```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AlgorithmsHW3
{
    class Program
    {
        static void Main(string[] args)
        {
            // Exercise 4.4 #10/A
            // Assuming pile always contains single fake coin
            int n = 0;
            Pile pile;
            Guid fakeCoin;
            Console.Write("Please provide whole positive number of coins: ");
            n = Convert.ToInt32(Console.ReadLine());
            pile = new Pile(n);
            Console.WriteLine();
            fakeCoin = pile.FindCoin(pile, n);
            Console.WriteLine("The fake coin has an id of {0}.", fakeCoin.ToString());
            Console.ReadLine();

            // Exercise 5.1 #2/A
            int[] array = null;
            MinMax.max = int.MinValue;
            MinMax.min = int.MaxValue;
            n = 0;
            Console.Write("Please provide whole positive number for array of numbers: ");
            n = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            array = ReturnPopulatedArray(n);
            DivideConquerArray(array, 0, array.Length);
            Console.WriteLine("The maxiumum is {0} and the minimum is {1}.", MinMax.max, MinMax.min);
            Console.ReadLine();
        }

        // Exercise 5.1 #2/A (cont.)
        // holds static min and max
        public static class MinMax
        {
            public static int min {get; set;}
            public static int max {get; set;}
        }

        // Exercise 5.1 #2/A (cont.)
        // populates array of size n with random numbers
        static int[] ReturnPopulatedArray(int n)
        {
            Random randomNumGen = new Random();
            int randomNum = 0;
            int[] array = new int[n];

            Console.Write("Initial array order: ");

            for (int i = 0; i < n; i++)
            {
                randomNum = randomNumGen.Next(1000, 9999);
                array[i] = randomNum;
                Console.Write(" {0} ", randomNum);
            }
            Console.WriteLine();
            return array;
        }
```

```
        // Exercise 5.1 #2/A (cont.)
        // divides array into subarrays, then updates max and min as it exits recursively
        static void DivideConquerArray(int[] array, int lower, int upper)
        {
            if (upper - lower > 2)
            {
                DivideConquerArray(array, lower, (lower + upper)/2);
                DivideConquerArray(array, (lower + upper)/2, upper);
            }
            else
            {
                int[] tempArray;
                if (upper - lower == 1)
                {
                    tempArray = new int[2];
                    tempArray[0] = array[0];
                    tempArray[1] = array[0];
                }
                else
                {
                    int tempIndex = 0;
                    tempArray = new int[upper - lower];
                    for (int i = lower; i < upper; i++)
                    {
                        tempArray[tempIndex] = array[i];
                        tempIndex++;
                    }
                }
                MinMax.max = GetMax(MinMax.max, GetMax(tempArray[0], tempArray[1]));
                MinMax.min = GetMin(MinMax.min, GetMin(tempArray[0], tempArray[1]));
            }
        }
        // Exercise 5.1 #2/A (cont.)
        // returns max value of two ints
        static int GetMax(int first, int second)
        {
            if (first > second)
            {
                return first;
            }
            return second;
        }
        // Exercise 5.1 #2/A (cont.)
        // returns max value of two ints
        static int GetMin(int first, int second)
        {
            if (first < second)
            {
                return first;
            }
            return second;
        }
    }
}
```