1. **What does an array look like in memory?**

Arrays are reference types, regardless of the type of their elements. This means that an array variable refers to a contiguous block of memory holding the array elements on the heap, just as a class variable refers to an object on the heap.

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

Heap

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

Stack. When you declare an array variable, you do not declare its size and no memory is allocated other than to hold the reference on the stack. The array is given memory on the heap only when the instance is created, and this is also the point at which you specify the size.

1. **Can an array hold values of different types? This is a trick question, the answer is, “It depends.” Explain.**

In general, the elements that you attempt to store in an array must match the type when you declare the array. Additionally, the compiler can infer the type of elements in the array, as in the example below, but all of the elements must be the same otherwise it results in a compile-time error. However, implicitly typed arrays can create an array of anonymous objects that hold different types, but the fields in the anonymous types must be the same for each element of the array. For example:

//implicitly typed array  
var names = new[] { new { Name = “John”, Age = 53 },  
 new { Name = “Diana”, Age = 26 } };

//compiler infers data type – all elements must be same type  
var names = new [] {“John”, “Diana”}; // compiler determines ‘names’ to be array of strings

1. **Describe the syntax of the condition or a foreach loop.**

The foreach statement declares an iteration variable (in this example, int pin) that automatically acquires the value of each element in the array. The type of this variable must match the type of the elements in the array. For example:

int[] pins = { 9, 3, 7, 2 };  
foreach (int pin in pins)  
{  
 Console.WriteLine(pin);  
}

Notes about foreach:

■ A foreach statement always iterates through the entire array. If you want to iterate through only a known portion of an array (for example, the fi rst half) or bypass certain elements (for example, every third element), it’s easier to use a for statement.

■ A foreach statement always iterates from index 0 through index Length – 1. If you want to iterate backward or in some other sequence, it’s easier to use a for statement.

■ If the body of the loop needs to know the index of the element rather than just the value of the element, you have to use a for statement.

■ If you need to modify the elements of the array, you have to use a for statement. This is because the iteration variable of the foreach statement is a read-only copy of each element of the array.

//using var in foreach loop when array contains anonymous objects:

Foreach (var familyMember in names)  
{  
 Console.WriteLine($”Name: {familyMember.Name}, Age: {familyMember.Age}”);  
}

1. **How do you make a deep copy of an array?**

The Clone, CopyTo, and Copy methods all create a shallow copy of an array. If the elements in the array being copied contain references, the Clone method simply copies the references rather than the objects being referred to. After copying, both arrays refer to the same set of objects. If you need to create a deep copy of such an array, you must use appropriate code in a for loop. For example:

int[] pins = { 9, 3, 7, 2 };  
int[] copy = new int[pins.Length];  
for (int i = 0; i < pins.Length; i++)  
{  
 copy[i] = pins[i];  
}

1. **What is the difference between a multi-dimensional array and an array of arrays?**

Multi-dimensional arrays are often referred to as rectangular arrays since each dimension has a regular shape. However, multi-dimensional arrays can consume lots of memory, and if the application uses only some of the data in each column, then allocating memory for unused elements is a waste. The solution is an array of arrays, which has only a single dimension, but the elements in that dimension are themselves arrays.

1. **How do you “flatten” a multidimensional array? In other words, take something that looks like a matrix**

**1 2 3   
4 5 6   
7 8 9**

**and turns it into an array [ 1 2 3 4 5 6 7 8 9 ] ?**

The matrix is a multidimensional array that might look like the following:

int[,] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

to turn it into an array, you would need to use a for loop