```
 1  using System;
 2  using System.Diagnostics;
 3  using System.Collections.Generic;
 4  using System.Linq;
 5
 6  namespace Homework1
 7  {
 8      class Program
 9      {
10          enum Sortings { None, Insertion, Quick };
11
12          //Presets
13          const int MIN_RANDOM = 1;
14          const int MAX_RANDOM = 10;
15          const Sortings SORT_FUNCTION = Sortings.Quick;
16
17
18          static void Main(string[] args)
19          {
20              //Test();
21              //Environment.Exit(0);
22              var input = GetSizeInput(args.ElementAtOrDefault(0));
23
24              if (input == -1)
25              {
26                  RunSequence();
27              }
28              else
29              {
30                  PrintHeaders();
31                  RunIteration(input, Sortings.None);
32              }
33          }
34
35          static void PrintHeaders()
36          {
37              Console.Write(" Arr. Len., Time (ms),");
38              for (int i = MIN_RANDOM; i < MAX_RANDOM + 1; i++)
39              {
40                  Console.Write("{0,10},", i);
41              }
42              Console.WriteLine();
43          }
44
45          static void RunSequence()
46          {
47              Console.WriteLine("No Sorting, 10-100M numbers:");
48              PrintHeaders();
49              for (int i = 1; i <= 10; i++)
50              {
51                  RunIteration(i * 10000000, Sortings.None);
52              }
```

```
53
54              Console.WriteLine("Insertion Sorting, 10k-100k numbers:");
55              PrintHeaders();
56              for (int i = 1; i <= 10; i++)
57              {
58                  RunIteration(i * 10000, Sortings.Insertion);
59              }
60
61              Console.WriteLine("Quick Sorting, 10k-100k numbers:");
62              PrintHeaders();
63              for (int i = 1; i <= 10; i++)
64              {
65                  RunIteration(i * 10000, Sortings.Quick);
66              }
67          }
68
69          static void RunIteration(int length, Sortings function)
70          {
71              Stopwatch stopwatch = new Stopwatch();
72              stopwatch.Start();
73              var randoms = GenerateRandomArray(length, function);
74              var hist = GenerateHistogram(randoms);
75              stopwatch.Stop();
76
77              Console.Write("{0,10},{1,10},", length,                    ↵
                  stopwatch.ElapsedMilliseconds);
78              for (int i = MIN_RANDOM; i < MAX_RANDOM + 1; i++)
79              {
80                  Console.Write("{0,10},", hist.GetValueOrDefault(i, 0));
81              }
82
83              Console.WriteLine();
84          }
85
86          static int GetSizeInput(string raw)
87          {
88              if (int.TryParse(raw, out int parsed))
89              {
90                  return parsed;
91              }
92              else
93              {
94                  return -1;
95              }
96          }
97
98          static int[] GenerateRandomArray(int length, Sortings function)
99          {
100             int[] randoms = new int[length];
101             Random factory = new Random();
102             for (int i = 0; i < randoms.Length; i++)
103             {
```

```
104                 randoms[i] = factory.Next(MIN_RANDOM, MAX_RANDOM + 1);
105             }
106 
107 
108             switch(function)
109             {
110                 case Sortings.Insertion:
111                     {
112                         InsertionSort(randoms);
113                         break;
114                     }
115                 case Sortings.Quick:
116                     {
117                         QuickSort(randoms);
118                         break;
119                     }
120             }
121             return randoms;
122         }
123 
124         static Dictionary<int, int> GenerateHistogram(int[] randoms)
125         {
126             var histogram = new Dictionary<int, int>();
127             foreach (int number in randoms)
128             {
129                 if (histogram.ContainsKey(number))
130                 {
131                     histogram[number]++;
132                 }
133                 else
134                 {
135                     histogram.Add(number, 1);
136                 }
137             }
138             return histogram;
139         }
140 
141         static void Test()
142         {
143             Console.WriteLine("Running Test:");
144             int[] randoms = GenerateRandomArray(20, SORT_FUNCTION);
145             var hist = GenerateHistogram(randoms);
146 
147             InsertionSort(randoms);
148             foreach (int i in randoms)
149             {
150                 Console.WriteLine(i);
151             }
152         }
153 
154         //Sorting Algorithms
155         static int[] InsertionSort(int[] array)
```

```
156         {
157             for (int outer = 0; outer < array.Length - 1; outer++)
158             {
159                 for (int inner = outer + 1; inner > 0; inner--)
160                 {
161                     if (array[inner - 1] > array[inner])
162                     {
163                         int swap = array[inner - 1];
164                         array[inner - 1] = array[inner];
165                         array[inner] = swap;
166                     }
167                 }
168             }
169             return array;
170         }
171 
172         // QuickSort/3 and Partition/3 were taken from:
173         // http://csharpexamples.com/c-quick-sort-algorithm-implementation/
174         // I wrote QuickSort/1
175 
176         static void QuickSort(int[] arr)
177         {
178             QuickSort(arr, 0, arr.Length - 1);
179         }
180 
181         static void QuickSort(int[] arr, int start, int end)
182         {
183             int i;
184             if (start < end)
185             {
186                 i = Partition(arr, start, end);
187 
188                 QuickSort(arr, start, i - 1);
189                 QuickSort(arr, i + 1, end);
190             }
191         }
192 
193         static int Partition(int[] arr, int start, int end)
194         {
195             int temp;
196             int p = arr[end];
197             int i = start - 1;
198 
199             for (int j = start; j <= end - 1; j++)
200             {
201                 if (arr[j] <= p)
202                 {
203                     i++;
204                     temp = arr[i];
205                     arr[i] = arr[j];
206                     arr[j] = temp;
207                 }
```

```
208             }
209
210             temp = arr[i + 1];
211             arr[i + 1] = arr[end];
212             arr[end] = temp;
213             return i + 1;
214         }
215     }
216 }
217
```