I couldn’t upload the ipynb file in the canvas, so I have taken screenshots of the code and results.
I hope this should be good, If not kindly let me know if you need ipynb I can send the code via mail.

Keerthana Golla-**kg58**

```
[43] from scipy.stats import invgamma
     from math import sqrt
     import numpy as np
```

```
[44] #getting data from data.txt
     allData = {}
     with open('data.txt', 'r') as data:
       for line in data:
         vals = [float(x) for x in line.split()]
         allData[int(vals[0])] = (vals[1], vals[2])
```

```
     m = 20.0
     c = 50.0
     sigma = 200.0
     alpha = 10.0
     beta = 1.0
     mu_zero_c = 50.0
     sigma_zero_c = 100.0
     mu_zero_m = 5.0
     sigma_zero_m = 10.0
```

```
[46] def SampleSigma(m, c):
         liScale = (sum((c + allData[i][0] * m - allData[i][1]) ** 2 for i in allData) / 2)
         piShape = alpha
         piScale = beta
         poShape = piShape + len(allData) / 2
         poScale = piScale + liScale
         sample = invgamma.rvs(a=poShape, scale=poScale)
         return sqrt(sample)
```

```
[47] print("PART 1.a. ")
     for i in range(10):
         print(SampleSigma(m, c))
```

```
PART 1.a.
1309.1513627613722
1260.6670637621069
1276.9882129722396
1271.6184421246094
1250.4141008269833
1242.3751152307775
1314.250718920492
1258.6518189047172
1282.98943809518
1282.893601525311
```

```
[48] def SampleC(m, sigma):
         liMean = (
             mu_zero_c / sigma_zero_c**2
             + len(allData) * np.mean([allData[i][1] - m * allData[i][0] for i in allData])
         ) / (1 / sigma_zero_c**2 + len(allData))
         liStd = np.sqrt(1 / (1 / sigma_zero_c**2 + len(allData)))
         return np.random.normal(liMean, liStd)
```

```
[49] print("PART 1.b. ")
     for i in range(10):
         print(SampleC(m, sigma))
```

```
PART 1.b.
-1233.4229614392057
-1233.4829841743203
-1233.4046449882785
-1233.4489438170197
-1233.476656320459
-1233.4627136097542
-1233.490081670365
```

```
[49] print("PART 1.b. ")
     for i in range(10):
         print(SampleC(m, sigma))
```

```
PART 1.b.
-1233.4229614392057
-1233.4829841743203
-1233.4046449882785
-1233.4489438170197
-1233.476656320459
-1233.4627136097542
-1233.490081670365
-1233.4619568802368
-1233.4716373925658
-1233.4642053114537
```

```
[50] def SampleM(c, sigma):

         mMeanNumerator = mu_zero_m / sigma_zero_m**2 + sum(
             allData[i][0] * (allData[i][1] - c) for i in allData
         )
         mMeanDenominator = 1 / sigma_zero_m**2 + sum(allData[i][0] ** 2 for i in allData)
         poMean = mMeanNumerator / mMeanDenominator
         poStd = np.sqrt(1 / mMeanDenominator)
         return np.random.normal(poMean, poStd)
```

```
print("PART 1.c. ")
for i in range(10):
    print(SampleM(c, sigma))
```

```
PART 1.c.
1.1401098889914127
1.1398729478859364
1.140402847072569
1.1395556505760922
1.1405531948881917
```

```
[51] print("PART 1.c. ")
     for i in range(10):
         print(SampleM(c, sigma))
```

```
PART 1.c.
1.1401098889914127
1.1398729478859364
1.140402847072569
1.1395556505760922
1.1405531948881917
1.1404618675419467
1.139788223353465
1.1405046426671213
1.1408848346960543
1.1394462682712705
```

```
[52] def getError ():
       error = 0.0
       count = 0
       for x in allData:
         y = allData[x]
         error += (c + y[0] * m - y[1]) * (c + y[0] * m - y[1])
         count += 1
       return error / count
```

```
print("PART 2)")
errors = []
for _ in range(1000):
    errors.append(getError())
    sigma = SampleSigma(m, c)
    m = SampleM(c, sigma)
    c = SampleC(m, sigma)
```

```
PART 2)
```

```
print("PART 2)")
errors = []
for _ in range(1000):
    errors.append(getError())
    sigma = SampleSigma(m, c)
    m = SampleM(c, sigma)
    c = SampleC(m, sigma)
```

```
PART 2)
```

```
[54] print("First 5 errors:", errors[:5])
```

```
First 5 errors: [1648445.9364176341, 118.04383222920956, 118.03333448181102, 118.0044782031103, 117.98404364366445]
```

```
[55] print("Last 5 errors:", errors[-5:])
```

```
Last 5 errors: [107.13555160945793, 107.13744662429939, 107.13655629283761, 107.13140953700616, 107.12511646775609]
```

```
print("Final estimates of the m,c and sigma are:")
print("m:", m)
print("c:", c)
print("sigma:", sigma)
```

```
Final estimates of the m,c and sigma are:
m: 2.205275360944926
c: -22.526581400882424
sigma: 9.954025076004749
```