1050 Programming Logic

Lab 4 (20 points total)

Name: Matthew Mosinski

*Paste your code and screenshots below.*

1. Describe the four basic elements of the counter-controlled repetition (2 points).

The four basic element of counter-controlled repetition are:

1. A control variable for loop counter.
2. The initial value of the control variable.
3. The increment (or decrement) by which the control variable is modified each interaction of the loop. Ex. ++counter. This instructs the loop to increment the control variable by 1 each time it conducts the loop.
4. The loop-continuation condition the determines if the loop should continue. Ex. *While* (counter <= 10). This instructs the loop to continue until it reaches 10.

1. Compare and contrast the while and for repetition statements (1 points).

The *for* and *while* repetition statements both execute when the Boolean expression evaluate to true, but differ in many other characteristics:

For

* The declaration of the *for* statement contains the initial variable, the condition, and increment (or decrement) all in the same statement.
* The *for* loop is used when we know the number of repetitions.
* The initialization of the *for*  loop is done only one time.

While

* The declaration of the *while* statement is written in multiple statements.
* The *while* loop is used when the number of repetitions is not known.
* The initialization of the *while* loop is done for each repetition of the loop.

1. Discuss a specific example when it would be more appropriate to use a do-while statement than a while statement. Explain why (2 points).

The *do-while­* statement always executes at least once, whereas the *while* statement will not execute if the condition is false. This is because the *while* statement test the expression before evaluating the body of the loop, whereas the *do-while* ­statement does the opposite. Therefore, the *do-while* statement should be used when and outcome is a must, such as such as when dealing with negative integers. The do while loop would be the preferable loop when the user input is false, so that the loop starts again and the user could reattempt to input something valid.

1. Create a for loop that goes from 1-100 using a variable named *i* as the counter. Each time through the loop, output whether or not the variable is even or odd
   1. Points)

*Hint:* Use and if-else statement and the modulus % operator to determine whether the variable is even or odd. Example: if ((i % 2) == 0) // it’s even

1. Use an if…else-if…else statement to output the following based on an int temp that is input by the user Prompt the user with “Please enter a temperature”.
   1. Points)

**Input output**

< 10 Polar Bear

< 20 Penguin

< 40 Moose

< 50 Reindeer

< 60 Deer

< 70 Turtle

< 80 Lion

< 90 Fish Default Bug

1. Use a switch statement to output the following based on an int input that corresponds to an exhibit at the zoo.

Prompt the user with “Please enter the exhibit number: “ (3 points).

**Input output**

* + 1. Polar Bear
    2. Penguin
    3. Moose
    4. Reindeer
    5. Deer
    6. Turtle
    7. Lion
    8. Fish
    9. Bug

1. The following code is meant to loop and output 10-20, each number on a separate line. What’s wrong? Fix the problem. (3 points)

* A dollar sign was added prior to the added quotations, to tell the compiler that the string is to be interpreted as an interpolated string.
* A necessary increment was added to tell the compiler that how to increment (or decrement) the variable.

*Corrections to the code are written in italic green:*

int i = 10;

while (*m* < 21)

{

Console.WriteLine(*$”{m}”*);

*++m;*

}

*i* was changed to *m* for this solution as it caused compiler errors in my lab 4 submission.

*Example output:*

1. *The following statement is supposed to output every number from 0-100 separated by a line with asterisks on it. What is wrong with the code? Fix it. (3 points)*

* A dollar sign was added prior to the added quotations, to tell the compiler that the string is to be interpreted as an interpolated string.
* The third line of code was eliminated as it was added to the second line of code as a line break followed by asterisks, telling the compiler to break the line with asterisk following each loop.

*Corrections to the code are written in italic green:*

for (int i = 0; i < 101; i++)

Console.WriteLine(*$”{*i*\n\*\*\*\*\*\*\*\*”*);

Console.WriteLine("\*\*\*\*\*\*\*\*"); *This line was emilimated*

*Example output:*

Shape

Description automatically generated with medium confidence

1. **Extra Credit:** Write an application that displays the following patterns separately, one below the other. Use for loops to generate the patterns. All asterisks (\*) should be displayed by a single statement of the form Console.Write( '\*' ); which causes the asterisks to display side by side. A statement of the form Console.WriteLine(); can be used to move to the next line. A statement of the form Console.Write( ' ' ); can be used to display a space for the last two patterns. There should be no other output statements in the application. [Hint: The last two patterns require that each line begin with an appropriate number of blank spaces.]

(4 Points – 1 per correct solution)

Look at page 273

