```
1  // Annex I
2  // Functions required for data acquisition and processing of accelerometer data
3  // Written by Ben Hesketh
4
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10
11 namespace TFM
12 {
13     class Helper
14     {
15         /// <summary>
16         /// A function which loads a CSV text file seperated by carriage returns into a list
17         /// </summary>
18         /// <param name="filepath"></param>The desired filepath to open
19         /// <returns></returns>A list of doubles of a size equalling the number of rows
20         public static List<double> Loadfile(string filepath, int count=38000)
21         {
22             Console.WriteLine("Accelerometer Reader version 1");
23             Console.WriteLine("");
24             Console.WriteLine("Loading from file: ");
25             Console.WriteLine(filepath);
26             Console.WriteLine("");
27
28             List<double> data = new List<double>();
29
30             int counter = 0;
31             string line;
32
33             System.IO.StreamReader file =
34             new System.IO.StreamReader(filepath);
35
36            while (((line = file.ReadLine()) != null) && counter < count)
37             {
38                 data.Add(Convert.ToDouble(line));
39                 counter++;
40             }
41
42             bool isEmpty = !data.Any();
43             if (isEmpty)
44             {
45                 Console.WriteLine("CSV file failed to load.");  // error message
46             }
47             else
48             {
49                 Console.WriteLine("CSV file successfully loaded.");
50             };
51
52             Console.WriteLine("Enter any key to continue: ");
```

```
53
54              Console.ReadLine();
55
56              file.Close();
57
58              Console.WriteLine("Lowest number is {0}", data.Min());
59              Console.WriteLine("Highest number is {0}", data.Max());
60              Console.WriteLine("Average is {0}", data.Average());
61
62              return data;
63          }
64
65          /// <summary>
66          /// A function which takes a moving average of a function
67          /// </summary>
68          /// <param name="data"></param> input vector of doubles
69          /// <param name="windowSize"></param> window size, default = 10
70          /// <returns></returns> output = input vector with moving average
              applied to it
71          public static List<double> Window(List<double> data, int windowSize =
              10)
72          {
73
74              Console.WriteLine("Initiating moving datarage process with a
                  window of: ");
75              Console.WriteLine(windowSize);
76
77              var ave = Enumerable
78                    .Range(0, data.Count - windowSize)
79                    .Select(n => data.Skip(n).Take(windowSize).Average())
80                    .ToList();
81
82              Console.WriteLine("Lowest number is {0}", ave.Min());
83              Console.WriteLine("Highest number is {0}", ave.Max());
84              Console.WriteLine("Average is {0}", ave.Average());
85
86              Console.ReadLine();
87
88              return ave;
89          }
90
91          /// <summary>
92          /// Function which saves a vector of doubles to a csv file with a
              carriage return between each value
93          /// </summary>
94          /// <param name="data"></param>input vector of doubles
95          /// <param name="filepath"></param>chosen filepath
96          public static List <double> Savefile(List<double> data, string
              filepath)
97          {
98              Console.WriteLine("Accelerometer Saver version 1");
99              Console.WriteLine("");
100             Console.WriteLine("Saving to file: ");
101             Console.WriteLine(filepath);
102             Console.WriteLine("");
103
```

```
104             System.IO.StreamWriter file = new System.IO.StreamWriter
                  (filepath);
105             foreach (var element in data)
106             {
107                 file.WriteLine(Convert.ToString(element),"\r");
108             }
109
110             Console.WriteLine("Enter any key to continue: ");
111
112             Console.ReadLine();
113
114             file.Close();
115
116             return data;
117
118         }
119
120         /// <summary>
121         /// Function which takes a vector of doubles and takes the absolute of
               each value.
122         /// </summary>
123         /// <param name="data"></param>input
124         /// <returns></returns> output =|input|
125         public static List <double> Abs(List <double> data)
126         {
127             for (int i = 0; i < data.Count; i++)
128             {
129                 if (data[i]<0)
130                 {
131                     data[i] = data[i] * -1;
132                 }
133             }
134             return data;
135         }
136
137         /// <summary>
138         /// Trims the first n elements from an input vector and returns the
              cumulative total of it.
139         /// </summary>
140         /// <param name="data"></param>input vector
141         /// <param name="trim"></param> if true, trimming from the start
              occurs
142         /// <param name="trimcount"></param> n; default value = 1000
143         /// <returns></returns>
144         public static List <double> Sum (List <double> data, bool trim =
              false, int trimcount = 12250)
145         {
146             if (trim==true)
147             {
148                 data.RemoveRange(0, trimcount);
149             }
150
151             for (int i = 1; i < data.Count; i++)
152             {
153                 data[i] = data[i]+= data[i-1];
154             }
```

```
155             return data;
156         }
157 
158         /// <summary>
159         /// Function which trims (if required) and adds the X, Y and Z axis
              together to form one vector.
160         /// </summary>
161         /// <param name="X"></param >x-axis vector
162         /// <param name="Y"></param> y-axis vector
163         /// <param name="Z"></param> z-axis vector
164         /// <returns></returns> one vector of doubles representing all 3
              signals
165         public static List <double> AddAxis (List <double> X, List<double> Y,
              List <double> Z)
166         {
167             List <double> data = new List<double>();
168 
169             int count = Math.Min(Math.Min(X.Count, Y.Count),Z.Count);//if the
                vectors are different sizes count integer becomes the size of
                smallest vector.
170 
171             for (int i = 1; i<count;i++)
172             {
173                 data.Add(X[i] + Y[i] + Z[i]);
174             }
175             return data;
176         }
177 
178         /// <summary>
179         /// Score = nth element of a vector, divided by n and multiplied by
              200
180         /// </summary>
181         /// <param name="data"></param> data input
182         /// <param name="total"></param> if true, n = size of data
183         /// <param name="count"></param> n; default = 1000
184         /// <returns></returns> The score as a double
185         public static double Score(List<double> data, bool total = true, int
              count = 1000)
186         {
187 
188             int i;
189 
190             if (total == true)
191             {
192                 i = data.Count - 1;
193             }
194 
195             else
196             {
197                 i = count;
198             }
199 
200             return data[i] / i * 200;//scaling factor, k=200
201         }
202     }
203 }
```