# HomeWork 3

## Task 1: *The hungry bird Problem*

In this assignment a parent bird fills the nest with worms that baby birds in the nest can eat. This was done by having two global semaphores *serve* (full), and *eat* (empty), that was held by either a baby bird or the parent. If the nest was filled with worms, the babies would take turns decrementing the serve semaphore, allowing them to have sole access to the worms in the nest. They would then eat a worm and increment the serve semaphore, allowing another baby to eat a worm.

If a baby discovers that no worms are in the nest, it would alert the parent by incrementing the eat semaphore. The parent would then decrement the eat semaphore claiming access over it, fill up the worms in the nest then increment serve, allowing the baby birds to once again eat.

This solution has a minimum of 2 threads, at least one baby and one parent. The solution is not fair since it prioritizes the babies first, meaning that they have to eat until the nest is empty before the parent is allowed to gather more worms. The solution does also not guarantee that all the babies are allowed to eat an equal amount of food.

## Task 2: *The Bear and the Honeybee problem*

In this task the goal was to feed a hungry bear some honey that a group of bees collected. So we had multiple producers and one customer. How this was done was to first initialize two semaphores that signalized when the pot was full or empty. The bees took the "empty" semaphore and decreased it when starting to collect honey and released it when the honey was at capacity H. When the honey is at H capacity the bear awakens and takes the "full" semaphore, decreases it and starts to eat all the honey and releases it when all the honey is eaten. With this use of semaphores the threads that are represented as many working bees and one hungry bear, the honey is never collected when it is being eaten, and vice versa. The simulation of the program is in the form of entertaining text when the different events of eating honey, collecting honey and when the honeypot is full or empty.

The solution is kind of fair, because of the one customer part. I would say that if every bee didn't get to collect an equal amount of honey that is not unfair, but just an uneven workload. If the write part would be more vital the solution would be more unfair but here it is just honey, or more correctly just an integer increment, and that is not an unfair workload or solution.