Similar Examples for Exercise 1

```
1. Monte Carlo Integration for a Quadratic Inequality
val a = 0.8
val N = 1000
val count: Float = sc.parallelize(1 to N).filter { _ =>
  val x = math.random
  val y = math.random
  x * x + a * y * y < a
}.count()
println(s"The result is ${count / N}")
```

```
2. Monte Carlo for Linear Condition
val b = 1.5
val N = 10000
val count: Float = sc.parallelize(1 to N).filter { _ =>
  val x = math.random
  val y = math.random
  x - b * y > 0
}.count()
println(s"The result is ${count / N}")
```

```
3. Monte Carlo for Circle Area Estimation
val r = 1.0
val N = 5000
val count: Float = sc.parallelize(1 to N).filter { _ =>
  val x = (math.random - 0.5) * 2 * r
  val y = (math.random - 0.5) * 2 * r
  x * x + y * y <= r * r
}.count()
println(s"The estimated fraction is ${count / N}")
```

```
4. Monte Carlo with Exponential Condition
val lambda = 2.0
val N = 2000
val count: Float = sc.parallelize(1 to N).filter { _ =>
  val x = math.random
  val y = math.random
  x * y < math.exp(-lambda * (x + y))
}.count()
println(s"The result is ${count / N}")
```

Similar Examples for Exercise 2

```
1. Estimating Pi Using Square and Circle
val R = 2.0
val N = 10000
val points = sc.parallelize(1 to N)
val insideCircle: Float = points.filter { _ =>
  val x = math.random * 2 * R - R
  val y = math.random * 2 * R - R
  x * x + y * y <= R * R
}.count()
val estimatePi = (insideCircle / N) * 4
println(s"Estimated Pi: $estimatePi")
```

2. Square within a Circle

```
val R = 5.0
val N = 5000
val points = sc.parallelize(1 to N)
val count: Float = points.filter { _ =>
  val x = (math.random - 0.5) * 2 * R
  val y = (math.random - 0.5) * 2 * R
  math.abs(x) + math.abs(y) <= R
}.count()
println(s"The fraction inside is ${count / N}")
```

3. Monte Carlo Estimation for Ellipse

```
val a = 3.0
val b = 2.0
val N = 8000
val ellipsePoints: Float = sc.parallelize(1 to N).filter { _ =>
  val x = (math.random - 0.5) * 2 * a
  val y = (math.random - 0.5) * 2 * b
  (x * x) / (a * a) + (y * y) / (b * b) <= 1
}.count()
println(s"Fraction of points inside ellipse: ${ellipsePoints / N}")
```

4. Random Disk Sampling

```
val R = 1.0
val N = 10000
val points = sc.parallelize(1 to N)
val inDisk: Float = points.filter { _ =>
  val u = math.random * R
  val v = math.random * R
  math.sqrt(u * u + v * v) <= R
}.count()
println(s"Fraction of points inside the disk: ${inDisk / N}")
```

Similar Examples for Exercise 3

1. Integral of a Quadratic Function

```
val N = 100000
val delta = 1.0 / N
val integral = sc.parallelize(1 to N).map(i => i * delta).map(x => x * x)
.reduce(_ + _) * delta
println(s"Integral of x^2 from 0 to 1: $integral")
```

2. Logarithmic Integral

```
val N = 1000000
val delta = 1.0 / N
val integral = sc.parallelize(1 to N).map(i => i * delta).map(x => math.log(x + 1))
.reduce(_ + _) * delta
println(s"Integral of log(x+1) from 0 to 1: $integral")
```

3. Reciprocal Function Integral

```
val N = 1000000
val delta = 1.0 / N
val integral = sc.parallelize(1 to N).map(i => i * delta).map(1 / _)
.reduce(_ + _) * delta
println(s"Integral of 1/x from 1 to 2: $integral")
```

4. Trigonometric Integral

```
val N = 100000
val delta = math.Pi / N
val integral = sc.parallelize(1 to N).map(i => i * delta).map(x => math.sin(x))
.reduce(_ + _) * delta
println(s"Integral of sin(x) from 0 to Pi: $integral")
```