab is to convert the Bottle Return program to Python code.

**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 *Lab5-4.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.

**Step** **3:** Start your program with the following code for main:

# Lab 5-4 The Bottle Return Program

# the main function def main(): # calls main

main()

**Step 4:** Python only supports while loops, so endProgram must be initialized to 'no'. Under def main():, create a variable called endProgram and set it to 'no' such as:

endProgram = 'no'

**Step 5:** The next step is to write a while statement with the condition of endProgram == 'no':. The statement should be aligned with the statement in Step 4. The code should be as follows:

while endProgram == 'no':

**Step 6:** The code inside of the while statement should be tabbed over and include your function calls. The function getBottles() will return totalBottles so the call should be set to that variable. The function calcPayout should pass totalBottles as an argument and will return totalPayout from the function. The function printInfo should pass totalBottles and totalPayout as arguments. The code should look as follows:

totalBottles = getBottles()

totalPayout = calcPayout(totalBottles)

printInfo(totalBottles, totalPayout)

**Step 7:** The next step is to modify the loop control variable. This is done with a simple input statement such as:

endProgram = input('Do you want to end the program? (Enter yes or no): ')

**Step 8:** The next function to code is getBottles. Write a definition for getBottles that accepts no arguments. The code should look as follows:

#this function will get the number of bottles returned

def getBottles():

**Step 9:** The first step in your function should be to set your variables to values. In Python and most programming languages, in order for variables to be used, they need to have a starting value. This also allows for a reset of the variables when the program runs again. Set totalBottles and todayBottles to 0 and counter to 1. Your code should look as follows:

totalBottles = 0 todayBottles = 0

counter = 1

**Step 10:** Write a while loop with the condition of counter <= 7. This code should look as follows:

while counter <= 7:

**Step 11:** Inside the while loop, write a statement that allows the user to enter the number of bottles for today. This code should look as follows:

todayBottles = input('Enter number of bottles for today: ')

**Step 12:** Next, write the accumulator statement. This code should look as follows:

totalBottles = totalBottles + todayBottles

**Step 13:** The last statement inside the loop should increment counter by one so the loop will end after seven iterations. This code should look as follows:

counter = counter + 1

**Step 14:** The final statement in the getBottles function is to return totalBottles back to main. This code should look as follows:

return totalBottles

**Step 15:** Create a function definition for calcPayment that accepts totalBottles in the parameter list. This function should first reset totalPayout to 0. This is done so that on multiple iterations of the program, totalPayout is reset to 0. The second step in this function is to calculate totalPayout as totalBottles times .10. The last step is to return totalPayout. Your code should look as follows:

# this function will calculate the payout

def calcPayout(totalBottles):

totalPayout = 0

totalPayout = totalBottles \* .10 return totalPayout

**Step 16:** The final function in this program is printInfo. This function accepts two variables in the parameter list so that it can display the total number of bottles returned and the total amount paid out. Your code should look as follows:

# this function will display the information

def printInfo(totalBottles, totalPayout):

print 'The total number of bottles collected is', totalBottles

print 'The total paid out is $', totalPayout

**Step 17:** Click Run and Run Module to see how your program processes. Test the following values to verify the expected output.

>>>

Enter number of bottles for today: 346

Enter number of bottles for today: 238

Enter number of bottles for today: 638

Enter number of bottles for today: 890

Enter number of bottles for today: 1035

Enter number of bottles for today: 899

Enter number of bottles for today: 536

The total number of bottles collected is 4582

The total paid out is $ 458.2

Do you want to end the program? (Enter yes or no): no

Enter number of bottles for today: 425

Enter number of bottles for today: 342

Enter number of bottles for today: 235

Enter number of bottles for today: 539

Enter number of bottles for today: 485

Enter number of bottles for today: 321

Enter number of bottles for today: 128

The total number of bottles collected is 2475

The total paid out is $ 247.5

Do you want to end the program? (Enter yes or no): yes

>>>

**Step** **18:** Execute your program so that it works and paste the final code below

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

# Name: David White

# Class: CIS-1400

# Assignment: Lab 5-4

# File: lab5-4.py

# Purpose: Bottle return program

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

print('\n\*\*\*David White\*\*\*\n') # Display author's name

# main function

def main():

endProgram = 'no'

while endProgram == 'no':

totalBottles = getBottles()

totalPayout = calcPayout(totalBottles)

printInfo(totalPayout, totalBottles)

endProgram = input(print('Do you want to end the program? y/n'))

# end while

return

# get the number of bottles

def getBottles():

totalBottles = 1

counter = 0

while counter < 7:

todayBottles = int(input('Enter the number of bottles for today: '))

totalBottles = totalBottles + todayBottles

counter = counter + 1

# end While

return totalBottles

# calculate payout

def calcPayout(totalBottles):

totalPayout = 0

totalPayout = totalBottles \* .10

return totalPayout

# display info

def printInfo(totalBottles, totalPayout):

print('The total number of bottles collected is', totalPayout)

print('The total paid out is $', totalBottles)

return

# the variables are switched for some reason

# I’ve set it so it makes sense in the program

# call main

main()

# Lab 5.5 – Programming Challenge 1 – Yum Yum Burger Joint

Write the Flowchart and Python code for the following programming problem and the pseudocode below.

Write a program that will calculate the cost of purchasing a meal. This program will include decisions and loops. Details of the program are as follows:

* Your menu items only include the following food with accompanied price:

o Yum Yum Burger = .99 o Grease Yum Fries = .79 o Soda Yum = 1.09

* Allow the user of the program to purchase any quantity of these items on one order.
* Allow the user of the program to purchase one or more types of these items on one order.
* After the order is placed, calculate total and add a 6% sales tax.
* Print to the screen a receipt showing the total purchase price.

Your sample output might look as follows:

Enter 1 for Yum Yum Burger

Enter 2 for Grease Yum Fries

Enter 3 for Soda Yum

Enter now -> 1

Enter the number of burgers you want 3

Do you want to end your order? (yes/no): no

Enter 1 for Yum Yum Burger

Enter 2 for Grease Yum Fries

Enter 3 for Soda Yum

Enter now -> 3

Enter the number of sodas you want 2

Do you want to end your order? (yes/no): no

Enter 1 for Yum Yum Burger

Enter 2 for Grease Yum Fries

Enter 3 for Soda Yum

Enter now -> 1

Enter the number of burgers you want 1

Do you want to end your order? (yes/no): no

Enter 1 for Yum Yum Burger

Enter 2 for Grease Yum Fries

Enter 3 for Soda Yum

Enter now -> 2

Enter the number of fries you want 2

Do you want to end your order? (yes/no): yes

The total price is $ 8.1832

Do you want to end program? (yes/no): no

Enter 1 for Yum Yum Burger

Enter 2 for Grease Yum Fries

Enter 3 for Soda Yum

Enter now -> 2

Enter the number of fries you want 2

Do you want to end your order? (yes/no): no

Enter 1 for Yum Yum Burger

Enter 2 for Grease Yum Fries

Enter 3 for Soda Yum

Enter now -> 3

Enter the number of sodas you want 2

Do you want to end your order? (yes/no): yes

The total price is $ 3.9856

Do you want to end program? (yes/no): yes

## The Pseudocode

Module main()

Call declareVariables(endProgram, endOrder, totalBurger, totalFry, totalSoda, total, tax, subtotal, option, burgerCount, fryCount, sodaCount)

// Loop to run program again

While endProgram == "no"

Call resetVariables(totalBurger, totalFry, totalSoda, total,

tax, subtotal)

// Loop to take in order

While endOrder == "no"

Display "Enter 1 for Yum Yum Burger"

Display "Enter 2 for Grease Yum Fries"

Display "Enter 3 for Soda Yum"

Input option

If option == 1 Then

Call getBurger(totalBurger, burgerCount)

Else If option == 2 Then

Call getFry(totalFry, fryCount) Else If option == 3 Then

Call getSoda(totalSoda, sodaCount)

End If

Display "Do you want to end your order? (yes/no): "

Input endOrder

End While

Call calcTotal(burgerTotal, fryTotal, sodaTotal, total,

subtotal, tax) Call printReceipt(total)

Display "Do you want to end the program? (yes/no): "

Input endProgram

End While

End Module

Module declareVariables(String Ref endProgram, String Ref endOrder,

Real Ref totalBurger, Real Ref totalFry, Real

Ref totalSoda, Real Ref total, Real Ref tax,

Real Ref subtotal, Real Ref option,

Real Ref burgerCount, Real Ref fryCount, Real

Ref sodaCount)

Declare String endProgram = "no"

Declare String endOrder = "no"

Declare Real totalBurger = 0

Declare Real totalFry = 0

Declare Real totalSoda = 0

Declare Real total = 0

Declare Real tax = 0

Declare Real subtotal = 0

Declare Integer option = 0

Declare Integer burgerCount = 0

Declare Integer fryCount = 0

Declare Integer sodaCount = 0

End Module

Module resetVariables (Real Ref totalBurger, Real Ref totalFry,

Real Ref totalSoda, Real Ref total,

Real Ref tax, Real Ref subtotal)

// reset variables totalBurger = 0 totalFry = 0 totalSoda = 0 total = 0

tax = 0

subtotal = 0

End Module

Module getBurger(Real Ref totalBurger, Integer burgerCount)

Display "Enter the number of burgers you want"

Input burgerCount

Set totalBurger = totalBurger + burgerCount \* .99

End Module

Module getFry(Real Ref totalFry, Integer fryCount)

Display "Enter the number of fries you want" Input fryCount

Set totalFry = totalFry + fryCount \* .79

End Module

Module getSoda(Real Ref totalSoda, Integer sodaCount)

Display "Enter the number of sodas you want"

Input sodaCount

Set totalSoda = totalSoda + sodaCount \* 1.09

End Module

Module calcTotal(Real totalBurger, Real totalFry, Real totalSoda,

Real Ref total, Real subtotal, Real tax)

Set subtotal = totalBurger + totalFry + totalSoda

Set tax = subtotal \* .06

Set total = subtotal + tax

End Module

Module printReceipt(Real total)

Display "Your total is $", total

End Module

**The Flowchart**

## 

## 

**The Python Code**

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

# Name: David White

# Class: CIS-1400

# Assignment: Lab 5-5

# File: lab5-5.py

# Purpose: Calculate the cost of a meal

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

print('\n\*\*\*David White\*\*\*\n') # Display author's name

# main function

def main():

endProgram = 'n'

endOrder = 'n'

totalBurger = 0

totalFry = 0

totalSoda = 0

tax = 0

subtotal = 0

total = 0

while endProgram == 'n':

resetVariables(totalBurger, totalFry, totalSoda, total, tax, subtotal)

while endOrder == 'n':

print('Enter 1 for Yum Yum burger')

print('Enter 2 for Grease Yum fries')

print('Enter 3 for Soda Yum')

option = int(input('Enter order number: '))

if option == 1:

totalBurger = getBurger()

elif option == 2:

totalFry = getFry()

elif option == 3:

totalSoda = getSoda()

endOrder = str(input("Do you want to end the order? y/n: "))

total = calcTotal(totalBurger, totalFry, totalSoda)

printReceipt(total)

endProgram = str(input('Do you want to end the program? y/n: '))

# var reset function

def resetVariables(totalBurger, totalFry, totalSoda, total, tax, subtotal):

totalBurger = 0

totalFry = 0

totalSoda = 0

total = 0

tax = 0

subtotal = 0

return totalBurger, totalFry, totalSoda, total, tax, subtotal

# get burger info

def getBurger():

burgerCount = int(input("Enter the number of burgers you want: "))

totalBurger = burgerCount \* .99

return totalBurger

# get fry info

def getFry():

fryCount = int(input("Enter the number of fries you want: "))

totalFry = fryCount \* .79

return totalFry

# get soda info

def getSoda():

sodaCount = int(input("Enter the number of sodas you want: "))

totalSoda = sodaCount \* 1.09

return totalSoda

# calculate Total

def calcTotal(totalBurger, totalFry, totalSoda):

subtotal = totalBurger + totalFry + totalSoda

tax = subtotal \* .06

total = subtotal + tax

return total, tax, subtotal

# show receipt

def printReceipt(total):

print('Your total is $', total)

return

# call main

main()

## 