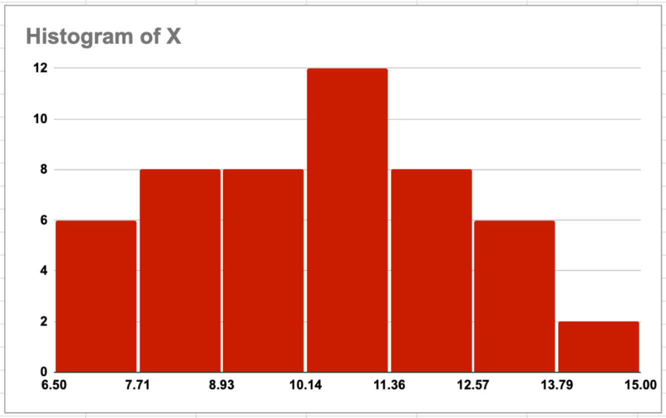
1. (10 points) Using the direct transform, generate 50 Gaussian random variables with mean 10 and variance 4, starting with the basic random number generator in Excel that generates R uniformly distributed in (0,1). Plot the histogram in Excel.

R1	R2	Z	Set Std Dev	X	X Std Dev
0.52533263	0.33406309	-0.57182829	2	8.85634341	1.991371442
0.28871807	0.15163427	0.91336179		11.82672358	
0.50651313	0.52816902	-1.14814511	**Set Mean**	7.70370978	**X Mean**
0.02048474	0.18124168	1.16759646	10	12.33519292	10.44277495
0.03214701	0.14100169	1.65857601		13.31715202	
0.40653969	0.39214776	-1.04524099		7.90951801	
0.39646369	0.57260088	-1.22118279		7.55763443	
0.41915333	0.80075468	0.41345106		10.82690213	
0.46728651	0.73153259	-0.14281202		9.71437595	
0.24436077	0.96412117	1.63627827		13.27255654	
0.99338749	0.47845248	-0.11413678		9.77172644	
0.78643685	0.69944764	-0.21648822		9.56702356	
0.48179810	0.47382261	-1.19218527		7.61562946	
0.07423099	0.34117952	-1.23624491		7.52751017	
0.25001203	0.99729166	1.66483924		13.32967848	
0.22349385	0.44958502	-1.64498329		6.71003342	
0.41520062	0.66442152	-0.67907704		8.64184591	
0.50628920	0.41669450	-1.01053304		7.97893392	
0.98419152	0.83175836	0.08772593		10.17545185	
0.42360447	0.47858885	-1.29884792		7.40230416	
0.68704574	0.63983411	-0.55298170		8.89403660	
0.43617369	0.28113406	-0.25039265		9.49921470	
0.66473831	0.20980469	0.22582195		10.45164391	
0.35025008	0.85645395	0.89822810		11.79645619	
0.81696265	0.17546603	0.28701629		10.57403257	
0.90168566	0.84990362	0.26718939		10.53437878	
0.35682020	0.01884589	1.42558144		12.85116288	



2. (10 points) Starting from a generator that generates uniform random variables between (0,1), R, determine the transformation function that needs to be applied to R, such as to generate a Weibull distribution with parameters a, b, where the Weibull cdf is given as:

$$F(x) = 1 - e^{-\left(\frac{x}{a}\right)^b}$$

Hint: use inverse transform method.

$$Let\ F(x) = R = 1 - e^{-(\frac{x}{a})^b}$$

$$Solve\ for\ x:$$

$$-R + 1 = e^{-(\frac{x}{a})^b}$$

$$ln(1 - R) = -(\frac{x}{a})^b$$

$$-(\frac{x}{a}) = \sqrt[b]{ln(1 - R)}$$

$$x = -a\sqrt[b]{ln(1 - R)}$$

3. (10 points) Assuming that for your project you get the following delay curves for two alternate design queues that you are simulating: an M/M/1 queue and an M/D/1 queue, each having the same service rate. For an arrival rate lambda = 50 packets/sec, you get the queuing delay for M/M/1 = 2 minutes, and the queueing delay for the M/D/1 = 2.2 minutes. Discuss if your results are valid, using face validity technique.
Hint: use what you know about how delay for M/M/1 and M/G/1 are related

Face Validity Assessment:

Assuming the service rate (μ) for both queueing models is the same, these results are **invalid**. With these parameters, the M/D/1 design would typically be faster than the M/M/1. Just from the fact that the M/D/1 queue yields a larger queueing delay of 2.2 minutes as opposed to the 2 minute delay of the M/M/1 queue, it can be concluded that this outcome is unlikely, and deduced as incorrect.

4. (10 points) Assume that, for your project, you ran your complex network simulation 3 times with different seeds, and collected average end-to-end delay results: 20.2 sec, 21.7 sec, 22.9 sec. Determine the point estimator for the end-to-end delay and the 90% confidence interval.

$$Point\ Estimate = \frac{20.2+21.7+22.9}{3} = 21.6 \qquad \alpha = 0.1\ at\ 90\%\ Confidence$$

$$\sigma = \sqrt{\left(\frac{1}{3(3-1)}\left((20.2 - 21.6)^2 + (21.7 - 21.6)^2 + (22.9 - 21.6)^2\right)\right)} = 0.781$$

$$t_{\frac{\alpha}{2}} = 2.92$$

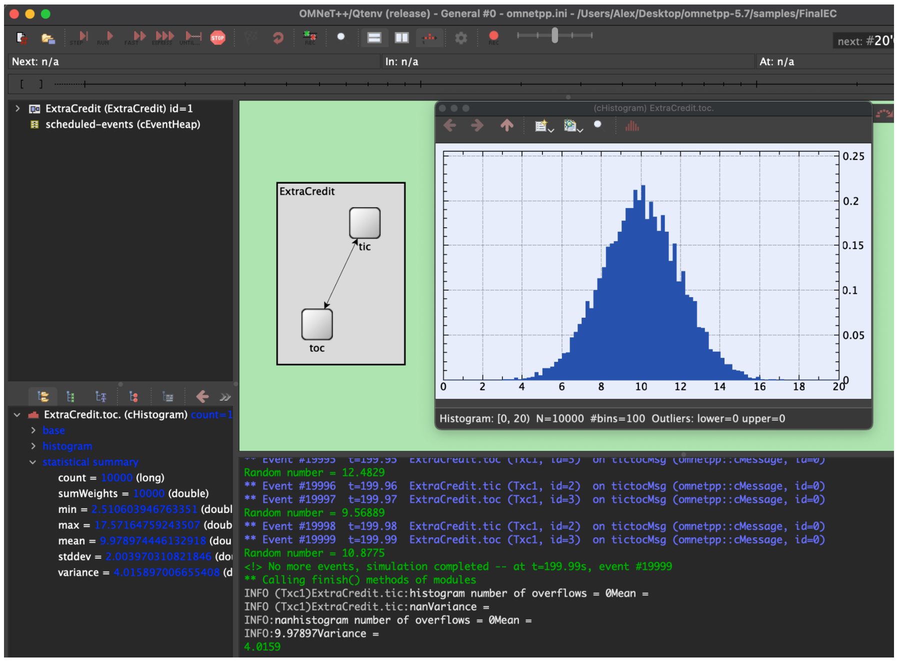
$$Interval_{Upper} = 21.6 - (2.92 \times 0.781) = 19.32$$

$$Interval_{Lower} = 21.6 + (2.92 \times 0.781) = 23.88$$

$$90\%\ Confidence\ Interval = (19.32,\ 23.88)$$

5. (15 points – Extra credit) code a tic-toc like simulation in omnet++. Tic starts sending messages. Every time Toc receives a message, it generates a Gaussian random variable with mean 10 and variance 4, and collects the random variable sample into a Histogram object, then sends the message back to Tic, which, when it receives a message, it sends it back to Toc. Run the simulation until 10000 Gaussian random variable samples are collected (Toc should stop sending back the message to Tic after the 10000th sample was generated and the simulation thus stops). Insert your code and a snapshot of the running network showing the histogram into the midterm solutions document.

Output



OMNeT++/Qtenv (release) - General #0 - omnetpp.ini - /Users/Alex/Desktop/omnetpp-5.7/samples/FinalEC
STOP
next: #20
Next: n/a
In: n/a
At: n/a
ExtraCredit (ExtraCredit) id=1
scheduled-events (cEventHeap)
ExtraCredit
tic
toc
(cHistogram) ExtraCredit.toc.
0.25
0.2
0.15
0.1
0.05
0
0
2
4
6
8
10
12
14
16
18
20

Histogram: [0, 20) N=10000 #bins=100 Outliers: lower=0 upper=0

```
ExtraCredit.toc. (cHistogram) count=1
  base
  histogram
  statistical summary
    count = 10000 (long)
    sumWeights = 10000 (double)
    min = 2.510603946763351 (doubl
    max = 17.57164759243507 (doub
    mean = 9.978974446132918 (dou
    stddev = 2.003970310821846 (do
    variance = 4.015897006655408 (d
```

```
** Event #19995  t=199.95  ExtraCredit.toc (Txc1, id=3)  on tictocMsg (omnetpp::cMessage, id=0)
Random number = 12.4829
** Event #19996  t=199.96  ExtraCredit.tic (Txc1, id=2)  on tictocMsg (omnetpp::cMessage, id=0)
** Event #19997  t=199.97  ExtraCredit.toc (Txc1, id=3)  on tictocMsg (omnetpp::cMessage, id=0)
Random number = 9.56889
** Event #19998  t=199.98  ExtraCredit.tic (Txc1, id=2)  on tictocMsg (omnetpp::cMessage, id=0)
** Event #19999  t=199.99  ExtraCredit.toc (Txc1, id=3)  on tictocMsg (omnetpp::cMessage, id=0)
Random number = 10.8775
<!> No more events, simulation completed -- at t=199.99s, event #19999
** Calling finish() methods of modules
INFO (Txc1)ExtraCredit.tic:histogram number of overflows = 0Mean =
INFO (Txc1)ExtraCredit.tic:nanVariance =
INFO:nanhistogram number of overflows = 0Mean =
INFO:9.97897Variance =
4.0159
```

Code

```cpp
#include <string.h>
#include <omnetpp.h>

using namespace omnetpp;


class Txc1 : public cSimpleModule
{
  protected:
    cStdDev stat;
    cHistogram histogram;
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
    virtual void finish();
  public:
    int count = 0;
};

Define_Module(Txc1);

void Txc1::initialize()
{
    stat.setName("Histogram");
    histogram.setRange(0,20);
    histogram.setUpBins();
    if (strcmp("tic", getName()) == 0) {
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}

void Txc1::handleMessage(cMessage *msg)
{
    if (strcmp("toc", getName()) == 0) {

        double num = normal(10,2);
        EV<<"Random number = ";
        EV<<num;
        stat.collect(num);
        histogram.collect(num);

        count++;
            if(count<10000) {
            send(msg, "out");
            } else {
                delete msg;
            }
        }
    else {
    send(msg, "out");
    }
}

void Txc1::finish()
{
  recordScalar("Mean ",stat.getMean());
  recordScalar("standard deviation ",stat.getStddev());
  recordScalar("Variance ",stat.getVariance());
  histogram.record();
  //test if the selected range was appropriate
  EV<<"histogram number of overflows = ";
  EV <<histogram.getOverflowCell();
  EV<<"Mean = \n";
  EV<<stat.getMean();
  EV<<"Variance = \n";
  EV<<stat.getVariance();
}
```

```
simple Txc1
{
    gates:
        input in;
        output out;
}

//
// Two instances (tic and toc) of Txc1 connected bo
// Tic and toc will pass messages to one another.
//
network ExtraCredit
{
    submodules:
        tic: Txc1;
        toc: Txc1;
    connections:
        tic.out --> {  delay = 10ms; } --> toc.in;
        tic.in <-- {  delay = 10ms; } <-- toc.out;
}
```