This ASSIGNMENT is done by Java

1. Declare and initialize the semaphores with the appropriate values.
2. //Declare the semaphores
3. public Semaphore printers, plotters, scanners;
5. //Initialize the semaphores with the appropriate values
6. private void initSemaphores() {
7. printers = new Semaphore(5);
8. plotters = new Semaphore(6);
9. scanners = new Semaphore(4);
10. }
11. Create a routine that loops through a sequence 4 times. In each iteration the process
    1. forks a child process.

 for(int i=1; i<=4; i++) {

1. //Create a child process to request a resource
2. new Resource(rm).start();
3. //Sleep for random time between 1 to 3 seconds
4. Thread.sleep(ThreadLocalRandom.current().nextInt(1, 3 + 1)\*1000);
5. }

The child process:

* + 1. uses a random number generator (1-3) to determine which resource it will request

1. //i.uses a random number generator (1-3)
2. //to determine which resource it will request
3. int num = ThreadLocalRandom.current().nextInt(1, 3 + 1);
   * 1. uses native semaphore function or one that you create to request the appropriate resource
4. switch (num) {
5. case 1: {//1-->Printer is requested
6. System.out.println("A Printer is requested. by process " + ManagementFactory.getRuntimeMXBean().getName());
7. s = rm.printers;
8. break;
9. }
10. case 2: {//2-->Plotter is requested
11. System.out.println("A Plotter is requested. by process " + ManagementFactory.getRuntimeMXBean().getName());
12. s = rm.plotters;
13. break;
14. }
15. case 3: {//3-->Scanner is requested
16. System.out.println("A Scanner is requested. by process " + ManagementFactory.getRuntimeMXBean().getName());
17. s = rm.scanners;
18. break;
19. }
20. }
    * 1. Print the process’ PID and the requested resource type
      2. Print the process’ PID and the success/failure of the request
      3. if the resource is available - sleep for a random time between 1-3 seconds and then release the resource using appropriate the semaphore function
21. //Show the value of the corresponding semaphore
22. System.out.println("Value of the semaphore = "+s.availablePermits());
23. boolean done = false;
24. while(!done) {
25. if(s.tryAcquire()) {//Resource is available
26. System.out.println("Request is a Success!\n");
27. try {
28. //Sleep for random time between 1 to 3 seconds
29. Thread.sleep(ThreadLocalRandom.current().nextInt(1, 3 + 1)\*1000);
30. } catch (InterruptedException e) {
31. e.printStackTrace();
32. }
    * 1. if the resource is not available – sleep for a random between 2-4 seconds and repeat the request (go to step ii).
33. }else {//Resource is not available
34. System.out.println("Request is a Failure!\n");
35. try {
36. //Sleep for random time between 2 to 4 seconds
37. Thread.sleep(ThreadLocalRandom.current().nextInt(2, 4 + 1)\*1000);
38. } catch (InterruptedException e) {
39. e.printStackTrace();
40. }
41. }
    * 1. Terminate
42. s.release();
    1. Sleeps for a random time between 1-3 seconds
43. //Sleep for random time between 1 to 3 seconds
44. Thread.sleep(ThreadLocalRandom.current().nextInt(1, 3 + 1)\*1000);