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

For each of the below questions, write a short sentence or two to express (in your own words) your answer. Keep the answers short, but use complete, correct, English sentences.

If it helps to clarify the questions, feel free to mentally prefix all the questions with the phrase "According to the video…"

1. After you’ve watched all the videos, please answer this question:  
   Of all the videos that you watched, if you could pick one video to be re-recorded by the instructor outside of class which would you choose? Why?  
   (Keep in mind the recording outside of class will omit any pauses from the instructor answering student questions, have less hemming and hawing, etc, and generally be more concise)

|  |
| --- |
| < Write your answer here |

**VIDEO: If, If/Else**

1. If you can’t remember the syntax for something in C#, you can try whatever you would have done in what other language (and it will probably (but not always) be right)?

|  |
| --- |
| Java |

1. Copy down the example of the if statement from the video (the one involving x, y, and printing “Yippee!”). For each part of the if statement put in a note about the syntax (just like the video does). This means that you should have a note about “if” being lowercase, etc.

|  |
| --- |
| (lower case) int x = 10;  (lower case) int y =20;  (lower case/ parentheses have to be there)if(x<y)  {  (block statement/ single statement) Console.WriteLine(“Yippee!”)  } |

1. Why is it useful for us to move rightwards 1 tab stop for all the statements that are “within” the if statement?

|  |
| --- |
| Its so you know that the the objects inside the if statement are being run under the if statement |

1. What is a “block statement”? Why is it useful?

|  |
| --- |
| Its something that allows for a collection of commands to be processed within a bigger statement (if,else,etc.) |

1. For the following program, EXACTLY what will be printed and why?

int x = 100;

int y = 20;

if( x < y )

Console.WriteLine(“Yippee!”);

Console.WriteLine(“Yippee!”);

Console.WriteLine(“Yippee!”);

Console.WriteLine(“End Of program”);

|  |
| --- |
| Yippee! Yippee! End of program |

1. How would you check if two variables are the same in C#?  
   How would you check if two variables are NOT the same in C#?  
   How would you check if one variable is less than another variable in C#?  
   How would you check if one variable is less than or equal to another variable in C#?

|  |
| --- |
| 1.==  2.!  3.<  4.<= |

1. Can you put an “else” clause onto anything other than an “if” statement?

|  |
| --- |
| no |

**VIDEO: Switch**

1. In light of the fact that we will NOT be doing a lot of the ‘warming up’ exercises from classes like BIT 115 (i.e., no find-and-fix errors in the homework assignments, no tracing, etc, etc), what does the instructor HIGHLY recommend that you do for each new concept that you want to use BEFORE you try to use it in something like assignment 1?

|  |
| --- |
| Create a project where you can just play around with coding. |

1. Instead of using a switch statement, how could we use if/else statements to accomplish the same goal?

|  |
| --- |
| You can make a chain of else and if’s to parse out individual inputs by using the Int32 c# |

1. Copy the example switch statement from the video (when it only has case 1 and case 2), then (briefly, intuitively) explain what will happen when the user userChoice has the value 2.

|  |
| --- |
| Console.WriteLine(“Type 1 to xxxx, type 2 to blah blah”);  In32.TryParse(Console.ReadLine(), out user Choice);  switch(userChoice)  {  Case 1:  Console.WriteLine(“You typed 1”);  Break;  Case 2:  Console.WriteLine(“You typed 2”);  Break;  } |

1. How can you have a switch statement do the same actions for when userChoice is either 0 or 1, WITHOUT duplicating code?

|  |
| --- |
| You leave the space underneath case 0 empty and you’ll get case 1’s code to trigger |

1. In C#, what data type do you typically use switch statements on?

|  |
| --- |
| numbers |

1. What limitations are there on the values after the word **case**? Can you use variables (like **case x:**)? Can you specify ranges (like **case 1 – 10:** )?

|  |
| --- |
| You can use anything that can be at some point shown as a number thus X is invalid but strings might be valid |

1. How do you catch the situation where none of the (other) cases match the variable that you’re switching on?

|  |
| --- |
| Use default |

1. Is the break statement required at the end of every case (in C#), and if so, why?

|  |
| --- |
| Break has to be used since the people who wrote the code decided that the need for it to be able to go without wasn’t necessary |

**VIDEO: Integer Division**

1. Around the 2:20 mark the video explains how to do integer division. Briefly explain (in your own words) how to do integer division:

|  |
| --- |
| Divide an integer by another |

1. What is the result of dividing x by y? How/why do you get this result?

|  |
| --- |
| 0, we get this because integer division drops everything after the decimal |

1. What triggers integer division?

|  |
| --- |
| Dividing 2 integers by each other |

1. If you wanted to divide x by 100 (in order to get, say, a percentage of the form .2 (instead of the form 20, as in 20%) ), what result would you get and why?

|  |
| --- |
| You get an error because decimals are larger than doubles |

1. Why is it ok for the compiler to automatically convert an int into a double?

|  |
| --- |
| Because a double is 64 bits and ints are 32 bits so basically there is plenty of space for it |

1. In C#, when does real division happen?

|  |
| --- |
| Real division is anytime your doing math with decimals |

1. How should you choose what data type each of your variables should be?

|  |
| --- |
| Based upon what answer your hoping for and what your putting in as variables |

1. What will the expression **x / (double) r** be (around the 8:30 mark)

|  |
| --- |
| It will be fine you will get your answer and all that changed is that you told the r to copy double then throw it out when its done |

1. What are the two names for the process demonstrated in the previous question?  
   (Hint: both name begin with “type”, as in “type \_\_\_\_\_\_\_\_”)

|  |
| --- |
| Type cast, type coercion |

1. What is one common symptom of accidentally doing integer division?

|  |
| --- |
| Setting stuff up in such a way that everything gets converted into a 0 at the end |

**VIDEO: Modulus (How does it work, mechanically)?**

1. In C#, when you’ve got a literal number with a decimal part (such as 10**.0** ), what data type is it?

|  |
| --- |
| double |

1. What problem does the following line of code have? After explaining what the problem is then re-write it so that it does not have that problem.  
   float x = 10.2;

|  |
| --- |
| If it was used it could lose precision due to being a double conversion down to a float thus, write it like 10.2f to get the most accuracy |

1. Complete this sentence: “The modulus operator (or the \_\_\_\_\_\_\_\_\_\_ operator) gets me the \_\_\_\_\_\_\_\_\_ of doing integer division”

|  |
| --- |
| The modulus operator (or the remainder operator) gets me the remainder of doing integer division |

1. In C#, what symbol is used for the modulus operator?

|  |
| --- |
| % |

1. What is the result of 21 % 10, and why?

|  |
| --- |
| 1 because mod just gives the remainder not the original number |

1. What is the result of 17 % 4, and why?

|  |
| --- |
| 25 because mod just gives the remainder not the original numberyo |

**VIDEO: Effective usage of the modulus operator**

1. One example of where the modulus / remainder operator is useful is figuring out how many items are left over after you pack equal-sized containers full of items. The video uses the specific example of packing 15 eggs into egg cartons (where each carton can contain up to 12 eggs).   
   Explain how to find out how many cartons are completely filled up, then explain how to find out how many eggs are left over after packing that 1 egg carton completely full:

|  |
| --- |
| You use integer division, than you use mod to get the remainder of the eggs |

1. How can the idea from the prior question be applied to figure out exactly which bit (which slot) to examine in the third integer? How do you know you want to look at the third integer?

|  |
| --- |
| Use the number of the bit your looking for than mod it by 32, you can divide the number by 32 |