**Name: Patrick O’Brien**

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)

|  |
| --- |
|  |

**VIDEO: Basic Parameters**

1. For an instance method, what do I need to have (in my program) before I have can have my program call the instance method?

|  |
| --- |
| needs an object so that you can call the instance method.  Demo d = new Demo();  d.Triple(); |

1. Copy the example from the video (up to the 1:30 mark) of defining a class named Demo with a method named Triple, along with an example of calling that method from within **main**. (At this point the Triple method doesn’t actually do anything)

|  |
| --- |
| Class Demo  {  Public void Triple(int x)  {  }  }  Class Program  {  Static void Main(string[] args)  {  Demo d = new Demo();  d.Triple();  int x = 0;  }  } |

1. How do you add a parameter to the Triple method’s definition?  
   How do you pass the value 7 into the method call for Triple in **main**?

|  |
| --- |
| Add it to d.Triple(7);  Int num = 100l  d.Triple(num) |

1. Does the name of the parameter (in the method’s definition) have any connection to any variables declare in the **main** function?

|  |
| --- |
| No |

1. Briefly, intuitively describe what the “Call stack” is, and what “stack frames” are.

|  |
| --- |
| Call stack or function  Every time a function is called, a memory space is allocated for the function or local variables  Stored in a separate frame. For instance main has its own stack frame. |

1. Will changing the value of **x** in the **Triple** method ever affect the value of **num** in the **main** method?

|  |
| --- |
| Num will remain unchanged. Only the local value of x is changed. |

1. If you want to list out multiple parameters, how do you separate them? How would you specify that you want an integer parameter named **x**, and a double named **y**.

|  |
| --- |
| Separate them with commas  Public void Triple(int x, double y)  {  } |

**VIDEO: Basic Return Values**

1. Briefly, intuitively describe what “Default parameters” are used to accomplish.

|  |
| --- |
| Declare a parameter like normal, and give it a default value like 0. It allows you define a value for a parameter that it will hold by default inside the parentheses of the method. |

1. Copy down the example from the video that has the Triple method taking it’s parameter, tripling it, and then sending that new value back:

|  |
| --- |
| Public int Triple(int x)  {  Return x \* 3;  } |

1. When Visual Studio tells you that your method has the “Not all code paths return a value” error, what problem does your code have and how do you fix it?  
   (Explain this clearly **in your own words**)

|  |
| --- |
| You are missing a return statement in one of the functions that you’ve created in the method. You might add return -1 outside of the functions in the method so that the compiler is happy. |

1. If a method produces a return value are you required to make use of that return value in, say, **main**?

|  |
| --- |
| No, you are not. |

**VIDEO: 02\_String\_Compare.mp4: Q + A: String Comparison with ==, String.Compare**

1. During the execution of the following code snippet, how many string objects will be created? Once the snippet has finished, what object will **input** refer to? What object will **input2** refer to?  
   string input= “ok”;  
   string input2= “ok”;

|  |
| --- |
| There are two string objects that are created. Input and input2  Input will refer to “ok” and input2 will refer to “ok” |

1. During the execution of the following code snippet, how many string objects will be created? Once the snippet has finished, what object will **input** refer to? What object will **input2** refer to?  
   string input= (string)“ok”.Clone();  
   string input2= (string)“ok”.Clone();

|  |
| --- |
| Both input and input2 will refer to a string that is okay.  Clone can be used to create a second reference position in memory for the string object of input and assign it to input2. But in the use case for this question “ok” is created twice and cloned twice. .Clone creates a new copy of the “ok” object according to the video. |

1. In C#,what does the double-equals ( == ) operator do? Continuing the code snippet from the prior question, will the following evaluate to true or false?  
   if( input == input2 )

|  |
| --- |
| It is a comparison operator, it checks to see if two values are equal and then returns true or false. In this case in C# it asks if it is the same text. In Java it would return false since they refer to different blocks of memory. |

1. What are the three possible results that the String.Compare( string1, string2, true ) method might produce? What does each value mean?

|  |
| --- |
| -1  == 0  1  One is true, one is false, one does not have any contents. |

1. What does the ASCII standard define? How does Unicode differ from ASCII?

|  |
| --- |
| ASCII= Each number responds to a different letter  Unicode= Expanded version of ascii that can non-english languages. Same designations as ASCII but it just has more space for characters. |

1. How can you control whether String.Compare is comparing the two strings in a case-sensitive manner or a case IN-sensitive manner?

|  |
| --- |
| 1. If (0 == String.Compare(input, input2, false)) 2. { 3. Console.WriteLine(input + “ and “ + input2 + “are the same, ignoring the capitals”) 4. } else 5. { 6. They are the same or not the same including the capitals 7. } |

**VIDEO: 04\_Basic\_Array\_Stuff.mp4: Basic Array of Primitive Types**

1. Give some examples (2-3) of C#’s “primitive data types” or “simple data types”.  
   What do all primitive/simple types have in common?

|  |
| --- |
| Any data types like:  Short  Int  Long  Etc  = primitive data type 🡪 if you were to declare an instance of that type, you get space for that thing in Main. |

1. Is an array a simple type? How do we know?

|  |
| --- |
| No, it is not. It has to request separate storage space for the array itself. |

1. Briefly, intuitively, what does **the heap** contain?

|  |
| --- |
| The heap is everything that is allocated when declared using “new” |

1. In terms of the length of time that variables and objects exist, how are the stack and the heap different?

|  |
| --- |
| Stack = part of the function, quickly reclaim parts of the function, local variables, temporary  Heap = longer lived, independent of any particular function. |

1. When C# creates an array of integers, what value will C# fill the array in with? What value is used to fill in an array of doubles?

|  |
| --- |
| Just like java, it fills in the default value of 0 for ints, for doubles it would be 0.0 |

1. What is a good way to think about your computer’s memory (about your computer’s RAM)?

|  |
| --- |
| Ram is a bunch of different addresses that are ultimately temporary. Information is stored in memory can be accessed more quickly than permanent storage, so it is good for solving problems quickly.  It is a giant array of bytes starting at 0 |

1. When the instructor talks about something being located at memory address 70,000 where is that thing located in RAM (explain this using the intuitive model you described in the previous question)?

|  |
| --- |
| New goes into the heap, finds enough space at memory. Really means 70,000 byte is the first byte in the integer of the array. Each individual int is 4 bytes. Then assigns that memory address to var nums. C++ you can know which address something is stored, C# and Java it changes so we can’t know that. |

1. What is the value of nums at minute 15? What does this value represent / what does it mean?

|  |
| --- |
| Nums represents the an integer array with 5 slots. The first slot in the array is a value of int 20 |

1. Give a short C# snippet to demonstrate how you would put the value 20 into slot 0 in the array referred to by the **nums** variable.  
   Also give a short C# snippet to demonstrate how you would print out the value stored in slot 0 in the array referred to by the **nums** variable.

|  |
| --- |
| Int[] nums = new int[5];  Nums[0] = 20;  Console.Writeline(nums[0]); |

1. In C#, what is the index/address of the first element of all arrays?

|  |
| --- |
| 0 |

1. If there are five elements in the array, what is the highest valid index?  
   If there are **N** elements in the array, what is the highest valid index?

|  |
| --- |
| 4  Total number in the array -1 |

1. In C#, if you create an array with 5 slots can you assign a value to slot 10?

|  |
| --- |
| No, it would result in an index out of bounds error. |

1. In C#, can you change the size of an array? If not, then what can you do that’s equivalent to changing the size of the array?

|  |
| --- |
| No, you can however create a new array with the same contents with one or more additional spaces. |

1. Typically, when you resize an array, is it a good idea to increase the size of the array by only 1 new element? Why or why not?

|  |
| --- |
| No, it would be more efficient performance wise to double the size of the array each time. This way we can reduce the number of times the entire array has to be copied. |

1. Memory that you cannot use because you have no variables that refer to it (directly or indirectly) is called what?

|  |
| --- |
| Memory, garbage memory |

1. If C# (or Java) runs out of memory, what does it do? What is this process called?

|  |
| --- |
| Go through all the memory, squish it together, it’s called garbage collection. If there is not enough space it will crash the program (Seg fault). |

1. Why is it safe for the computer to jump directly to element 10 of an array (rather than searching through memory, slot by slot, for element 10)?

|  |
| --- |
| Because slot 10 has already been allocated to memory In the array. It can jump directly to that space in memory without causing problems. |

1. If you wanted to print how many elements are in an array, how could you do that? (Give a brief snippet of C# code)

|  |
| --- |
| For (int index = 0; index < nums.length; index++)  {  Console.WriteLine(“Slot {0} contains the value: {1}, index, nums[index]);  } |

1. Why is it useful to choose the size of an array (when we first create/allocate the array) using a variable?

|  |
| --- |
| So that the user can choose the size of the array. That way it will have enough space for the incoming data. |

1. Copy the example C# code that can print out all the elements of an array into the space below

|  |
| --- |
| 1. For (int index = 0; index < nums.length; index++) 2. { 3. Console.WriteLine(“Slot {0} contains the value: {1}, index, nums[index]); 4. } |

1. Briefly explain how your code in your answer to the previous question works.

|  |
| --- |
| The value of index is the space in the array. By iterating through the array by increasing index value by 1 each time we can print each individual element of the array according to its length. |

**VIDEO: 05\_How\_To\_Adjust\_Console\_Buffer.mp4: How To Adjust The Console's Buffer Size**

1. How can you adjust the console’s buffer size in Microsoft Windows? Assuming that the console window is open, list each of the steps that you need to go through (first click here, then click this, etc, etc)

|  |
| --- |
| Click on icon top left  Edit properties  Layout  Screen buffer size  Height as large as 9999  No larger than 10k |

**VIDEO: 06\_Array\_Params.mp4: Arrays as Parameters (syntax)**

1. When calling a method that accepts an array as a parameter, how do you print the name of the array?  
   (I.e., in main, how do you write out the name of the array)

|  |
| --- |
| Main {  Demo d = new Demo(); // Declare a new object  Double[] vals = new double[7] // Then define the name of the array  Vals[0] = 42;  Vals[4] = 13.33;  Vals[6] = 99999;  d.PrintArray(vals) // assign the array value to val and call the printArray method  } |

1. In the example, how was the array parameter declared for the PrintArray method?  
   Do you need to include square brackets in the method’s definition?

|  |
| --- |
| Yes, for the array that you’re taking as a parameter you need to declare the array in square brackets.  Public void PrintArray(double [] nums) |

1. Copy over the entire definition for the Demo class and the PrintArray method (use the version that has only the array parameter)

|  |
| --- |
| 1. Main { 2. Demo d = new Demo(); 3. Double[] vals = new double[7] 4. Vals[0] = 42; 5. Vals[4] = 13.33; 6. Vals[6] = 99999; 7. d.PrintArray(vals) 8. } 9. Class Demo 10. { 11. Public void PrintArray[double [] nums) 12. For (int I = 0l I < nums.Length; i++) 13. { 14. Console.WriteLine(nums[i]); 15. } 16. } |

**VIDEO: 07\_Pass\_By\_Reference\_Value.mp4: Arrays as Parameters (pass by reference vs. pass by value)**

1. In the example in the video at the 2:00 minute mark, what value do both **vals** and **nums** have? What does this mean in terms of which actual block of memory each one is referring to?

|  |
| --- |
| Both val and nums share the same memory address (70,000) So they share the exact same array. |

1. Because of your answer to the prior question, executing the C# code for nums[1] = 10; in the PrintArray method has what side effect?

|  |
| --- |
| It changes the value of val[1] to 10 |

1. What is the name for passing a variable this way?

|  |
| --- |
| Passing a variable by reference |

1. Fill in the blank: “When passing a simple type, you actually get a \_\_\_\_\_\_\_\_\_”

|  |
| --- |
| A copy |

1. At the 5:00 minute mark in the video, if you change the value of y what effect does that have on the variable x?

|  |
| --- |
| X is directly copied to y, but y is not directly copied to x. |

1. What is the name for passing a variable this way?

|  |
| --- |
| Simple types are passed by value, passed value to the new var, and it makes a copy. It does not affect the original. |

**VIDEO: 08\_Return\_Values.mp4: Arrays as return types**

1. Copy down the example that demonstrates how to return an array from a new MakeArray method from the video (at the 1:30 mark)

|  |
| --- |
| Public int[] MakeArray(int size)  {  Int[] newNums = new int[size];  Return newNums;  } |

1. When returning an array, do you use square brackets?   
   Give a short C# snippet to demonstrate how one might return an array named newNums:

|  |
| --- |
| No  Return newNums; |

1. Describe (briefly and intuitively) the two different types of multidimensional arrays that are available in C#. What is the name of the second type?

|  |
| --- |
| Two types   1. Giant block/matrix in memory int[,] newNums = new int[size, size]; 🡪 official multidimensional array (perfect block), directly jump – processing efficient 2. Other way, (jagged array) int[][] test = new int[5][]; 🡪 different sized rows for every rows- memory efficient, not processing |

1. Copy down the example that demonstrates how to receive an array from the MakeArray method and assign it to a variable named differentName (around the 6:00 mark)

|  |
| --- |
| Int[] differentName;  differentName = d.MakeArray(10); |

1. Given the following code snippet, how many arrays are allocated? At the end of the snippet, which array does differentName refer to?  
   int[] differentName;  
   differentName = d.MakeArray(10);  
   differentName = d.MakeArray(10);

|  |
| --- |
| Two different arrays are allocated  differentName[10] 🡪 address 80,000 (still exists in the heap, but garbage memory)  Call it second time  differentName[10] 🡪 addresss 90,000 |