Homework 3
Multiple producers - one consumer

Printed trace of simulated events:

The following trace below **showcases** which bee adds honey to the pot, with each bee corresponding to an individual thread. We have initialized the pot's capacity to **10 portions**, while the number of bees is set to **5**. Although we can increase the number of bees beyond the pot's capacity, it is not guaranteed that each bee will contribute equally, which may lead to potential unfairness. When the pot is full, the bear wakes up, eats the honey, and resets the pot. The next cycle of bees adding honey then begins.

Analyzing the Program's output:
When analyzing the program's output, we should look for the following patterns:

- **Bee actions:** Each bee should take turns adding honey, ensuring that contributions are evenly distributed over multiple cycles.
- **Bear behavior:** The bear should wake up **only** when the pot reaches its full capacity (`H`). It should then **completely empty** the pot before allowing the bees to start refilling it.

What does fairness mean in this context?
Fairness in this problem means that no bee gets significantly more opportunities to wake the bear than the others. Each bee should have an equal chance of adding the last portion of honey and waking the bear over multiple cycles.

Although the solution is mostly fair, fairness may vary depending on thread scheduling. Given the current implementation, all bees have an equal opportunity to add honey. The randomization of bee arrivals is controlled by sleep(rand() % 3 + 1), which simulates realistic timing and ensures that bees reach the pot at different times, creating a natural balance.

Additionally, bees only wait when the pot is full and are never permanently blocked from contributing (sem_wait(empty_pot)). Once the pot is emptied, all bees are released at the same time, allowing them to compete fairly for the next cycle.

Observing multiple rounds of execution, we see that different bees wake the bear due to the randomized sleep times.

When Could There Be Unfairness?

Although we have implemented a randomized sleep time, the bee that receives the shortest sleep duration may return faster than others, increasing its chances of filling the pot more frequently. Another reason for potential unfairness could be caused by the unintentional priority of certain bees, where If some bees are created **before others** in the loop, they might get slightly earlier CPU execution time. Since the threads are created in a loop from `i = 0` to `N-1`, the first bees created might have a slight timing advantage in getting the CPU first. A potential solution for this is to randomize the order of bee thread creation and to introduce a small randomized delay after the bear releases the bees with: "usleep(rand() % 50000);".

Printed trace of simulated events:

```
Bee 5 added honey. Pot now has 1/10 portions.
Bee 1 added honey. Pot now has 2/10 portions.
Bee 4 added honey. Pot now has 3/10 portions.
Bee 2 added honey. Pot now has 4/10 portions.
Bee 4 added honey. Pot now has 5/10 portions.
Bee 3 added honey. Pot now has 6/10 portions.
Bee 5 added honey. Pot now has 7/10 portions.
Bee 2 added honey. Pot now has 8/10 portions.
Bee 1 added honey. Pot now has 9/10 portions.
Bee 5 added honey. Pot now has 10/10 portions.
Bee 5 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...
Bear goes back to sleep. Pot is now empty.

Bee 4 added honey. Pot now has 1/10 portions.
Bee 1 added honey. Pot now has 2/10 portions.
Bee 3 added honey. Pot now has 3/10 portions.
Bee 2 added honey. Pot now has 4/10 portions.
Bee 5 added honey. Pot now has 5/10 portions.
Bee 1 added honey. Pot now has 6/10 portions.
Bee 3 added honey. Pot now has 7/10 portions.
Bee 4 added honey. Pot now has 8/10 portions.
Bee 2 added honey. Pot now has 9/10 portions.
Bee 5 added honey. Pot now has 10/10 portions.
Bee 5 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...
Bear goes back to sleep. Pot is now empty.

Bee 4 added honey. Pot now has 1/10 portions.
Bee 2 added honey. Pot now has 2/10 portions.
Bee 1 added honey. Pot now has 3/10 portions.
Bee 3 added honey. Pot now has 4/10 portions.
Bee 5 added honey. Pot now has 5/10 portions.
Bee 1 added honey. Pot now has 6/10 portions.
Bee 3 added honey. Pot now has 7/10 portions.
Bee 5 added honey. Pot now has 8/10 portions.
Bee 4 added honey. Pot now has 9/10 portions.
Bee 2 added honey. Pot now has 10/10 portions.
Bee 2 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...
Bear goes back to sleep. Pot is now empty.

Bee 3 added honey. Pot now has 1/10 portions.
Bee 5 added honey. Pot now has 2/10 portions.
Bee 4 added honey. Pot now has 3/10 portions.
Bee 1 added honey. Pot now has 4/10 portions.
Bee 2 added honey. Pot now has 5/10 portions.
Bee 3 added honey. Pot now has 6/10 portions.
Bee 5 added honey. Pot now has 7/10 portions.
Bee 3 added honey. Pot now has 8/10 portions.
Bee 4 added honey. Pot now has 9/10 portions.
```

Bee 1 added honey. Pot now has 10/10 portions.
Bee 1 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...
Bear goes back to sleep. Pot is now empty.

Bee 2 added honey. Pot now has 1/10 portions.
Bee 5 added honey. Pot now has 2/10 portions.
Bee 3 added honey. Pot now has 3/10 portions.
Bee 4 added honey. Pot now has 4/10 portions.
Bee 1 added honey. Pot now has 5/10 portions.
Bee 1 added honey. Pot now has 6/10 portions.
Bee 2 added honey. Pot now has 7/10 portions.
Bee 5 added honey. Pot now has 8/10 portions.
Bee 3 added honey. Pot now has 9/10 portions.
Bee 4 added honey. Pot now has 10/10 portions.
Bee 4 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...
Bear goes back to sleep. Pot is now empty.

Bee 2 added honey. Pot now has 1/10 portions.
Bee 1 added honey. Pot now has 2/10 portions.
Bee 5 added honey. Pot now has 3/10 portions.
Bee 3 added honey. Pot now has 4/10 portions.
Bee 4 added honey. Pot now has 5/10 portions.
Bee 2 added honey. Pot now has 6/10 portions.
Bee 2 added honey. Pot now has 7/10 portions.
Bee 3 added honey. Pot now has 8/10 portions.
Bee 4 added honey. Pot now has 9/10 portions.
Bee 1 added honey. Pot now has 10/10 portions.
Bee 1 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...
Bear goes back to sleep. Pot is now empty.

Bee 5 added honey. Pot now has 1/10 portions.
Bee 3 added honey. Pot now has 2/10 portions.
Bee 2 added honey. Pot now has 3/10 portions.
Bee 4 added honey. Pot now has 4/10 portions.
Bee 1 added honey. Pot now has 5/10 portions.
Bee 2 added honey. Pot now has 6/10 portions.
Bee 3 added honey. Pot now has 7/10 portions.
Bee 4 added honey. Pot now has 8/10 portions.
Bee 2 added honey. Pot now has 9/10 portions.
Bee 5 added honey. Pot now has 10/10 portions.
Bee 5 filled the pot! Waking up the bear...
Bear wakes up! Eating all the honey...