**Case Study**

Simulate a traffic signal with four sides, denoted by a 3x3 grid as given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **North** | | |  |
| **West** | A1 | A2 | A3 | **East** |
| B1 | B2 | B3 |
| C1 | C2 | C3 |
|  | **South** | | |  |

Cars can come in from North, South, East or West and can only cross straight to the opposite side. For example, the path for cars coming in from North is A2, B2, C2 and then they cross over to South side. The path for cars coming in from East is B3, B2, B1 and then they cross over to West side.

Following are the restrictions to be imposed:

1. There can be traffic in only one direction on the signal at any given time.
2. Cars can only cross the traffic signal in a straight line. (i.e. North to South, South to North, East to West and West to East.)
3. Only one car can occupy a tile (i.e. A2, B2, C2 etc.) at any given time.
4. Traffic from any direction may not start until the whole grid is free from any previous cars.

**Each** side will have its own queue of waiting cars with a limit of **maximum 40 cars** waiting at the same time. All cars should be processed in the same order as they have been added to the queue.

A creator thread will generate cars randomly for all sides.

Each direction should be simulated with a thread, with its own queue as described above.

Use round robin algorithm to decide which direction gets possession of the traffic signal.

**Make sure to get rid of deadlocks and starvation.**

Create a reporting thread which will sleep for 1 second intervals and then print the number of waiting cars on each direction like:

North: 5

South: 7

East:10

West:3