**Variables used:**

1. **Counting Variables**:
   1. neast: for number of vehicles going east to west
   2. nnorth: for number of vehicles going north to south
   3. intersection: number of vehicles present inside the intersection
2. **Boolean Variables:**
   1. iseast: True if first vehicle is of East else false
   2. isnorth: True if first vehicle is of North else false
3. **Semaphores:**
   1. north\_available: For controlling access such that at a time only one vehicle enters the intersection from north
   2. east\_available: For controlling access such that at a time only one vehicle enters the intersection from east
   3. intersection\_available\_east: To wait (busy waiting) if the intersection is not available for east.
   4. Intersection\_available\_north: To wait (busy waiting) if the intersection is not available for north.
   5. lock: To control access of shared variable “intersection” such that it is not written by two threads at the same instance.

**Code Explanation:**

**Input:** The program takes input from user about how many cars are coming from east and north.

**Working:**

The program treats cars as threads. If a car wants to go to East-West it calls the routine east\_algo() and if a car wants to go to North-South it calls the routine north\_algo()

In the function we first check if the point of intersection is occupied by any car going for the opposite direction. For example: If the intersection is occupied by a car waiting to go to East, and the next car that comes in the intersection wants to go to North, then it will have to wait until all the cars willing to go to East clears the intersection such that no collision occurs.

**Logic:**

If the first cars comes is for a certain direction (say east-west) then it will set its respective Boolean variable true (iseast). The cars going for opposite direction will have to wait until all the cars willing to go to West have cleared the intersection. The order of the cars (threads) is arbitrary and not unique.

**Output:**

