Output :-

Consider there are N philosophers seated around a circular table with one chopstick between each pair of philosophers.

- Each philosopher needs two chopsticks to eat.
- A philosopher may eat only if he can pick up both adjacent chopsticks.
- One chopstick can be picked up by only one of its adjacent philosophers at a time (not both).

Write a program to solve the problem using **process synchronization technique**.

Code:-

```
  GNU nano 7.2                                                    dp.c *
}

int main() {
    pthread_t thread[N];
    int id[N];

    for(int i = 0; i < N; i++)
        sem_init(&chopstick[i], 0, 1);

    for(int i = 0; i < N; i++) {
        id[i] = i;
        pthread_create(&thread[i], NULL, philosopher, &id[i]);
    }

    for(int i = 0; i < N; i++)
        pthread_join(thread[i], NULL);

    return 0;
}
```

Output:-

```
himani@DELL:~$ nano dp.c
himani@DELL:~$ gcc dp.c -o dp -lpthread
himani@DELL:~$ ./dp
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 2 is thinking
Philosopher 2 is eating
Philosopher 2 is thinking
Philosopher 1 is eating
Philosopher 1 is thinking
Philosopher 0 is eating
Philosopher 0 is thinking
Philosopher 4 is eating
Philosopher 4 is thinking
Philosopher 3 is eating
Philosopher 3 is thinking
Philosopher 2 is eating
Philosopher 2 is thinking
Philosopher 1 is eating
Philosopher 1 is thinking
Philosopher 0 is eating
Philosopher 0 is thinking
Philosopher 4 is eating
Philosopher 4 is thinking
Philosopher 3 is eating
Philosopher 3 is thinking
Philosopher 2 is eating
Philosopher 2 is thinking
Philosopher 1 is eating
^C
himani@DELL:~$
```