**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_Geovanny Henein\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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)

|  |
| --- |
| The switch video because it got so confusing. |

**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)?

|  |
| --- |
| Yes, if it’s right that will excute the program if not then it will fail but you can try. |

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.

|  |
| --- |
| 1. int x = 100; 2. int y = 20; 3. if( x < y ) 4. Console.WriteLine(“Yippee!”); 5. Console.WriteLine(“Yippee!”); 6. 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?

|  |
| --- |
| It helps you see and identify them. |

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

|  |
| --- |
| A structure of the code that is grouped together. And it’s a statement |

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  Because the real indentation of the if |

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

|  |
| --- |
| No you can’t. |

**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?

|  |
| --- |
| Play around with the quick projects, and stuff to exercise. |

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

|  |
| --- |
| We can use else if |

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.

|  |
| --- |
| 1. If ( userchoice == 1 ) 2. Else if (userchoice ==2) 3. Else if (userchoice ==3) |

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

|  |
| --- |
| Switch (user choice)  Case1:  Console.writeline |

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

|  |
| --- |
| If it has more than 1 value |

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 have to repeat it, you cant do case 1-10 |

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

|  |
| --- |
| If it’s a const |

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

|  |
| --- |
| Yes, in order to activate the next one. |

**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:

|  |
| --- |
| We just divide x by y or y by x |

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

|  |
| --- |
| 0 because if we have that it does the integer division, and it drops everything after the decimal point. |

1. What triggers integer division?

|  |
| --- |
| When an integer is divided by an integer |

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?

|  |
| --- |
| It will give you an error |

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

|  |
| --- |
| Because it’s the 64 bit so it’s already done the math |

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

|  |
| --- |
| When you divide two integers, the result is always an integer. For example, the result of 7 / 3 is 2. This is not to be confused with floored division, as the / operator rounds towards zero: -7 / 3 is -2. |

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

|  |
| --- |
| Sensitive data types you should make it an integer. |

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

|  |
| --- |
| r/x |

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

|  |
| --- |
| Type casting, type corrorsion |

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

|  |
| --- |
| Usually wrong orders |

**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?

|  |
| --- |
| It’s a 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;

|  |
| --- |
| We’re telling the operator it’s a double but we have to convert it to float.  float x = 10.2f; |

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

|  |
| --- |
| Remainder |

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

|  |
| --- |
| % |

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

|  |
| --- |
| 1 because there’s only 1 left |

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

|  |
| --- |
| 1 because the remainder |

**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:

|  |
| --- |
| Well because we will know 1 egg carton takes 12 and we have 15 so 15/12=3 or 15%12=3 eggs left  1 full carton |

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?

|  |
| --- |
| Well if I want to get to bit number 72 and imagine of having an array 32 bit per integer so it will be 72%32 will get 2, because I’m trying to find individuals one. |