The goal of this exercise is to generate normally, exponentially and uniformly distributed random numbers and to plot histograms for the distributions using C++. We also calculate the absolute error for the generated random numbers when compared with the theoretical distributions.

1. **Generating vectors for normal N(0,1), uniform U(0,1) and exponential E(1) distributions**

I have also organized the code in a hierarchical manner with a parent class (Distribution.hpp) inherited by the three distributions. The three distributions share most attributes / characteristics and it made sense to share common functions and to enforce methods that need to be implemented for all three. To generate the random distributions, I utilized the packaged random number generators in the <random> library. Here are the code snippets of how I achieved this for the three distributions:

Normal distribution

```
void populateSample(std::vector<double> &values) {
    std::random_device rd;
    std::mt19937 gen(rd());
    std::normal_distribution<> d(mean, sqrt(variance));
    dataPoints["Mean"] = mean;
    dataPoints["Variance"] = variance;
    for (int n = 0; n < values.size(); ++n) {
        values[n] = d(gen);
    }
}
```

Uniform distribution

```
void populateSample(std::vector<double> &values) {
    std::random_device rd;
    std::mt19937 gen(rd());
    dataPoints["Begin"] = begin;
    dataPoints["End"] = end;

    std::uniform_real_distribution<> dis(begin, end);
    for (int n = 0; n < values.size(); ++n) {
        values[n] = dis(gen);
    }
}
```

Exponential Distribution

```
void populateSample(std::vector<double> &values) {
    std::random_device rd;
    std::mt19937 gen(rd());
    dataPoints["Lambda"] = lambda;
    std::exponential_distribution<> d(lambda);
    for (int n = 0; n < values.size(); ++n) {
        values[n] = d(gen);
```

```
    }
  }
```

For all the three implementation, we parse in a reference variable vector initialized with the size of the dataset desired. We then loop through as we populate the vector with random numbers.. After *Distribution#populateSample()*, the vector parsed in will be populated with the desired random numbers.

To generate random N(0,1), U(0,1) and E(1), run the script ./runDistributions.sh in the root directory. When prompted, enter desired data size, number of bins[1] and indicate that you wish for these three distributions to be generated. *StatisticsUtil::mean* and *StatisticsUtil::variance* will also be invoked and the mean and variance will be calculated as well. After execution, You will can see the generated data in the **data.txt** file at the root of the directory.

2. **Creating normalised histograms for the distributions generated and stored in the vectors in (1)**

Based on the inputs in (1) above, we generate a histogram of the data. Here is the code snippet for histogram initialization:

```
std::map histogram;
…..
void generateHistogram(std::vector<double> &values, bool normalized, int numberOfBins) {
    std::sort(values.begin(), values.end());
    const unsigned long vectorSize = values.size();
    double max = values[vectorSize - 1];
    double min = values[0];

    const double diff = max - min;
    double binWidth = diff / (numberOfBins);

    double *binValues = new double[numberOfBins];
    for (int i = 0; i < vectorSize; i++) {
       ++binValues[(int) ((values[i] - min) / binWidth)];
    }
    for (int i = 0; i < numberOfBins; ++i) {
       double minBin = min + i * binWidth;
       if (normalized) {
          binValues[i] = binValues[i] / (double) vectorSize;
       }
       histogram[minBin] = binValues[i];
    }
  }
```

[1]To generate the histogram for a given set of data points, we need to group them in bins containing a range of data. This is to ensure that the population distribution frequency can be observed.

We initialized a map, histogram that will house a list of key-value pairs of minimum value in each bin and the absolute frequency (relative frequency if normalisation is selected). We will need this histogram for part 3 as well. So we instantiate it as a global variable of the Distribution. Invoking *Distribuiton#printHistogram()* will dump the histogram in **histogram.txt** found at the root of the directory. This step will also be executed when the steps in 1 are executed. That is : *./runDistribution.sh*

3. **Calculating absolute error for fixed bin size but different sample sizes for distributions generated above:**

We observe that as the sample size increases, the absolute error (sum of the absolute difference between the theoretical distribution and the empirical distribution)decreases. This is because we get a better approximation of the distribution with a lot of data points than with a few. At the limit of infinity, I expect that the absolute error for all three distributions will approach zero. After running the code as stipulated above (*./runDistribution.sh*). The absolute error statistics will be found in **absoluteError.txt** at the root directory of the project.

4. **Generalisation of the above steps to allow for user input**

Finally, we generalize the code so that it can work for user input data. For instance, the user can select any data size, any distribution, choose whether to normalize or not, choose variance, lambda, mean etc. This makes the code more reusable and more versatile.

APPENDIX

1. Run Script

```
tituskc@tituskc-Inspiron-7537:~/Documents/Fall2016/Gits/Distributions$ ./runDistribution.sh
Build directory already exists! Do you want to recompile? [Y/N]
Y
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tituskc/Documents/Fall2016/Gits/Distributions/build
Scanning dependencies of target Distributions
[ 25%] Building CXX object CMakeFiles/Distributions.dir/Distributions.cpp.o
[ 50%] Linking CXX executable Distributions
[ 50%] Built target Distributions
Scanning dependencies of target functions
[ 75%] Building CXX object CMakeFiles/functions.dir/Distributions.cpp.o
[100%] Linking CXX shared library libfunctions.so
[100%] Built target functions
How many data points do you need?: 10000
How many bins for your histogram (recommended less than sqrt(size))?: 30
Do you want to execute for N(0,1), U(0,1) and E(1) (Y / N)? Y

See the files: histograms.txt, data.txt and absoluteError.txt for output in root directory
tituskc@tituskc-Inspiron-7537:~/Documents/Fall2016/Gits/Distributions$
```

2. Sample data from data.txt

```
Normal Distribution: Data size = 10, Mean = 0.373078, Variance = 1.58113

-1.59139, -1.32777, -0.563949, 0.26117, 0.479319, 0.635242, 0.802873, 0.820814, 1.87499, 2.33947
Uniform Distribution: Data size = 10, Mean = 0.4798, Variance = 0.11455

0.0400615, 0.120641, 0.155892, 0.296146, 0.349287, 0.482482, 0.682867, 0.852281, 0.863307, 0.95503
Exponential Distribution: Data size = 10, Mean = 0.953594, Variance = 0.63795

0.0705131, 0.0937079, 0.234404, 0.341785, 0.428273, 1.40922, 1.46509, 1.61407, 1.61882, 2.26005
~
```

3. Absolute Error stats

```
ABSOLUTE ERROR:
Normal Dist, 10000000 data points: Abs Err: 0.0441666
Exponential Dist, 10000000 data points: Abs Err: 0.00135824
Uniform Dist, 10000000 data points: Abs Err: 0.0027739
```

4. Normal Distribution in histogram.txt

```
-------------------------------------------------------
Normal Distribution
-------------------------------------------------------
Data Size: 1000
Mean = 0
Sample Mean = 0.00275974
Sample Variance = 1.046
Variance = 1

Histogram:
Min for Bin                 Frequency
    -3.08869     |* 0.002
    -2.85932     |0.001
    -2.62994     |-* 0.003
    -2.40057     |---* 0.005
    -2.17119     |------------* 0.015
    -1.94182     |------------* 0.015
    -1.71244     |------------------------------* 0.036
    -1.48307     |-------------------------------* 0.037
    -1.25369     |-------------------------------------* 0.044
    -1.02432     |-------------------------------------------------* 0.059
   -0.794943     |-------------------------------------------------------* 0.063
   -0.565568     |-----------------------------------------------------------------------* 0.083
   -0.336193     |-------------------------------------------------------------------------------------------------------* 0.114
   -0.106819     |-----------------------------------------------------------------------------------------* 0.098
    0.122556     |-------------------------------------------------* 0.059
    0.351931     |-----------------------------------------------------------------* 0.079
    0.581306     |-----------------------------------------------------------* 0.068
     0.81068     |---------------------------------------------* 0.052
     1.04006     |-------------------------------------------------* 0.058
     1.26943     |-------------------------* 0.031
      1.4988     |------------------------------* 0.036
     1.72818     |-------------* 0.017
     1.95755     |-------* 0.009
     2.18693     |----* 0.006
      2.4163     |-* 0.003
     2.64568     |--* 0.004
     2.87505     |0
     3.10443     |0.001
      3.3338     |0
     3.56318     |0.001
```

5. Uniform Distribution from histogram.txt

```
-------------------------------------------------------
Uniform Distribution
-------------------------------------------------------
Data Size: 1000
Begin = 0
End = 1
Sample Mean = 0.506286
Sample Variance = 0.0818555

Histogram:
Min for Bin                  Frequency
  0.00201176     |-------------------------------------------------* 0.035
    0.035231     |-------------------------------------------------* 0.034
   0.0684503     |-------------------------------------------------* 0.034
     0.10167     |-----------------------------------------* 0.03
    0.134889     |-------------------------------------* 0.029
    0.168108     |-------------------------------------------------* 0.036
    0.201327     |-------------------------------------* 0.029
    0.234547     |-------------------------------------* 0.028
    0.267766     |-------------------------------------------------* 0.033
    0.300985     |-------------------------------------* 0.029
    0.334205     |-----------------------------------* 0.026
    0.367424     |-----------------------------------------------------------------* 0.04
    0.400643     |-------------------------------------------------------------------* 0.041
    0.433862     |-------------------------------------------------* 0.033
    0.467082     |-----------------------------------* 0.026
    0.500301     |-------------------------------------------------* 0.035
     0.53352     |-------------------------------------* 0.028
     0.56674     |-------------------------------------------------* 0.036
    0.599959     |-------------------------------------------------------------------* 0.041
    0.633178     |-----------------------------------------------------------------* 0.04
    0.666397     |-------------------------------------------------------* 0.038
    0.699617     |-------------------------------------------------------------* 0.039
    0.732836     |-----------------------------------* 0.025
    0.766055     |-------------------------------------------------------* 0.038
    0.799275     |-------------------------------------------------* 0.032
    0.832494     |-------------------------------------------------------* 0.038
    0.865713     |-----------------------------* 0.021
    0.898932     |-------------------------------------------------------------* 0.042
    0.932152     |-------------------------------------------* 0.03
    0.965371     |-------------------------------------------------* 0.033
```

6. Exponential Distribution from histogram.txt

```
-------------------------------------------------------
Exponential Distribution
-------------------------------------------------------
Data Size: 1000
Lambda = 1
Sample Mean = 1.01596
Sample Variance = 1.04677

Histogram:
Min for Bin                 Frequency
 0.000752803     |------------------------------------------------------------------* 0.228
    0.257419     |-----------------------------------------------------* 0.178
    0.514085     |-----------------------------------------* 0.139
    0.770751     |---------------------------* 0.095
     1.02742     |---------------------* 0.074
     1.28408     |-------------* 0.048
     1.54075     |---------------* 0.055
     1.79741     |--------------* 0.051
     2.05408     |--------* 0.032
     2.31075     |----* 0.019
     2.56741     |---* 0.015
     2.82408     |----* 0.017
     3.08074     |--* 0.012
     3.33741     |* 0.006
     3.59408     |--* 0.01
     3.85074     |* 0.004
     4.10741     |0.002
     4.36407     |* 0.005
     4.62074     |0.002
     4.87741     |0.002
     5.13407     |0.001
     5.39074     |0.001
      5.6474     |0
     5.90407     |0.002
     6.16074     |0.001
      6.4174     |0
     6.67407     |0
     6.93073     |0
      7.1874     |0
     7.44406     |0
```