Module 2 Assignment: Statistical Class

Design

Project requirements state that the code must create a class to track statistics for a distribution of doubles, following the given interface:

```
class Statistic
{
public:
    //Constructor: initialize private data
    Statistic();
    // add an item to the statistics
    void add(double x);
    // get average
    double average() const;
    // get Standard deviation: using the hint
    double STD() const;

private:
    // private member data
};
```

The Statistic object must be able to receive doubles one at a time, and be able to retrieve the sample mean and sample distribution after values are added.

• The sample mean for a distribution can be calculated with:

$$\mu_{\text{sample}} = \frac{\Sigma x_{i}}{n}$$

• The sample standard deviation can be calculated with

$$\sigma_{sample} = \frac{\sum x_i^2 - (\frac{\sum x_i \sum X_i}{n})}{n-1}$$

This means in order to track the mean and standard deviation, I will have to keep a tally of:

- n numPoints
- $\Sigma x_i sum$
- $\Sigma x_i^2 summedSquares$

	Statistics
- in	t numPoints
- do	ouble sum
- do	ouble summedSquares
+ v	oid add(double x)
+ d	ouble average()
+ d	ouble STD()
- ca	lculateStandardDev()

```
1 // Accepts command line args for double values
 2 // (ints promoted to doubles) and calculates the
 3 // sample mean and sample standard deviation for
 4 // the given distribution
 5
 6 #include "Statistic/statistic.hpp"
 7 #include <iostream>
 8 #include <string>
9
10 using namespace std;
12 // Function main begins statistic evaluation loop
13 int main(int argc, char *argv[])
14 {
15
       Statistic *statCalc = new Statistic();
16
17
       // Iterate over command line args
       for (int i = 1; i < argc; i++)
18
19
20
           // Add numbers to distribution
21
           try
22
           {
23
               double x = stod(argv[i]);
               statCalc->add(x);
24
25
           // Catch inputs that can't be converted to numbers
26
27
           catch (const std::invalid_argument &e)
28
               cout << "Could not interpret <" << argv[i] << "> as a double." << endl;</pre>
29
               cout << "Please update inputs to only include doubles." << endl;</pre>
30
31
               return 1;
32
           }
33
       }
34
35
       // Print out statistical info
36
       cout << "Mean: " << statCalc->average() << endl;</pre>
       cout << "Std Deviation: " << statCalc->STD() << endl;</pre>
37
       return 0;
38
39 } // End function main
```

localhost:41197 1/1

```
1 |#include <list>
 2 using namespace std;
 4 // Creates Statistic object that provides
 5 // sample mean and sample standard deviations for
 6 // provided distributions. Values are added to
 7 // the distribution as floats one at a time.
 8 class Statistic
9 {
10 public:
11
      // Uses Default empty constructor
12
      // Add given double to the distribution
13
      void add(double x);
14
15
      // Retrieve the average (sample mean) for the
16
      // current distribution
17
      double average() const;
18
19
20
      // Retrieve the sample standard deviation for
      // the current distribution
21
22
      double STD() const;
23
24 private:
25
      // Number of points added to the distribution
26
      int numPoints = 0;
      // Current total of all values in distribution
27
28
      double sum = 0;
      // Current sum of the squares of all values in distribution
29
      double summedSquares = 0;
30
31
32
      // Calculates and returns the current sample varience
      // for the distribution
33
34
      double calcSampleVarience() const;
35 };
```

localhost:40335

```
1 #include "statistic.hpp"
 2 #include <iostream>
 3 #include <cmath>
 4 using namespace std;
 5
 6 // Function to calculate and retrieve the average
 7 // (sample mean) of the current distribution
 8 double Statistic::average() const
9 {
10
      // Calculate average of values added
      if (this->numPoints > 0)
11
12
           // Integer numPoints promoted to double
13
           return this->sum / this->numPoints;
14
15
      }
      // Avoid divide by 0 error
16
17
      else
18
       {
19
           return 0;
20
21 } // End function average
22
23 // Function to calculate and retrieve the sample standard deviation of
24 // the current distribution
25 double Statistic::STD() const
26 {
27
       return sqrt(calcSampleVarience());
28 } // End function STD
30 // Function to add new value to the distribution.
31 void Statistic::add(double x)
32 {
33
      // Update data point count
      ++this->numPoints;
34
35
      // Update sums required for tracking the
36
37
      // running mean/standard dev
      this->sum += x;
38
39
      this->summedSquares += x * x;
40 } // End function add
41
42 // Function to calculate the sample varience of the current distribution
43 double Statistic::calcSampleVarience() const
44 {
45
      // Var = [\Sigma x^2 * (\Sigma x^* \Sigma x/n)] / (n-1)
       if (this->numPoints > 1)
46
47
           // Integer numPoints is promoted to double
48
           double squaredSum = this->sum * this->sum / this->numPoints;
49
           return (this->summedSquares - squaredSum) / (this->numPoints - 1);
50
51
       }
52
      // Avoid divide by 0 error
53
       else
54
       {
55
           return 0;
56
57 | // End function calcSampleVarience
```

localhost:37339 1/2

```
1 """ Runs tests comparing the Python 'statistics' to C++ class
 3 This will read lines from file 'tests.csv' stored in
 4 the same folder as this file, and calculate the sample
 5 mean and sample standard deviation using the
 6 python 'statistics' module, and the TestStatistics C++
 7 class stored in the ../ folder.
8 | " " "
9
10 import csv
11 import os
12 import statistics
13
14
15 def run_stats(values):
       """Calculate the mean and samle std dev for given values
16
17
18
      Args:
          values (list[int or float]): List of values to run stats on
19
       0.00
20
21
      try:
22
           # Attempt conversion to floats
23
           values = [float(v) for v in values]
           # Calculate statistics
24
25
           print('Mean:', statistics.mean(values))
           print('Std Deviation:', statistics.stdev(values))
26
27
      # Catch float conversion errors
       except ValueError as e:
28
29
           print(e)
30
31
32 def main():
33
       """Main function. Reads distributions from tests.csv
       and calculates stats in two different ways.
34
35
36
      # Setup working directory
37
       start_path = os.getcwd()
       os.chdir(os.path.dirname(os.path.abspath(__file__)))
38
39
      # Read 'tests.csv' in current folder
40
      with open(r'tests.csv') as file:
41
42
           csv_reader = csv.reader(file)
           for row in csv_reader:
43
               row = [value.strip() for value in row]
44
               if row:
45
                   # Print distribution to console
46
47
                   print('CSV Values:', row)
48
                   # Run python stats and print results
49
                   print('\nPython Results:')
50
                   run_stats(row)
51
52
53
                   # Run C++ stats and print results
                   print('\nC++ Results:')
54
                   test_string = '../StatisticTest ' + ' '.join(row)
55
56
                   print(test_string)
57
                   os.system(test_string)
```

localhost:34067 1/2

localhost:34067 2/2

```
1 // C++ style comments were removed for test running
 3 // Test lists of doubles
 4 1.0, 2.0, 3.0, 4.0, 5.0
 5 10.1, 12.0, 23.1, 23.0, 16.1, 21.0, 16.1
 7 // Test lists of integers
8 1, 2, 3, 4, 5
9 10, 12, 23, 23, 16, 21, 16
10
11 // Test mixed values
12 56, 18, 22, 0.5
13
14 // Test invalid characters
15 1, 2, f
16 a, b, c
17 $, ^, 4@
18
19 // Test longer list of doubles
20 // Newlines added to help with printing
21 83.2831,65.5039,14.6599,73.6619,76.3575,84.1906,79.1789,90.2266,50.2694,64.0549,26.8703,
22 28.4595, 11.3642, 23.7067, 43.0236, 30.9179, 76.0219, 76.9771, 6.4089, 11.4425, 74.7286, 15.9673,
23 26.8426,30.3849,5.306,12.0642,81.2853,35.3401,70.3817,6.4583,41.6357,17.7007,8.6751,
24 | 6.6893, 89.0175, 14.228, 84.4672, 67.0832, 22.9599, 10.5394, 61.427, 40.5194, 81.9153, 94.8238,
25 46.3909, 10.3985, 78.6326, 84.1039, 84.6922, 24.8002, 46.4194, 31.6437, 76.2617, 77.3686, 61.7926,
26 24.7442,33.3867,59.8989,47.6458,86.7503,85.2989,60.5278,15.3923,17.3211,46.4935,57.0191,
27 51.2853,47.4769,19.5403,18.9644,20.5448,3.8412,27.3385,86.3373,75.0321,13.1798,77.21,
28 20.9916, 30.6003, 21.5349, 22.6894, 1.5226, 80.0837, 63.0935, 71.3989, 43.3208, 69.7409, 6.3577,
29 23.613, 3.6974, 91.8189, 56.3994, 78.6034, 69.981, 86.1948, 89.2944, 98.4073, 51.8113, 92.3735,
30 65.1536
31
32 // Calculate stats for sequence from Mod 1 discussion
33 1, 1, 2, 5, 15, 52, 203, 877, 4140
```

localhost:34373 1/1

```
1 CSV Values: ['1.0', '2.0', '3.0', '4.0', '5.0']
 3 Python Results:
 4 Mean: 3.0
 5 Std Deviation: 1.5811388300841898
 7 C++ Results:
 8 .../StatisticTest 1.0 2.0 3.0 4.0 5.0
 9 Mean: 3
10 Std Deviation: 1.58114
11
12
13 CSV Values: ['10.1', '12.0', '23.1', '23.0', '16.1', '21.0', '16.1']
14
15 Python Results:
16 Mean: 17.34285714285714
17 Std Deviation: 5.206040447677788
18
19 C++ Results:
20 .../StatisticTest 10.1 12.0 23.1 23.0 16.1 21.0 16.1
21 Mean: 17.3429
22 Std Deviation: 5.20604
23
24
25 CSV Values: ['1', '2', '3', '4', '5']
26
27 Python Results:
28 Mean: 3.0
29 Std Deviation: 1.5811388300841898
30
31 C++ Results:
32 .../StatisticTest 1 2 3 4 5
33 Mean: 3
34 Std Deviation: 1.58114
35
36
37 CSV Values: ['10', '12', '23', '23', '16', '21', '16']
38
39 Python Results:
40 Mean: 17.285714285714285
41 Std Deviation: 5.219012860502955
42
43 C++ Results:
44 ../StatisticTest 10 12 23 23 16 21 16
45 Mean: 17.2857
46 Std Deviation: 5.21901
47
48
49 CSV Values: ['56', '18', '22', '0.5']
50
51 Python Results:
52 Mean: 24.125
53 Std Deviation: 23.21053999084611
54
55 C++ Results:
56 .../StatisticTest 56 18 22 0.5
57 Mean: 24.125
```

localhost:43933 1/3

```
58 Std Deviation: 23.2105
 59
 60
 61 CSV Values: ['1', '2', 'f']
 63 Python Results:
 64 could not convert string to float: 'f'
 65
 66 C++ Results:
 67 .../StatisticTest 1 2 f
 68 Could not interpret <f> as a double.
 69 Please update inputs to only include doubles.
 70
 71
 72 CSV Values: ['a', 'b', 'c']
 73
 74 Python Results:
 75 could not convert string to float: 'a'
 76
 77 C++ Results:
 78 .../StatisticTest a b c
 79 Could not interpret <a> as a double.
 80 Please update inputs to only include doubles.
 81
 82
 83 CSV Values: ['$', '^', '4@']
 84
 85 Python Results:
 86 could not convert string to float: '$'
 87
 88 C++ Results:
 89 \../StatisticTest $ \quad 4@
 90 Could not interpret <$> as a double.
 91 Please update inputs to only include doubles.
 92
 93
     CSV Values: ['83.2831', 05.5039', 14.0000', '179.1789', '90.2266', '50.2694', '64.0549', '26.8703', '28.4595', '11.3642', '23.7067', '43.0236', '30.9179', '76.0219', '76.9771', '6.4089', '11.4425'
 94 CSV Values: ['83.2831', '65.5039', '14.6599', '73.6619', '76.3575', '84.1906',
                                                                                  , '0.35,5 ,
'28.4595', '11.3642',
'6.4089', '11.4425',
                                                                ', '76.5.
'12.0642', '81.265
'203', '89.0175',
     '15.9673', '26.8426', '30.3849', '5.306',
                                                                               '81.2853',
                                                                                                               '70.3817',
                                  '17.7007'
                                                 '8.6751',
                                                               '6.6893',
                                                                                              '14.228', '84.4672'
     6.4583'
                   '41.6357'
     '6.4583', '41.6357', '17.7007', '8.6751', '6.6893', '89.0175', '14.228', '8
'67.0832', '22.9599', '10.5394', '61.427', '40.5194', '81.9153', '94.8238',
'10.3985', '78.6326', '84.1039', '84.6922', '24.8002', '46.4194', '31.6437'
'76.2617', '77.3686', '61.7926', '24.7442', '33.3867', '59.8989', '47.6458'
'86.7503', '85.2989', '60.5278', '15.3923', '17.3211', '46.4935', '57.0191'
'51.2853', '47.4769', '19.5403', '18.9644', '20.5448', '3.8412', '27.3385',
'75.0321', '13.1798', '77.21', '20.9916', '30.6003', '21.5349', '22.6894',
                                                                                                 '94.8238', '46.3909'
                                                                                                  '31.6437',
                                                                                  '59.8866
'46.4935', '57.846
'412', '27.3385',
                                                                                                  '47.6458',
                                                                                                  '57.0191',
     '80.0837', '63.0935', '71.3989', '43.3208', '69.7409', '6.3577', '23.613', '3.6974'
                  , '56.3994', '78.6034', '69.981', '86.1948', '89.2944', '98.4073', '51.8113'
     91.8189
     '92.3735', '65.1536'
 95
 96 Python Results:
 97 Mean: 48.534356
 98 Std Deviation: 29.286976359940972
 99
100 C++ Results:
101|../StatisticTest 83.2831 65.5039 14.6599 73.6619 76.3575 84.1906 79.1789 90.2266
     50.2694 64.0549 26.8703 28.4595 11.3642 23.7067 43.0236 30.9179 76.0219 76.9771 6.4089
     11.4425 74.7286 15.9673 26.8426 30.3849 5.306 12.0642 81.2853 35.3401 70.3817 6.4583
```

localhost:43933 2/3

```
41.6357 17.7007 8.6751 6.6893 89.0175 14.228 84.4672 67.0832 22.9599 10.5394 61.427
   40.5194 81.9153 94.8238 46.3909 10.3985 78.6326 84.1039 84.6922 24.8002 46.4194
   31.6437 76.2617 77.3686 61.7926 24.7442 33.3867 59.8989 47.6458 86.7503 85.2989
   60.5278 15.3923 17.3211 46.4935 57.0191 51.2853 47.4769 19.5403 18.9644 20.5448 3.8412
   27.3385 86.3373 75.0321 13.1798 77.21 20.9916 30.6003 21.5349 22.6894 1.5226 80.0837
   63.0935 71.3989 43.3208 69.7409 6.3577 23.613 3.6974 91.8189 56.3994 78.6034 69.981
   86.1948 89.2944 98.4073 51.8113 92.3735 65.1536
102 Mean: 48.5344
103 Std Deviation: 29.287
104
105 CSV Values: ['1', '1', '2', '5', '15', '52', '203', '877', '4140']
106
107 Python Results:
108 Mean: 588.44444444445
109 Std Deviation: 1361.823053035077
110
111 C++ Results:
112 .../StatisticTest 1 1 2 5 15 52 203 877 4140
113 Mean: 588.444
114 Std Deviation: 1361.82
```

localhost:43933 3/3