```
 1  namespace ResumenFunciones
 2  {
 3      public class Containers
 4      {
 5          // FUNCION que devuelve el numero de los valores de una lista
              mayores de 0
 6          public static int GetGreaterThanZero(List<double> list)
 7          {
 8              if(list == null)
 9              {
10                  Console.WriteLine("Excepcion: La lista es null");
11                  return 0;
12              }
13
14              if(list.Count == 0)
15              {
16                  Console.WriteLine("Excepcion: La lista no contiene
                      elementos");
17                  return 0;
18              }
19
20              int result = 0;
21
22              for (int i = 0; i < list.Count; i++)
23              {
24                  if (list[i] > 0)
25                      result++;
26              }
27              return result;
28          }
29
30          // FUNCION que devuelve el numero de los valores de un array
              mayores de 0
31          public static int GetGreaterThanZero(double[] list)
32          {
33              int result = 0;
34
35              for (int i = 0; i < list.Length; i++)
36              {
37                  if (list[i] > 0)
38                      result++;
39              }
40              return result;
41          }
42
43          // FUNCION que devuelve true si una lista contiene un numero
              dado
44          public static bool ContainsNumber(List<int> list, int number)
45          {
46              if (list == null)               //  COMPROBACION DE QUE LA
                  LISTA NO ES NULL
47                  return false;
48
```

```
49              for (int i = 0; i < list.Count; i++)
50              {
51                  if (list[i] == number)
52                      return true;
53              }
54              return false;
55          }
56
57
58          /// <summary>
59          /// Como documentar funciones:
60          /// Funcion que devuelve el mayor valor de una lista de enteros
61          /// </summary>
62          /// <param name="list">lista de enteros</param>
63          /// <returns>numero mas alto, si l es igual a null, devuelve
              int.MinValue</returns>
64          public static int GetMajor(List<int> list)
65          {
66              if (list == null || list.Count == 0)
67                  // return -1;
68                  return int.MinValue;                //  retorno de
                      valor minimo para numeros dentro de la lista
69
70              int result = list[0];
71              for (int i = 0; i < list.Count; i++)    //  cuidado con
                  inicializar el for => for(int i = 1; i < list.Count; i++)
72              {
73                  if (list[i] > result)
74                      result = list[i];
75              }
76              return result;
77          }
78
79          // FUNCION que devuelve el índice del numero mayor de una lista
80          public static int GetIndexMajor(List<int> list)
81          {
82              if (list == null || list.Count == 0)
83                  return -1;                          //  retorno de -1
                      para indices
84
85              int index = 0;
86
87              for (int i = 0; i < list.Count; i++)
88              {
89                  if (list[i] == GetMajor(list))      // recorre la lista
                       y luego la vuelve a recorrer. Esto no es del todo
                      correcto
90                      index = i;
91              }
92              return index;
93          }
94
95          public static int GetIndexMajor2(List<int> list)
```

```
96          {
97              if (list == null || list.Count == 0)
98                  return -1;
99
100             int index = 0;
101             int aux = list[0];
102             for (int i = 0; i < list.Count; i++)
103             {
104                 if (list[i] > aux)
105                 {
106                     index = i;
107                     aux = list[i];
108                 }
109             }
110             return index;
111         }
112
113
114         // FUNCION que devuelve true si una lista está ordenada de
              forma ascendente
115         public static bool IsOrdered(List<int> list)
116         {
117             if (list == null || list.Count == 0)
118                 return false;
119
120             bool result = true;
121             for (int i = 0; i < list.Count - 1; i++)
122             {
123                 if (list[i] > list[i + 1])
124                     result = false;
125             }
126             return result;
127         }
128
129
130         // FUNCION que ordena una lista de forma ascendente
131         // TODO: Ordenar una lista de forma ascendente
132         public static List<int> SortPersonal(List<int> list)
133         {
134             if (list == null || list.Count == 0)
135                 throw new Exception("Error de parametros");
136
137             List<int> result = new List<int>();
138             int aux;
139             for (int i = 0; i < list.Count - 1; i++)
140             {
141                 for (int j = list.Count - 1; j > 0; j--)
142                 {
143                     if (list[j] < list[j - 1])
144                     {
145                         aux = list[j - 1];
146                         list[j - 1] = list[j];
147                         list[j] = aux;
```

```
148                     }
149                 }
150                 result.Add(list[i]);
151             }
152 
153             return result;
154         }
155 
156         // FUNCION Binary Search con listas
157         // TODO: Binary Search
158         public static bool BinarySearch(List<int> list, int number)
159         {
160             if (list == null || list.Count == 0)
161                 return false;
162 
163             int minPosition = 0;
164             int maxPosition = list.Count - 1;
165             int midPosition;
166 
167             while (minPosition <= maxPosition)
168             {
169                 midPosition = (minPosition + maxPosition) / 2;
170 
171                 if (list[midPosition] == number)
172                     return true;
173 
174                 if (number > list[midPosition])
175                     minPosition = midPosition + 1;
176                 else
177                     maxPosition = midPosition - 1;
178             }
179             return false;
180         }
181 
182         // FUNCION Binary search con arrays
183         public static bool BinarySearch(double[] array, double number)
184         {
185             if (array == null || array.Length == 0)
186                 return false;
187 
188             int minPosition = 0;
189             int maxPosition = array.Length - 1;
190             int midPosition;
191 
192             while (minPosition <= maxPosition)
193             {
194                 midPosition = (minPosition + maxPosition) / 2;
195 
196                 if (array[midPosition] == number)
197                     return true;
198 
199                 if (number > array[midPosition])
200                     minPosition = midPosition + 1;
```

```
201                 else
202                     maxPosition = midPosition - 1;
203             }
204             return false;
205         }
206 
207         // FUNCION que devuelve el valor medio de un array
208         public static double GetAverage(double[] array)
209         {
210             if (array == null || array.Length == 0)
211                 return 0;
212 
213             double aux = 0.0;
214 
215             for (int i = 0; i < array.Length; i++)
216             {
217                 aux += array[i];
218             }
219             return aux / array.Length;
220         }
221 
222         // FUNCION que devuelve el mayor medio de los elementos de un array que superen el threshold
223         public static double GetThresholdAverage(double[] array, double Threshold)
224         {
225             if (array == null || array.Length == 0)
226                 return double.NaN;
227 
228             double aux = 0.0;
229             int count = 0;
230 
231             for (int i = 0; i < array.Length; i++)
232             {
233                 if (array[i] > Threshold)
234                 {
235                     aux += array[i];
236                     count++;
237                 }
238             }
239 
240             if (count == 0)
241                 return double.NaN;
242 
243             return aux / count;
244         }
245 
246         // FUNCION que devuelve el numero de repeticiones de valores de una lista
247         public static int GetMaxNumberRepeated(List<int> list)
248         {
249             int result = 0;
250             int maxNumber;
```

```
251             maxNumber = GetMajor(list);
252
253             for (int i = 0; i < list.Count; i++)
254             {
255                 if (list[i] == maxNumber)
256                     result++;
257             }
258             return result;
259         }
260
261         // FUNCION que ordena los elementos de una lista de forma
              inversa
262         public static List<int> GetReverseList(List<int> list)
263         {
264             if (list == null || list.Count < 0)
265                 return new List<int>();
266
267             List<int> result = new List<int>();
268
269             for (int i = list.Count - 1; i >= 0; i--)
270             {
271                 result.Add(list[i]);
272             }
273             return result;
274         }
275
276         // FUNCION reverse con arrays
277         public static int[] GetReverseList(int[] array)
278         {
279             int[] result = new int[array.Length];
280             int cont = 0;
281
282             for (int i = array.Length - 1; i >= 0; i--)
283             {
284                 result[cont] = array[i];
285                 cont++;         // ESTA MAL
286             }
287             return result;
288
289         }
290
291         // FUNCION para imprimir series
292         public static void PrintList(List<int> list)
293         {
294             for (int i = 0; i < list.Count; i++)
295             {
296                 Console.WriteLine(list[i]);
297             }
298         }
299
300         public static void PrintList(int[] array)
301         {
302             for (int i = 0; i < array.Length; i++)
```

```
303             {
304                 Console.WriteLine(array[i]);
305             }
306         }
307 
308         // FUNCION que añade un valor a un array
309         // TODO: Añadir un valor a un array
310         public static int[] AddValuetoArray(int[] array, int number)
311         {
312             if (array == null)
313             {
314                 int[] output = new int[1];
315                 output[0] = number;
316                 return output;
317             }
318             int[] result = new int[array.Length + 1];
319             for (int i = 0; i < array.Length; i++)
320             {
321                 result[i] = array[i];
322             }
323             result[array.Length] = number;
324             return result;
325         }
326     }
327 }
328 
```