"The Three Little Pigs" synchronization problem: In this problem, three threads representing the three little pigs need to access a shared resource, a brick house, but only one pig can use the house at a time. The house has a limited capacity of one pig, so if more than one pig tries to access the house simultaneously, a race condition occurs. The solution involves the use of Monitor.Wait and Monitor.Pulse to coordinate the pigs' access to the house and ensure that only one pig can use the house at a time.

"Goldilocks and the Three Bears" synchronization problem: In this problem, three threads representing three bears have access to a shared resource, a bowl of porridge. Goldilocks, another thread, also wants to access the porridge, but she must wait until the bears are finished. Similarly, the bears must wait for Goldilocks to finish before they can access the porridge again. The solution involves the use of Monitor.Wait and Monitor.Pulse to coordinate the threads' access to the porridge and ensure that each thread can access the porridge in turn.

"Cinderella" synchronization problem: In this problem, a thread representing Cinderella wants to access a shared resource, a glass slipper, but the slipper is guarded by a lock. The lock can only be released by the fairy godmother, another thread. Cinderella must wait for the fairy godmother to release the lock before she can access the slipper. The solution involves the use of Monitor.Wait and Monitor.Pulse to coordinate Cinderella's access to the slipper and ensure that she can only access it when the lock is released by the fairy godmother.

"Jack and the Beanstalk" synchronization problem: In this problem, two threads representing Jack and the giant want to access a shared resource, a beanstalk. Jack wants to climb up the beanstalk, while the giant wants to climb down. The problem is that the beanstalk can only support one climber at a time, and if both try to climb at the same time, a race condition occurs. The solution involves the use of Monitor.Wait and Monitor.Pulse to coordinate the climbers' access to the beanstalk and ensure that only one climber can use the beanstalk at a time.

"Little Red Riding Hood" synchronization problem: In this problem, two threads representing Little Red Riding Hood and the big bad wolf want to access a shared resource, a basket of goodies. Little Red Riding Hood wants to add goodies to the basket, while the big bad wolf wants to take goodies out. The problem is that the basket can only hold a limited number of goodies, and if both try to access it simultaneously, a race condition occurs. The solution involves the use of Monitor.Wait and Monitor.Pulse to coordinate the threads' access to the basket and ensure that only one thread can add or remove goodies at a time.