The output for Task 3 may be like the following results:

```
$ java Boot
threadOS ver 2.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
shell[1]% Test2
Test2
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=2)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=2)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=2)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=2)
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=2)
Thread[b]: response time = 3990 turnaround time = 4991 execution time = 1001
Thread[e]: response time = 6990 turnaround time = 7491 execution time = 501
Thread[c]: response time = 4991 turnaround time = 7994 execution time = 3003
Thread[a]: response time = 2989 turnaround time = 7996 execution time = 5007
Thread[d]: response time = 5991 turnaround time = 12000 execution time = 6009
Test2b finished; total time = 12012
shell[2]% exit
exit
-->q
q
$
```

The Loader.java should wait for Shell.java to exit. Shell.java waits for Test2.java to complete its execution. Test2.java spawns 5 children such as Thread[a], Thread[b], Thread[c], Thread[d], and Thread[e] and thereafter waits for the termination of all these children. Program 3B does not care about response time, execution time, and total time. We focus on only this cascading thread synchronization.