**C# Homework 10**

**Question 1**

What does an array look like in memory?

**Answer**

An array is an unordered sequence of item. All the items in an array have the same type, unlike the fields in a structure or class, which can have different types. The items in an array live in a contiguous block of memory and are accessed by using an index, unlike fields in a structure or class, which are accessed by name.

**Question 2**

Where is memory allocated to hold an array, on the stack or on the heap?

**Answer**

Stored on the heap.

**Question 3**

Where is memory allocated to hold an array reference, on the stack or on the heap?

**Answer**

Stored on the stack.

**Question 4**

Can an array hold values of different types? This is a trick question, the answer is, “it depends.” What determines the types that an array can hold?

**Answer**

Depends on type of array. An array can only hold values of a specified type. An array can hold a value of different types if it is an object array.

**Question 5**

Describe the syntax of the condition for a foreach loop.

**Answer**

The foreach body must be enclosed in {} braces unless it consists of a single statement. The code in Listing 1 creates an array of odd numbers and uses foreach loop to loop through the array items and read them.

Int[] pins = {9, 3, 7, 2};

Foreach (int pin in pins)

{

Console.WriteLien(pin);

}

Pins contains the address. It is a variable and it contains a pointer that point to the first address in the contiguous block of memory

For each contains internal counter starts at 0-10 on each iteration of the for each loop it takes the internal counter and it multiplies it by the size of the type and it adds it to the memory address in pins to access the next memory address.

**Question 6**

How do you make a deep copy of an array?

**Answer**

If you need to create a deep copy of such an array, you mut use appropriate code in a for loop. Element by element.

Int[] rhay = {2, ,4, 6, 8, 10};

Foreach(int I in rhay)

Console.WriteLine(1);

Int betty = rhay.Length;

Console.WriteLin($”variable betty is {betty}”);

//making a deep copy of rhay

Int [] Anthony = new int[betty];

For(int I = 0; I < betty; i++)

Anthony[i] = rhay [i];

Console.WriteLine(“This is the deep copy of rhay, into Anthony”);

Foreach(int j in Anthony)

Console.WriteLine(j);

**Question 7**

What is the difference in the syntax between a multi-dimensional array and an array of arrays?

**Answer**

To create multi-dimensional array you need to use comma inside the square brackets. There is nothing inside the square brackets for Array of arrays.

Array of Arrays

Int [] []

Multi-Dimensional Arrays (always gives you a SQUARE array)

Int [ , ]

**Question 8**

What is the difference in the uses for a multi-dimensional array and an array of arrays? In other words, what determines whether you would use one as opposed to the other?

**Answer**

To create multi-dimensional array you need to use comma inside the square brackets. There is nothing inside the square brackets for Array of arrays.

Array of Arrays

Int [] []

Multi-Dimensional Arrays (always gives you a SQUARE array)

Int [ , ]

**Question 9**

How do you “flatten” a multidimensional array? In other words, take something that looks like a matrix and turn it into an array [1,2,3,4,5,6,7,8,9]? Write the code to do this in English.

**Answer**

Use Nested loops.

For(int I = 0; I < outer.Length; I ++)

For(int j = 0; , < inner.Length; j++)

Console.WriteLine(I, j);

0.0, 0.1, 0.2, 1.0, 1.1, 1.2,

**Question 10**

(Thought question) When we use a for loop, we can change the values of the array elements inside the loop. When we use a foreach loop, we cannot change the values of the array elements inside the loop. Why not? How is for different from foreach?

**Answer**

A foreach loop Give you access to the values. A for loop give you access to the location where the value is stored