# ICE CREAM PARTY NIGHT

Vishal Vijay Devadiga (CS21BTECH11061)

## Code Design

I have used:

- 1 Semaphore for number of ice cream available
- 1 Semaphore for queue implementation
- 2 Semaphore for adding up total times of refilling and eating respectively.

All families enter a queue for starvation-free eating. If pot is empty, waits until the vendor refills it, else it sleeps(eats) then sleeps(community) then re-enters the queue.

Vendor keeps checking whether the pot is empty, if empty then adds the number of trays to the semaphore, and then sleeps again.

The following is the code flow:

- Opens input file in main and get value of k(number of family threads), m(Number of trays of ice cream), n(number of iterations of critical section), and average values of sleep function $\alpha$, $\beta$, $\lambda$, $\mu$
- Create k threads that execute the family function
- Create 1 thread that execute the vendor function.
- Family waits its chance in the queue. It then sleeps(eats ice cream if available), else waits. It then sleeps(community service) and repeats until it finishes its quota of critical section.
- Vendor sleeps until pot is empty, then refills it and sleeps again
- Car threads exit by a global variable when all family threads exit.