Operating Systems
Assignment 3: Deadlock Detection and Resolution
Johnathon Power A00405363
Ben Jollymore A00400128

Introduction

Deadlock can occur when a series of processes or threads require a specific set of resources for execution, and these resources are shared amongst the processes. If multiple processes hold data, and must acquire held data from other processes in such a way that they form a circular dependency, the program will reach an impasse, thus creating deadlock.

Approach

The pseudocode approach for the problem is as follows.

Manager (Assignment3.c)

```
Unlink all semaphores from memory;
Open semaphores;
Validate semaphores were opened and closed correctly;

Acquire deadlock probability from command line parameter;

Read in sequence of trains;
Initialize matrix based on number of trains;

Spawn new trains in the order and direction of sequence;
sleep(1-probability);

while(train processes are alive)
    Read in matrix;
    Check matrix for deadlock;
    If deadlock;
        Unlink semaphores;
        End process on failure code;

Unlink semaphores;
End process on success code;
```

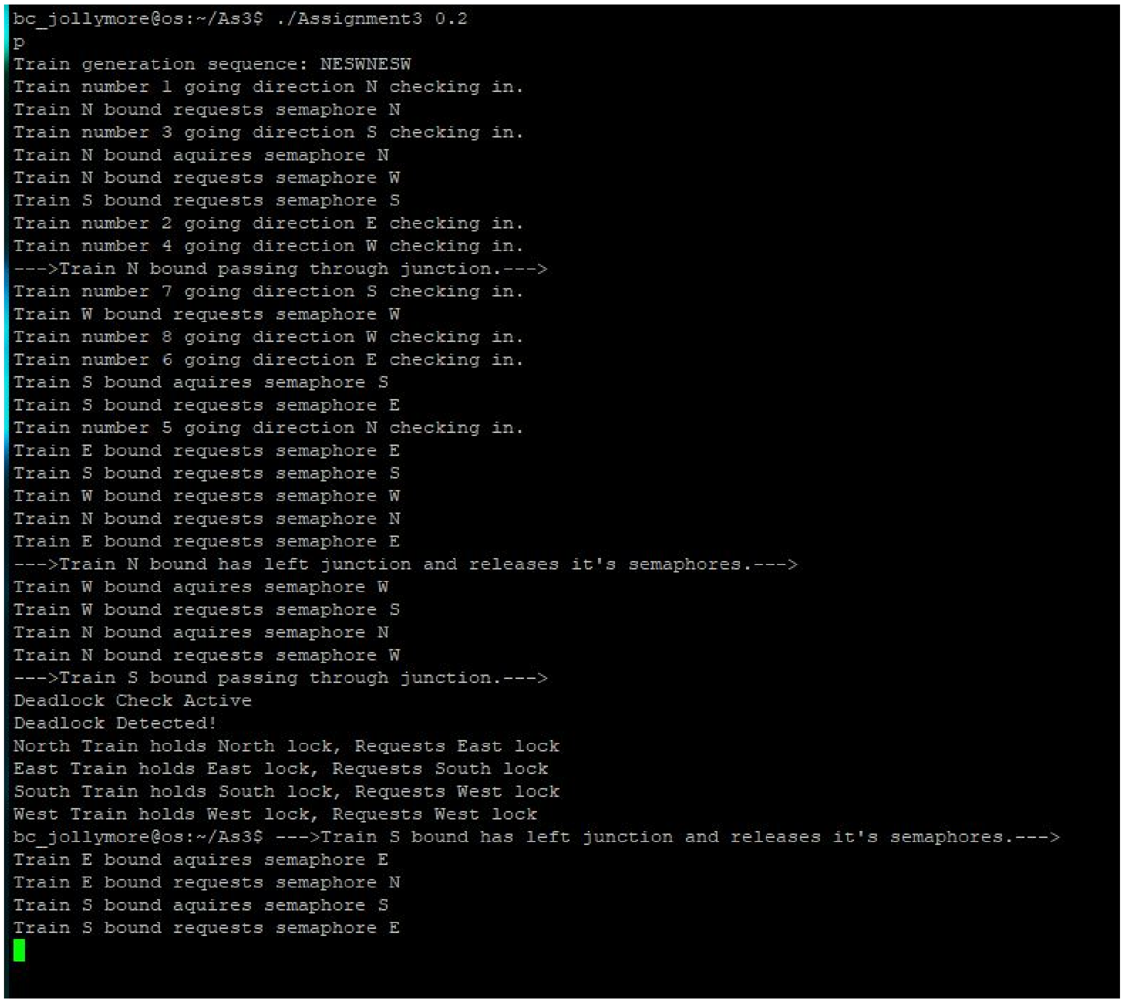
Train (newTrain.c)

```
Open file IO semaphore;

Retrieve direction and train number from command line parameter;
```

```
Get number of overall trains from matrix file;
Initalize matrix;

Open junction, direction and right line semaphores and initialize read/write controls based on
direction of train;

Validate opened semaphores;

Request directional semaphore;
Update matrix per request;
Acquire directional semaphore;
Update matrix per acquisition;

Request right semaphore;
Update matrix per request;
Acquire right semaphore;
Update matrix per acquisition;

Request junction semaphore;
Acquire junction semaphore;

Pass junction;

Release directional, junction and right semaphores;
Update matrix per release;

Terminate on success code.
```

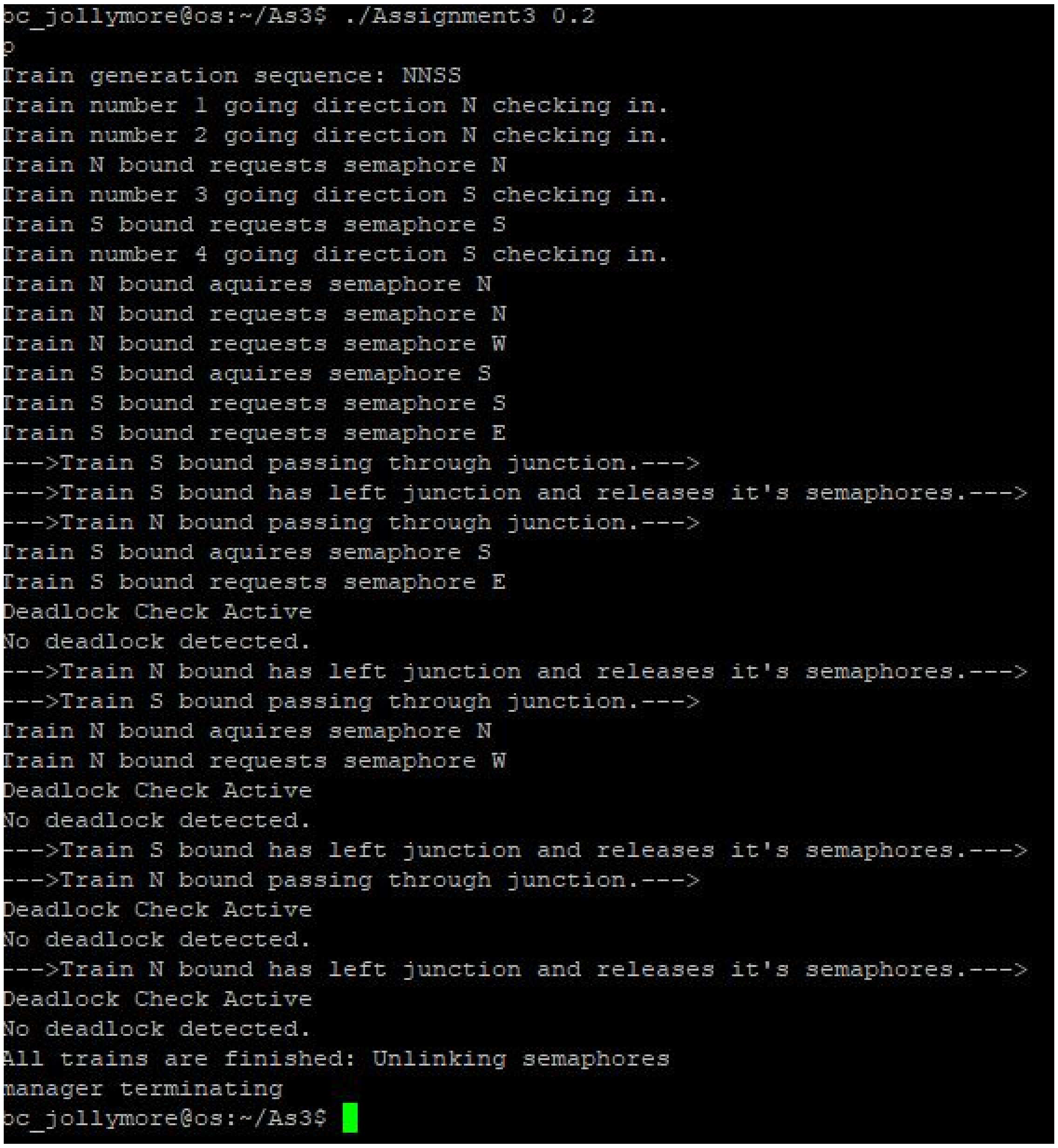
Results
All tests were done using sequence NESWNESW, which is very likely to cause a deadlock, and NNSS, which are very unlikely to cause deadlock. Some print statements are asynchronous and continue even after the cycle has been detected.

0.2 probability
NESWNESW (deadlock).



```
bc_jollymore@os:~/As3$ ./Assignment3 0.2
p
Train generation sequence: NESWNESW
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train number 3 going direction S checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train S bound requests semaphore S
Train number 2 going direction E checking in.
Train number 4 going direction W checking in.
--->Train N bound passing through junction.--->
Train number 7 going direction S checking in.
Train W bound requests semaphore W
Train number 8 going direction W checking in.
Train number 6 going direction E checking in.
Train S bound aquires semaphore S
Train S bound requests semaphore E
Train number 5 going direction N checking in.
Train E bound requests semaphore E
Train S bound requests semaphore S
Train W bound requests semaphore W
Train N bound requests semaphore N
Train E bound requests semaphore E
--->Train N bound has left junction and releases it's semaphores.--->
Train W bound aquires semaphore W
Train W bound requests semaphore S
Train N bound aquires semaphore N
Train N bound requests semaphore W
--->Train S bound passing through junction.--->
Deadlock Check Active
Deadlock Detected!
North Train holds North lock, Requests East lock
East Train holds East lock, Requests South lock
South Train holds South lock, Requests West lock
West Train holds West lock, Requests West lock
bc_jollymore@os:~/As3$ --->Train S bound has left junction and releases it's semaphores.--->
Train E bound aquires semaphore E
Train E bound requests semaphore N
Train S bound aquires semaphore S
Train S bound requests semaphore E
```

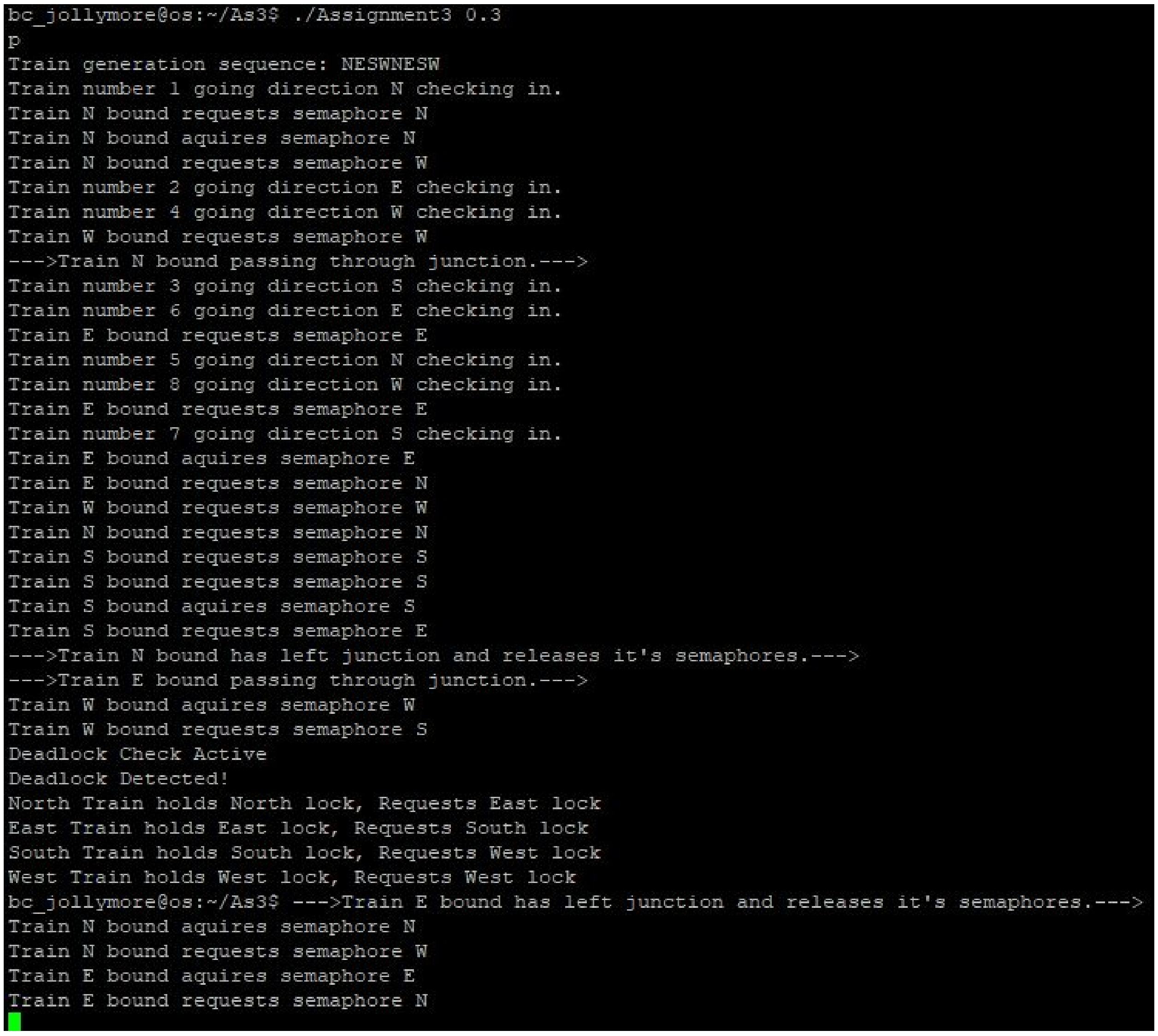
NNSS



```
bc_jollymore@os:~/As3$ ./Assignment3 0.2
p
Train generation sequence: NNSS
Train number 1 going direction N checking in.
Train number 2 going direction N checking in.
Train N bound requests semaphore N
Train number 3 going direction S checking in.
Train S bound requests semaphore S
Train number 4 going direction S checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore N
Train N bound requests semaphore W
Train S bound aquires semaphore S
Train S bound requests semaphore S
Train S bound requests semaphore E
--->Train S bound passing through junction.--->
--->Train S bound has left junction and releases it's semaphores.--->
--->Train N bound passing through junction.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
--->Train N bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
Deadlock Check Active
No deadlock detected.
All trains are finished: Unlinking semaphores
manager terminating
bc_jollymore@os:~/As3$
```

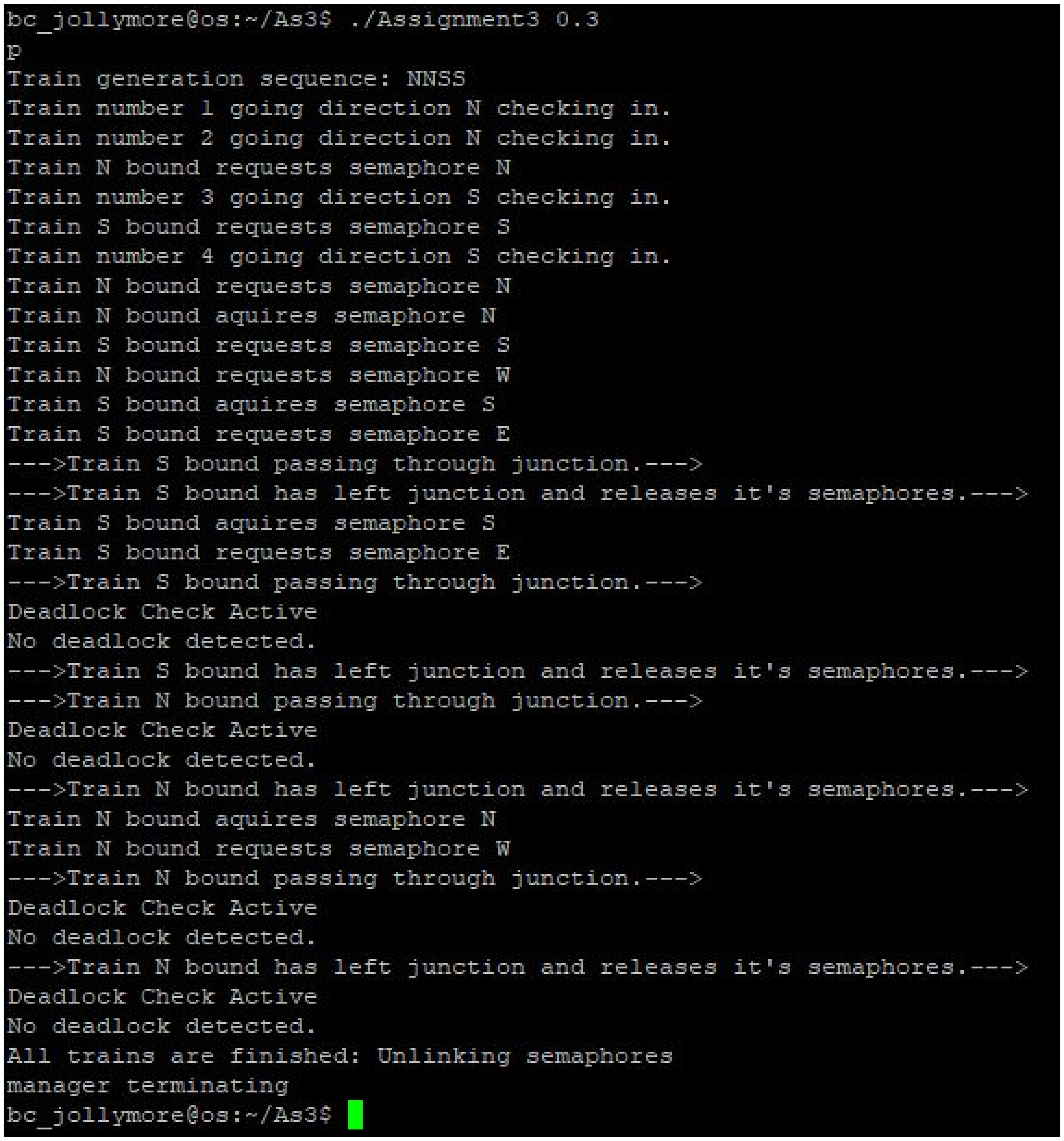
0.3 probability

NESWNESW (deadlock).



```
bc_jollymore@os:~/As3$ ./Assignment3 0.3
p
Train generation sequence: NESWNESW
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train number 2 going direction E checking in.
Train number 4 going direction W checking in.
Train W bound requests semaphore W
--->Train N bound passing through junction.--->
Train number 3 going direction S checking in.
Train number 6 going direction E checking in.
Train E bound requests semaphore E
Train number 5 going direction N checking in.
Train number 8 going direction W checking in.
Train E bound requests semaphore E
Train number 7 going direction S checking in.
Train E bound aquires semaphore E
Train E bound requests semaphore N
Train W bound requests semaphore W
Train N bound requests semaphore N
Train S bound requests semaphore S
Train S bound requests semaphore S
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train N bound has left junction and releases it's semaphores.--->
--->Train E bound passing through junction.--->
Train W bound aquires semaphore W
Train W bound requests semaphore S
Deadlock Check Active
Deadlock Detected!
North Train holds North lock, Requests East lock
East Train holds East lock, Requests South lock
South Train holds South lock, Requests West lock
West Train holds West lock, Requests West lock
bc_jollymore@os:~/As3$ --->Train E bound has left junction and releases it's semaphores.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train E bound aquires semaphore E
Train E bound requests semaphore N
```

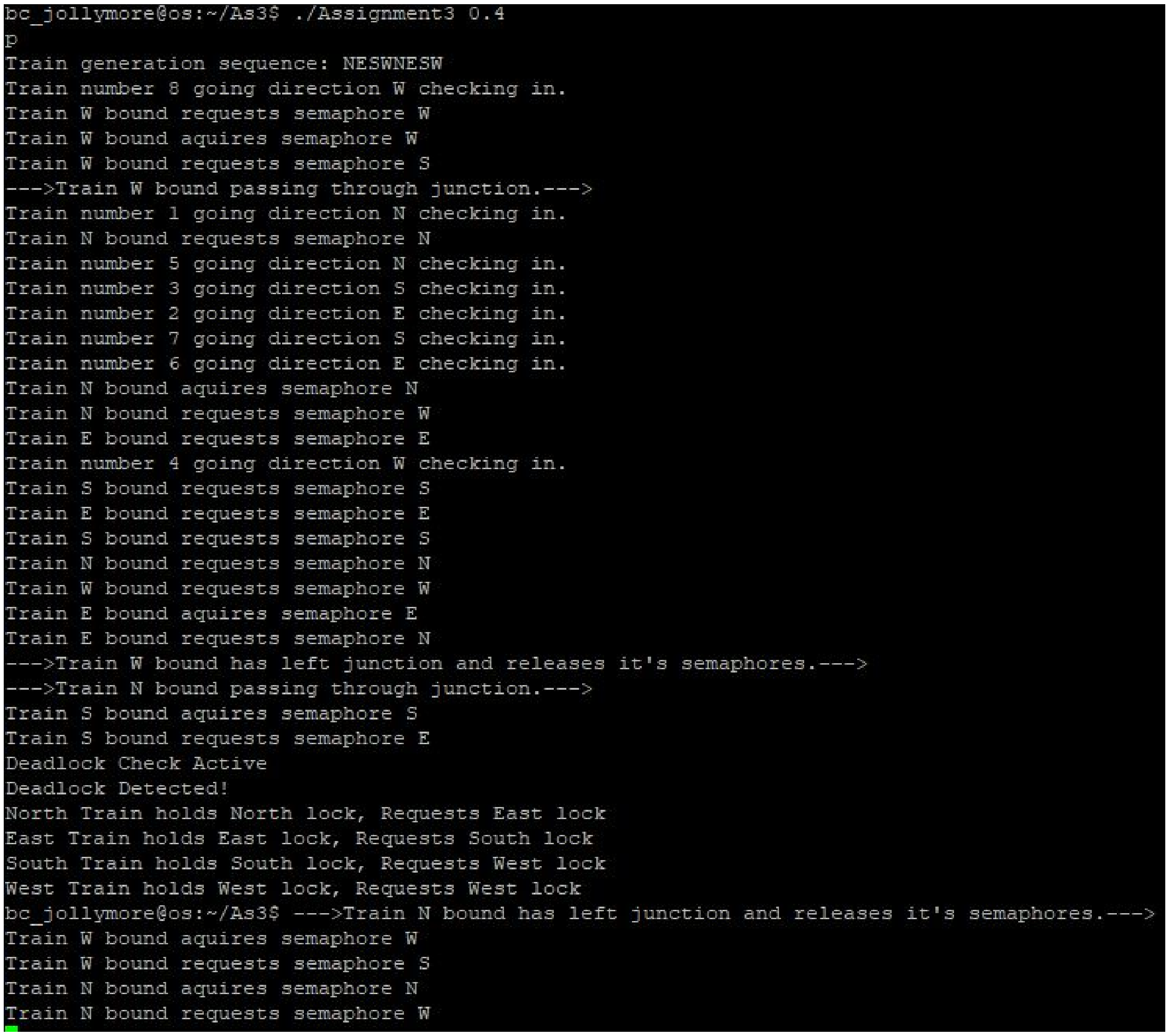
NNSS



```
bc_jollymore@os:~/As3$ ./Assignment3 0.3
p
Train generation sequence: NNSS
Train number 1 going direction N checking in.
Train number 2 going direction N checking in.
Train N bound requests semaphore N
Train number 3 going direction S checking in.
Train S bound requests semaphore S
Train number 4 going direction S checking in.
Train N bound requests semaphore N
Train N bound aquires semaphore N
Train S bound requests semaphore S
Train N bound requests semaphore W
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train S bound passing through junction.--->
--->Train S bound has left junction and releases it's semaphores.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train S bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
--->Train N bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
--->Train N bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
Deadlock Check Active
No deadlock detected.
All trains are finished: Unlinking semaphores
manager terminating
bc_jollymore@os:~/As3$
```

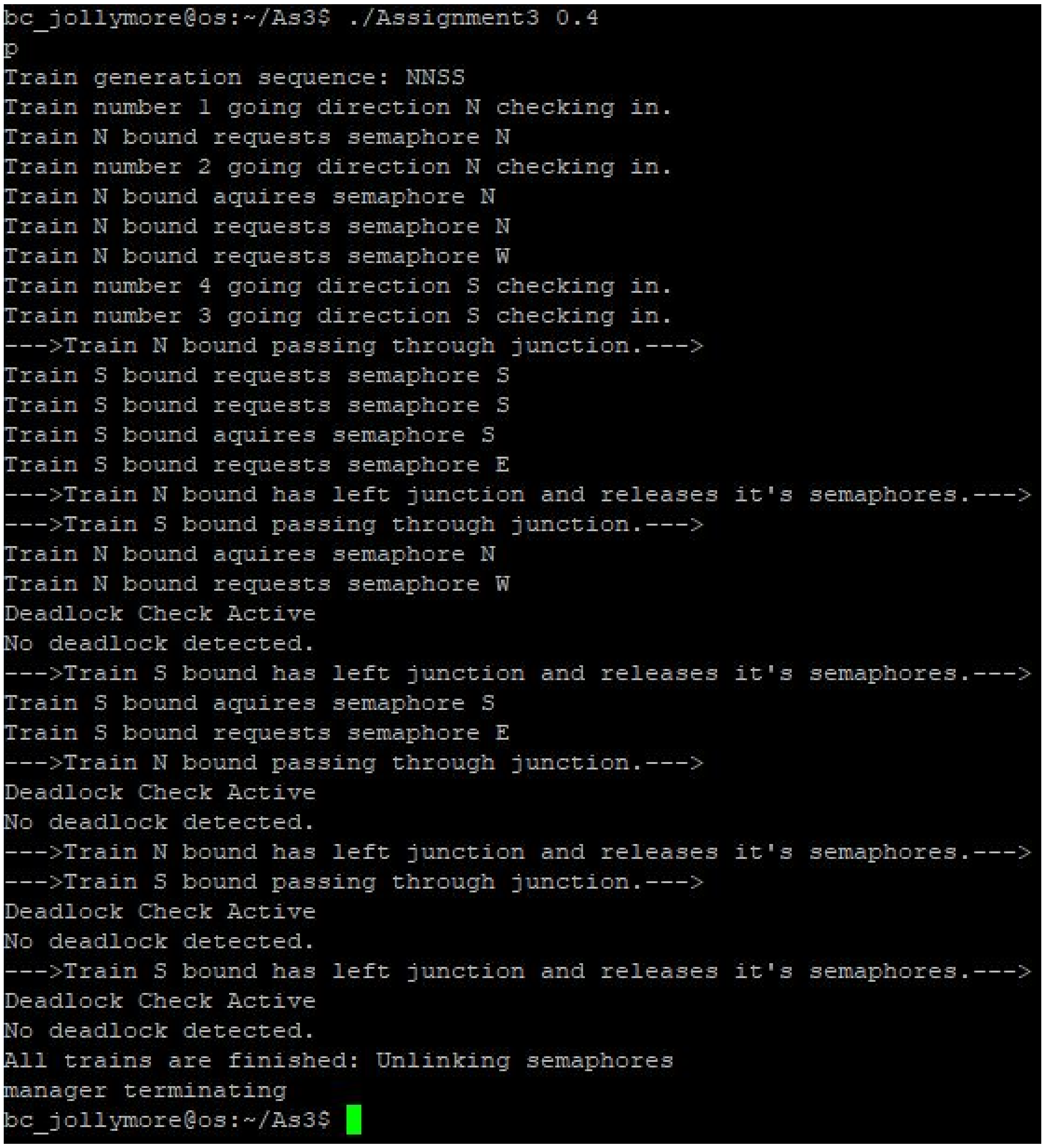
0.4 probability

NESWNESW (deadlock)



```
bc_jollymore@os:~/As3$ ./Assignment3 0.4
p
Train generation sequence: NESWNESW
Train number 8 going direction W checking in.
Train W bound requests semaphore W
Train W bound aquires semaphore W
Train W bound requests semaphore S
--->Train W bound passing through junction.--->
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train number 5 going direction N checking in.
Train number 3 going direction S checking in.
Train number 2 going direction E checking in.
Train number 7 going direction S checking in.
Train number 6 going direction E checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train E bound requests semaphore E
Train number 4 going direction W checking in.
Train S bound requests semaphore S
Train E bound requests semaphore E
Train S bound requests semaphore S
Train N bound requests semaphore N
Train W bound requests semaphore W
Train E bound aquires semaphore E
Train E bound requests semaphore N
--->Train W bound has left junction and releases it's semaphores.--->
--->Train N bound passing through junction.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
Deadlock Check Active
Deadlock Detected!
North Train holds North lock, Requests East lock
East Train holds East lock, Requests South lock
South Train holds South lock, Requests West lock
West Train holds West lock, Requests West lock
bc_jollymore@os:~/As3$ --->Train N bound has left junction and releases it's semaphores.--->
Train W bound aquires semaphore W
Train W bound requests semaphore S
Train N bound aquires semaphore N
Train N bound requests semaphore W
```

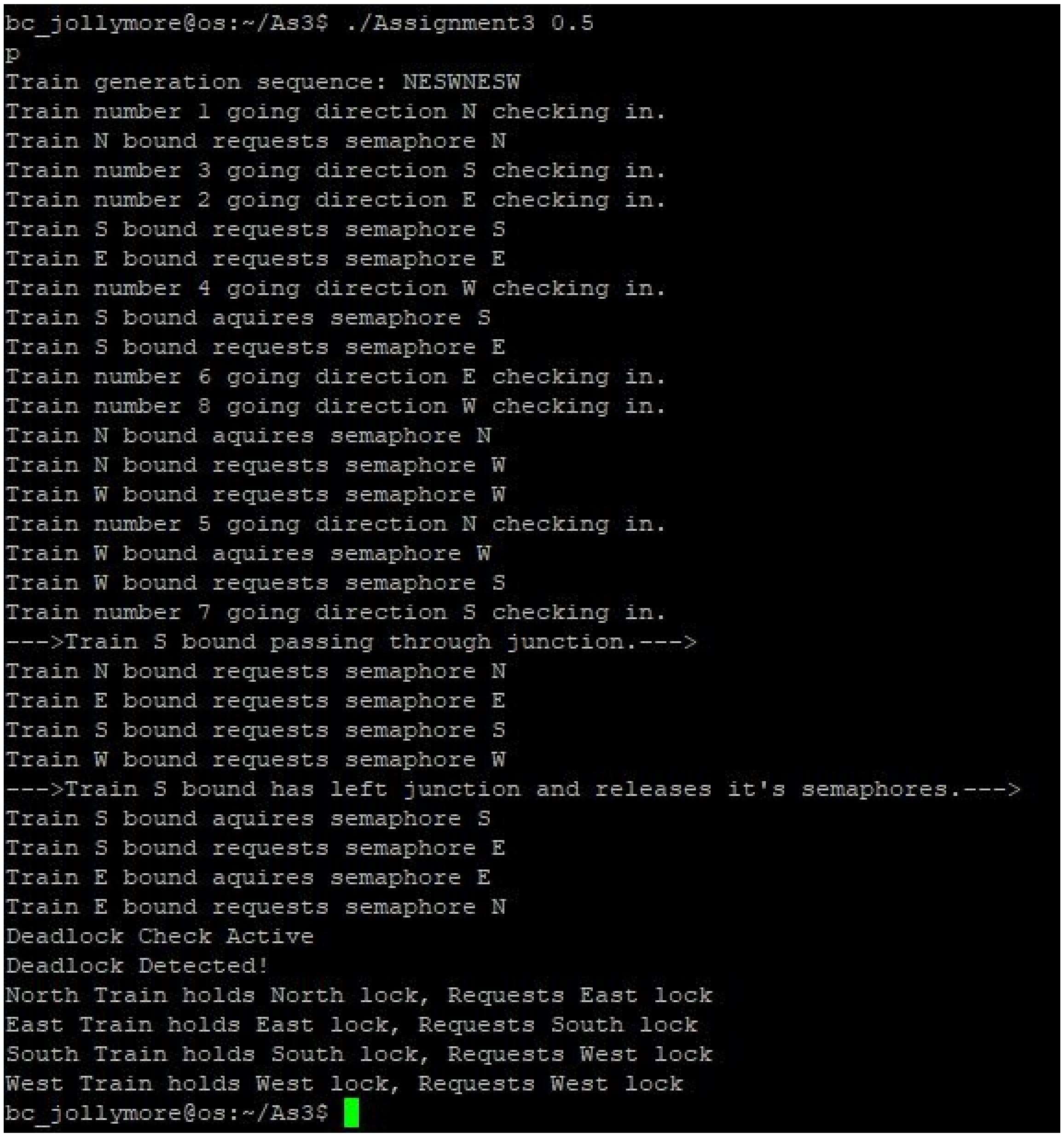
NNSS



```
bc_jollymore@os:~/As3$ ./Assignment3 0.4
p
Train generation sequence: NNSS
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train number 2 going direction N checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore N
Train N bound requests semaphore W
Train number 4 going direction S checking in.
Train number 3 going direction S checking in.
--->Train N bound passing through junction.--->
Train S bound requests semaphore S
Train S bound requests semaphore S
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train N bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
Deadlock Check Active
No deadlock detected.
All trains are finished: Unlinking semaphores
manager terminating
bc_jollymore@os:~/As3$
```

0.5 probability

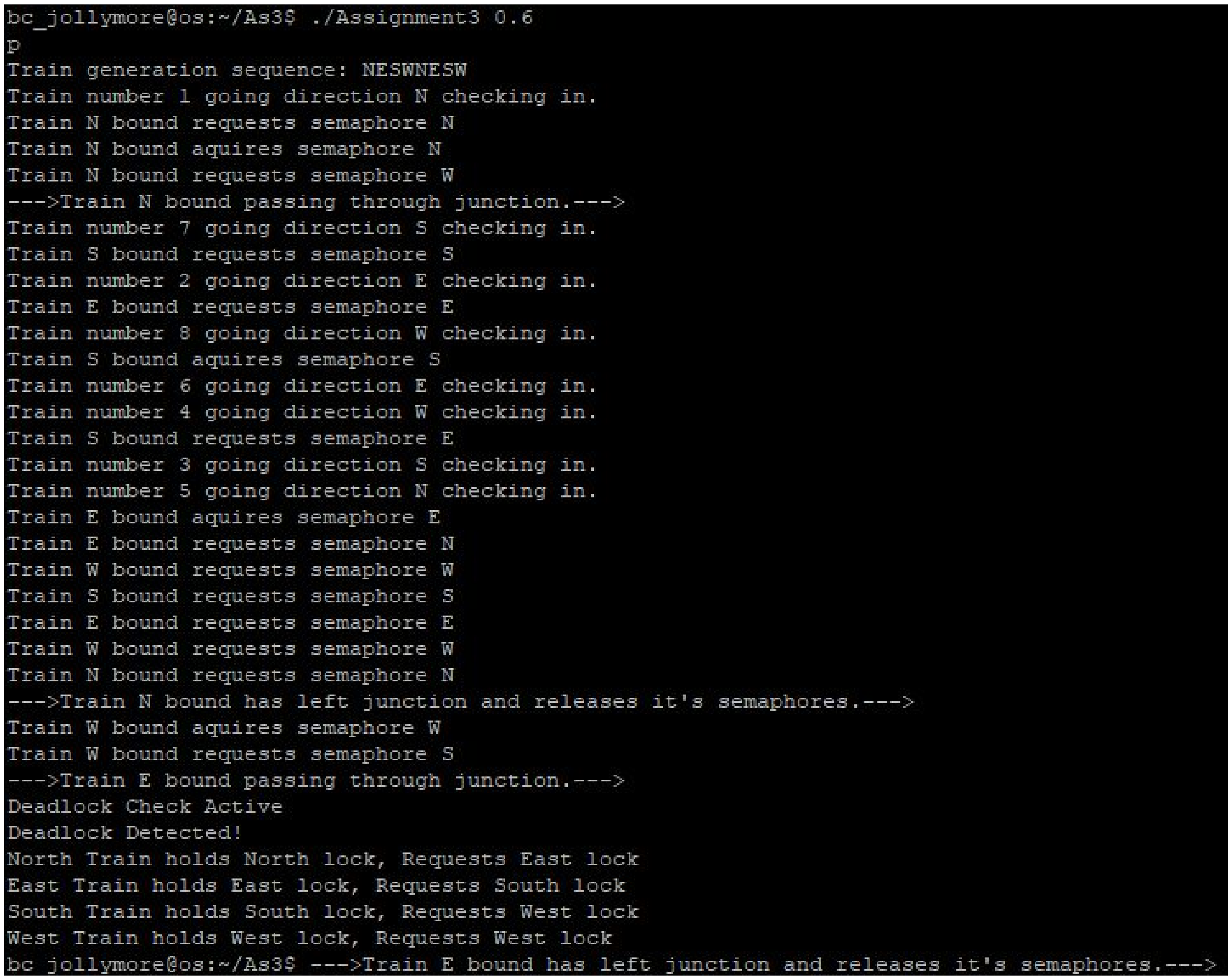
NESWNESW (deadlock)



```
bc_jollymore@os:~/As3$ ./Assignment3 0.5
p
Train generation sequence: NESWNESW
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train number 3 going direction S checking in.
Train number 2 going direction E checking in.
Train S bound requests semaphore S
Train E bound requests semaphore E
Train number 4 going direction W checking in.
Train S bound aquires semaphore S
Train S bound requests semaphore E
Train number 6 going direction E checking in.
Train number 8 going direction W checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train W bound requests semaphore W
Train number 5 going direction N checking in.
Train W bound aquires semaphore W
Train W bound requests semaphore S
Train number 7 going direction S checking in.
--->Train S bound passing through junction.--->
Train N bound requests semaphore N
Train E bound requests semaphore E
Train S bound requests semaphore S
Train W bound requests semaphore W
--->Train S bound has left junction and releases it's semaphores.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
Train E bound aquires semaphore E
Train E bound requests semaphore N
Deadlock Check Active
Deadlock Detected!
North Train holds North lock, Requests East lock
East Train holds East lock, Requests South lock
South Train holds South lock, Requests West lock
West Train holds West lock, Requests West lock
bc_jollymore@os:~/As3$
```

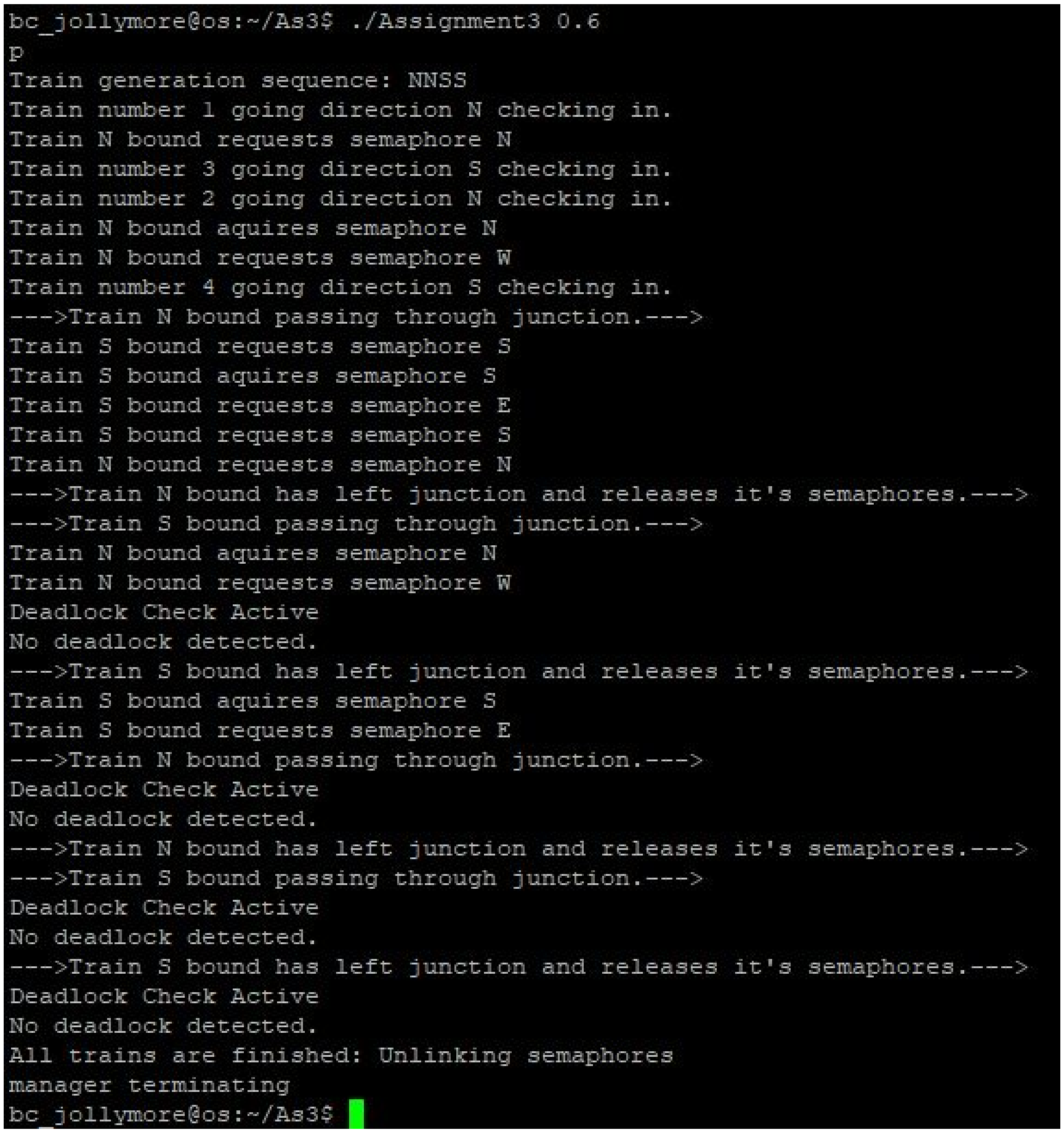
NNSS

0.6 probability
NESWNESW(deadlock)



```
bc_jollymore@os:~/As3$ ./Assignment3 0.6
p
Train generation sequence: NESWNESW
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train N bound aquires semaphore N
Train N bound requests semaphore W
--->Train N bound passing through junction.--->
Train number 7 going direction S checking in.
Train S bound requests semaphore S
Train number 2 going direction E checking in.
Train E bound requests semaphore E
Train number 8 going direction W checking in.
Train S bound aquires semaphore S
Train number 6 going direction E checking in.
Train number 4 going direction W checking in.
Train S bound requests semaphore E
Train number 3 going direction S checking in.
Train number 5 going direction N checking in.
Train E bound aquires semaphore E
Train E bound requests semaphore N
Train W bound requests semaphore W
Train S bound requests semaphore S
Train E bound requests semaphore E
Train W bound requests semaphore W
Train N bound requests semaphore N
--->Train N bound has left junction and releases it's semaphores.--->
Train W bound aquires semaphore W
Train W bound requests semaphore S
--->Train E bound passing through junction.--->
Deadlock Check Active
Deadlock Detected!
North Train holds North lock, Requests East lock
East Train holds East lock, Requests South lock
South Train holds South lock, Requests West lock
West Train holds West lock, Requests West lock
bc_jollymore@os:~/As3$ --->Train E bound has left junction and releases it's semaphores.--->
```

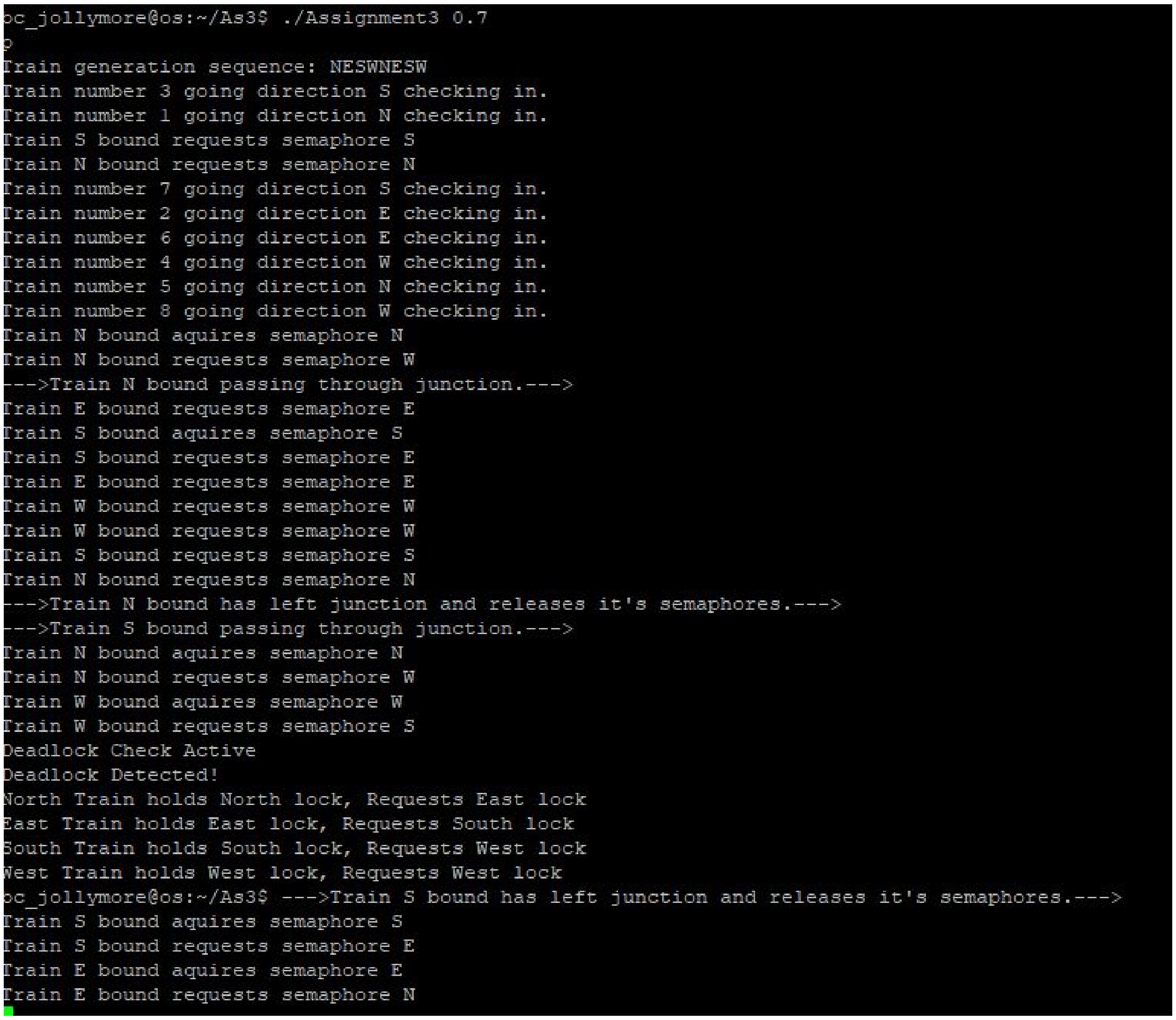
NNSS



```
bc_jollymore@os:~/As3$ ./Assignment3 0.6
p
Train generation sequence: NNSS
Train number 1 going direction N checking in.
Train N bound requests semaphore N
Train number 3 going direction S checking in.
Train number 2 going direction N checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train number 4 going direction S checking in.
--->Train N bound passing through junction.--->
Train S bound requests semaphore S
Train S bound aquires semaphore S
Train S bound requests semaphore E
Train S bound requests semaphore S
Train N bound requests semaphore N
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train N bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
Deadlock Check Active
No deadlock detected.
All trains are finished: Unlinking semaphores
manager terminating
bc_jollymore@os:~/As3$
```

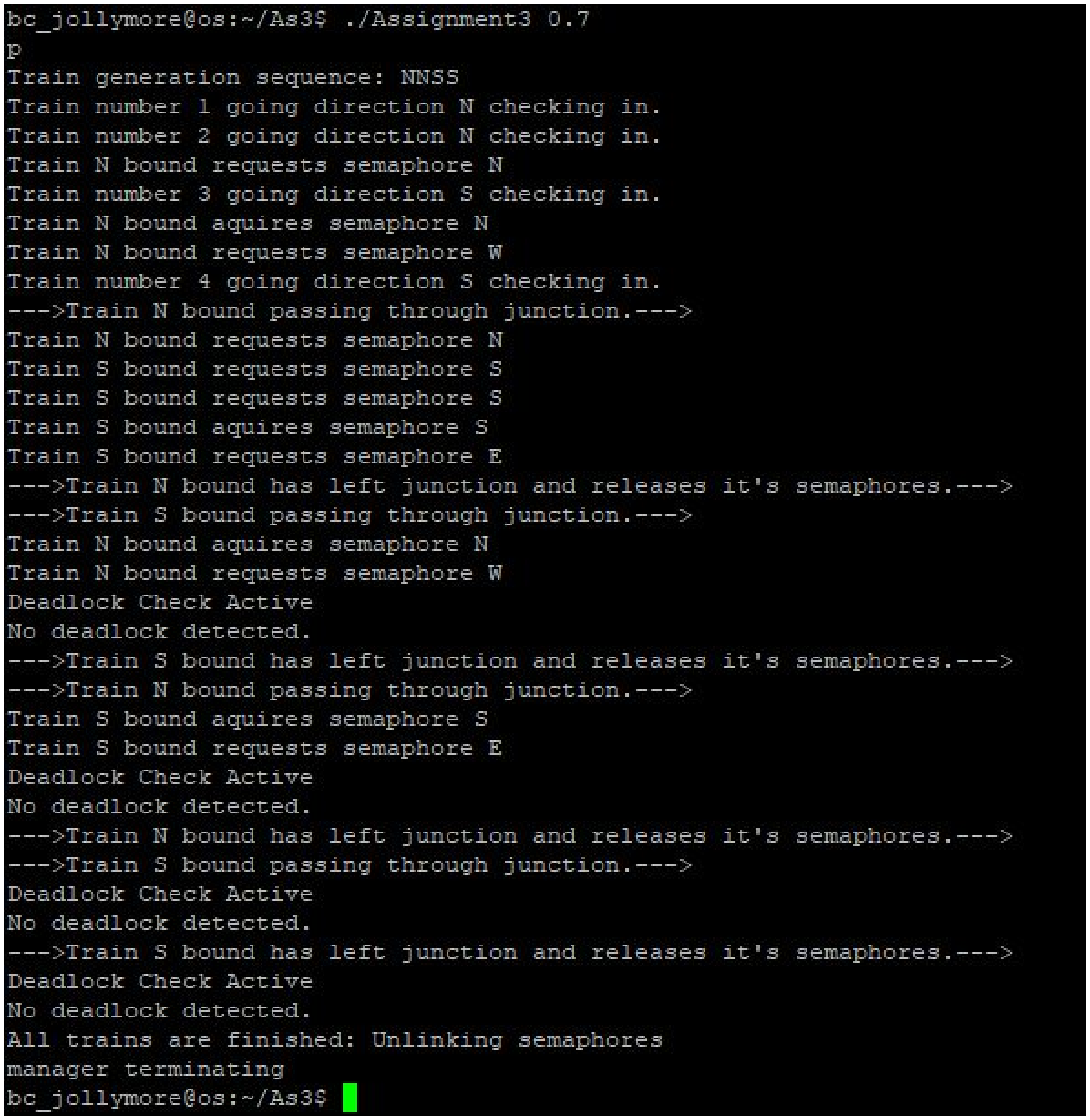
0.7 probability

NESWNESW



```
bc_jollymore@os:~/As3$ ./Assignment3 0.7
p
Train generation sequence: NESWNESW
Train number 3 going direction S checking in.
Train number 1 going direction N checking in.
Train S bound requests semaphore S
Train N bound requests semaphore N
Train number 7 going direction S checking in.
Train number 2 going direction E checking in.
Train number 6 going direction E checking in.
Train number 4 going direction W checking in.
Train number 5 going direction N checking in.
Train number 8 going direction W checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore W
--->Train N bound passing through junction.--->
Train E bound requests semaphore E
Train S bound aquires semaphore S
Train S bound requests semaphore E
Train E bound requests semaphore E
Train W bound requests semaphore W
Train W bound requests semaphore W
Train S bound requests semaphore S
Train N bound requests semaphore N
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train W bound aquires semaphore W
Train W bound requests semaphore S
Deadlock Check Active
Deadlock Detected!
North Train holds North lock, Requests East lock
East Train holds East lock, Requests South lock
South Train holds South lock, Requests West lock
West Train holds West lock, Requests West lock
bc_jollymore@os:~/As3$ --->Train S bound has left junction and releases it's semaphores.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
Train E bound aquires semaphore E
Train E bound requests semaphore N
```

NNSS



```
bc_jollymore@os:~/As3$ ./Assignment3 0.7
p
Train generation sequence: NNSS
Train number 1 going direction N checking in.
Train number 2 going direction N checking in.
Train N bound requests semaphore N
Train number 3 going direction S checking in.
Train N bound aquires semaphore N
Train N bound requests semaphore W
Train number 4 going direction S checking in.
--->Train N bound passing through junction.--->
Train N bound requests semaphore N
Train S bound requests semaphore S
Train S bound requests semaphore S
Train S bound aquires semaphore S
Train S bound requests semaphore E
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Train N bound aquires semaphore N
Train N bound requests semaphore W
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
--->Train N bound passing through junction.--->
Train S bound aquires semaphore S
Train S bound requests semaphore E
Deadlock Check Active
No deadlock detected.
--->Train N bound has left junction and releases it's semaphores.--->
--->Train S bound passing through junction.--->
Deadlock Check Active
No deadlock detected.
--->Train S bound has left junction and releases it's semaphores.--->
Deadlock Check Active
No deadlock detected.
All trains are finished: Unlinking semaphores
manager terminating
bc_jollymore@os:~/As3$
```

Switching from Matrix Shared Memory to Timed Semaphores

The switch to timed semaphores allows the processes sharing the memory to temporarily release their lock on a resource, incrementing a timer by a few nanoseconds to prevent the same process from immediately grabbing the resource again. This allows for a process currently waiting to grab the resource it needs (NT requesting W lock. After certain system time, the lock will release temporarily, preventing both N and W trains from grabbing their directional lock.

Whichever train arrived first will be released from the sleep and grab both semaphores before the other trains wake up. This allows the N train to complete its time in the junction and release its locks.) A deadlock is thus prevented at the cost of performance/speed.

A timeout for the semaphores will occur if the absolute time specified in abstime passes.
If a timeout occurs, the function will perform a sem_post() and release the lock to another process currently waiting. The process which just timed out will then call sem_wait() until it receives the resources needed to complete its function.

A process that would take 30 seconds to complete (assuming 10 trains spending 3 seconds in a junction) would be at best the same speed if a deadlock doesn't occur or (n) nanoseconds slower. The function sem_timedwait() allows for trains to avoid deadlocking by giving up their resource such that another train can complete its task.

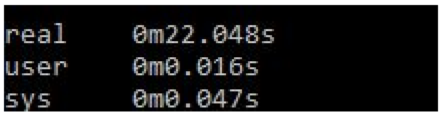
Troubleshooting

The first iterations started by passing semaphores through pipes. This came with a myriad of issues such as execl(…) breaking and not passing through a semaphore or passing through a semaphore and causing the child process to crash. To fix this, we implemented shared memory semaphores. Using shared memory allowed the uses of post, wait and unlink. Post allowed a process to release its lock, wait requested a lock and unlinked closed the semaphore. Unlink was particularly important since if a process closed prematurely without unlinking, the semaphore would still be stuck in memory causing semaphore associated errors. This issue was fixed by unlinking any semaphores during process start and unlinking semaphores at the end of process.

Conclusion

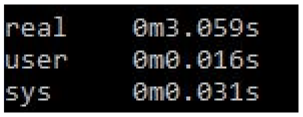
Shared memory is a powerful tool but needs to be managed correctly. As demonstrated by the assignment, a deadlock can occur if a cycle occurs when requesting and acquiring locks.
For example, when 4 trains acquire their locks but require the next train's lock then a cycle is created causing a deadlock. ...N>T1>W>T2>S>T3>E>T4>N… The simplest way to deal with this method is to just shut down all processes when a deadlock occurs or might occur. Another method we tested was using timed semaphores to periodically release locks back into the pool from which the next train in queue will acquire. This method avoided deadlocks but it came at a cost of performance and speed.

Timed Semaphore: NESWNESW



```
real    0m22.048s
user    0m0.016s
sys     0m0.047s
```

Close Processes: NESWNESW



```
real    0m3.059s
user    0m0.016s
sys     0m0.031s
```

As can been seen, it may be more efficient to just crash the processes and start again from a non deadlocked state then it would be to avoid a deadlock. But the other method is just as valid, especially if sensitive data is involved that must be kept intact at the expense of performance.

References

https://www.justsoftwaresolutions.co.uk/threading/locks-mutexes-semaphores.html
https://linux.die.net/man/3/sem_timedwait
http://vega.cs.kent.edu/~mikhail/classes/aos.f99/l14deadlocks.PDF
https://www.geeksforgeeks.org/c-program-count-number-lines-file/
https://stackoverflow.com/questions/1345670/stack-smashing-detected
https://stackoverflow.com/questions/48936647/read-a-matrix-from-a-file
https://cboard.cprogramming.com/c-programming/127472-how-pass-float-type-argument-command-line.html
http://fileadmin.cs.lth.se/cs/Education/EDAF55/resourcegraphs.pdf