```
class Program
{
  /*
   * Plus One:
   * Given a non-empty array of digits representing a non-negative integer, plus one to the integer.
   * The digits are stored such that the most significant digit is at the head of the list,
   * and each element in the array contain a single digit.
   * You may assume the integer does not contain any leading zero, except the number 0 itself.
   *
   * Example:
   * Input: [1,2,3]
   * Output: [1,2,4]
   * Explanation: The array represents the integer 123.
   */
  static void Main(string[] args)
  {
    int[] inputArr = { 1, 2, 3 };
    int[] inputArr2 = { 0, 3, 2, 1 };
    int[] inputArr3 = { 9, 9, 9 };

    Console.WriteLine("What is the answer to the first given array?");
    foreach (var item in PlusOne(inputArr))
      Console.Write($"{item}");

    Console.WriteLine("\n\nWhat is the answer to the second given array?");
    foreach (var item in PlusOne(inputArr2))
      Console.Write($"{item}");

    Console.WriteLine("\n\nWhat is the answer to the third given array?");
    foreach (var item in PlusOne(inputArr3))
      Console.Write($"{item}");
  }
  /*
   * Approach
   * If any leading zero, except the number 0 itself.
   * If not, plus one to the last element.
   * What if the given input array is {9,9,9}?
   */
  public static int[] PlusOne(int[] nums)
  {
    if(nums[0] == 0)                    // If any leading digit is zero,
      nums = nums.Skip(1).ToArray();    // then skip the first element.

    nums[nums.Length - 1]++;            // If not, plus one to the last element.

    for (int i = nums.Length - 1; i > 0; i--)
    {
      if (nums[i] >= 10)                // If the given array is {9,9,9},
      {
        nums[i - 1]++;                  // the result should be {10,0,0}
        nums[i] = (nums[i] % 10);
      }
    }

    if(nums[0] >= 10)                   // It does not matter to be honest,
    {                                   // but, if you want {1,0,0,0} rather than {10,0,0}
      int[] firstDigit = new int[1];
      firstDigit[0] = 1;
      nums[0] = (nums[0] % 10);

      return firstDigit.Concat(nums).ToArray();
    }
    return nums;                        // Time Complexity = O(n), Space Complexity = O(1)
  }
}
```