```
class Program
  {
    /*
     * Rotate Array
     *
     * Given an array, rotate the array to the right by k steps, where k is non-negative.
     *
     * Example:
     * Input: nums = [1,2,3,4,5,6,7], k = 3
     * Output: [5,6,7,1,2,3,4]
     *
     * Explanation:
     * rotate 1 steps to the right: [7,1,2,3,4,5,6]
     * rotate 2 steps to the right: [6,7,1,2,3,4,5]
     * rotate 3 steps to the right: [5,6,7,1,2,3,4]
     */
    static void Main(string[] args)
    {
      int[] inputArry = { 1, 2, 3, 4, 5, 6, 7 };
      int rotateNum = 3;

      Console.WriteLine("What is the answer to the given question?");
      foreach(var items in RotateArray01(inputArry, rotateNum))
      {
        Console.Write($"{items}");
      }
    }

    /*
     * Approach 01 - Brute Force
     * This approach is the way to rotate one-by-one, which is the simplest.
     * 1. Store arr[0] in a temporary variable "temp"
     * 2. Move arr[1] to arr[0], arr[2] to arr[1]... and finally temp to arr[n-1]
     */
    public static int[] RotateArray01(int[] arr, int rotateNum)
    {
      int temp;
      int previous;

      for(int i=0; i < rotateNum; i++)       // Time Complexity = rotateNum (k)
      {
        previous = arr[arr.Length - 1];     // previous = 7

        for(int j=0; j < arr.Length; j++)   // Time Complexity = n
        {
          temp = arr[j];        // j=0, temp = 1    | j=1, temp = 2     | j=2, temp = 3
          arr[j] = previous;    // arr[0] = 7       | arr[1] = 1        | arry[2] = 2
          previous = temp;      // previous = 1     | previous = 2      | previous = 3
        }
      }

      return arr;                       // Time Complexity = O(n+k), Space Complexity = O(1)
    }
```

```
    /*
     * Approach 02 - Using Extra Array
     * Will copy the new array to the original one.
     * The solution is that the number at index i in the original array is placed at the index (i+k)%length of array.
     */
     public static int[] RotateArray02(int[] arr, int k)
    {
        int[] newArr = new int[arr.Length];

        for(int i=0; i < arr.Length; i++)          //Time complexity = n
        {
            newArr[(i + k) % arr.Length] = arr[0];  //newArr[3]=1, newArr[4]=2, newArr[5]=3, newArr[6]=4
                                                    //newArr[0]=5, newArr[1]=6, newArr[2]=7
        }

        for(int i=0; i < arr.Length; i++)          //Time complexity = n
        {
            arr[i] = newArr[i];                 //copy the new arr to the original arr
        }

        return newArr;                          //Time complexity = n, Space Complexity = n
    }

}
```