**Lab – 8**

**Parallel cellular algorithm and program**

**import numpy as np**

**def traffic\_flow\_simulation(road\_length, num\_vehicles, V\_max, p\_slow, steps):**

**road = -1 \* np.ones(road\_length, dtype=int)**

**vehicle\_positions = np.random.choice(range(road\_length), num\_vehicles, replace=False)**

**for pos in vehicle\_positions:**

**road[pos] = 0**

**def distance\_to\_next\_vehicle(pos):**

**distance = 1**

**while road[(pos + distance) % road\_length] == -1 and distance < road\_length:**

**distance += 1**

**return distance**

**for step in range(steps):**

**new\_road = -1 \* np.ones(road\_length, dtype=int)**

**velocities = np.copy(road)**

**for i in range(road\_length):**

**if road[i] != -1:**

**v = road[i]**

**v = min(v + 1, V\_max)**

**d = distance\_to\_next\_vehicle(i) - 1**

**v = min(v, d)**

**if np.random.rand() < p\_slow:**

**v = max(v - 1, 0)**

**velocities[i] = v**

**for i in range(road\_length):**

**if road[i] != -1:**

**v = velocities[i]**

**new\_pos = (i + v) % road\_length**

**new\_road[new\_pos] = v**

**road = new\_road**

**road\_state = ''.join(['.' if x == -1 else str(x) for x in road])**

**print(f"Step {step + 1}: {road\_state}")**

**road\_length = 30**

**num\_vehicles = 10**

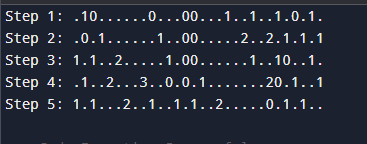
**V\_max = 5**

**p\_slow = 0.3**

**steps = 5**

**traffic\_flow\_simulation(road\_length, num\_vehicles, V\_max, p\_slow, steps)**

**output:**

****