1) Create a stream of 2D points.

nc is the command which runs netcat, a simple Unix utility that reads and writes data across network connections, using the TCP or UDP protocol.

Creating input stream on localhost 6781 port using nc utility.

Input:

[satish.patil678_gmail@ip-20-0-41-53 ~]$ nc -lk 6781

4 9

-9 3

-9 -6

8 -7

0 0

0 4

7 0

324

34 76

-22 -44

0 0

2) create spark streaming context to analyse the streaming points.

Ssc is the spark streaming context with 10 second window and sc is spark context which is required for the spark streaming context to run. Ssc is the entry point for streaming which in turn uses spark context (sc).

Code:

```
import sys

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

sc = SparkContext(appName='Cordirant locating')

ssc = StreamingContext(sc,10)
```

3) Apply quadrant rule and identify points in each quadrant.

Get_quadrant() is the function which takes each line as input and identifies the points belongs to which quadrant and returns quadrant name with count as 1.

Code:

```
def get_quadrant(line):
    # Convert the input string into a pair of numbers
    try:
        (x, y) = [float(x) for x in line.split()]
    except:
        print ("Invalid input")
        return ('Invalid points', 1)

    # Map the pair of numbers to the right quadrant
    if x > 0 and y > 0:
        quadrant = 'First quadrant'
    elif x < 0 and y > 0:
        quadrant = 'Second quadrant'
    elif x < 0 and y < 0:
        quadrant = 'Third quadrant'
    elif x > 0 and y < 0:
        quadrant = 'Fourth quadrant'
    elif x == 0 and y != 0:
        quadrant = 'Lies on Y axis'
    elif x != 0 and y == 0:
        quadrant = 'Lies on X axis'
    else:
        quadrant = 'Origin'

    return (quadrant, 1)
```

lines = ssc.socketTextStream('20.0.41.53',6781)

updateFunction = lambda new_values, running_count: sum(new_values) + (running_count or 0)

running_counts = lines.map(get_quadrant).updateStateByKey(updateFunction)

4) Count the number of points falling into a specific quadrant.

updateStateByKey() is a special functions which takes list of new values adds them and in turn sums with running count to get the total count. pprint() prints the output to console.

Code:

running_counts.pprint()

ssc.start()

ssc.awaitTermination()

Output:

```
Time: 2018-08-04 18:57:10
-------------------------------------------
('First quadrant', 1)

-------------------------------------------
Time: 2018-08-04 18:57:20
-------------------------------------------
('First quadrant', 1)
('Second quadrant', 1)
('Third quadrant', 1)

-------------------------------------------
Time: 2018-08-04 18:57:30
-------------------------------------------
('First quadrant', 1)
('Origin', 1)
('Lies on Y axis', 1)
('Third quadrant', 1)
('Second quadrant', 1)
('Lies on X axis', 1)
('Fourth quadrant', 1)

-------------------------------------------
Time: 2018-08-04 18:57:40
-------------------------------------------
('First quadrant', 2)
('Origin', 1)
```

```
('Lies on Y axis', 1)
('Invalid points', 1)
('Third quadrant', 1)
('Second quadrant', 1)
('Lies on X axis', 1)
('Fourth quadrant', 1)

-----------------------------------------
Time: 2018-08-04 18:57:50
-----------------------------------------
('First quadrant', 2)
('Origin', 2)
('Lies on Y axis', 1)
('Invalid points', 1)
('Third quadrant', 2)
('Second quadrant', 1)
('Lies on X axis', 1)
('Fourth quadrant', 1)
```