Example of deadlock

```
Semaphore chopstick[5]={1};
Int i;
Void philosopher (i)
{
while (true)  {
wait(chopstick[i]); // left chopstick
wait(chopstick[(i+1)%5 ]); // right chopstick
// eat
signal(chopstick[i]); // left chopstick
signal(chopstick[(i+1)%5 ]); // right chopstick
// think
}
```

In the above structure, first wait operation is performed on chopstick[i] and chopstick[(i+1) % 5]. This means that the philosopher i has picked up the chopsticks on his sides. Then the eating function is performed.

After that, signal operation is performed on chopstick[i] and chopstick[(i+1) % 5]. This means that the philosopher i has eaten and put down the chopsticks on his sides. Then the philosopher goes back to thinking.

The above solution makes sure that no two neighboring philosophers can eat at the same time. But this solution can lead to a deadlock. This may happen if all the philosophers pick their left chopstick simultaneously. Then none of them can eat and **deadlock** occurs.

Example of deadlock Solution

```
Semaphore chopstick[5]={1};
Int i;
Void philosopher (i)
{
while (true)  {
if (chopstick[i] && chopstick[(i+1)%5]) {
wait(chopstick[i]); // left chopstick
wait(chopstick[(i+1)%5 ]); // right chopstick
// eat
signal(chopstick[i]); // left chopstick
signal(chopstick[(i+1)%5 ]); // right chopstick
// think
}
}
```

[if (chopstick[i] && chopstick[(i+1)%5])**]** is the **solution for deadlock** that a philosopher should only be allowed to pick their chopstick if both are available at the same time.