# Problem 02: Automating Biryani Serving

The line for serving Biryani is growing day by day in the Kadamb Mess. Annoyed by waiting for hours in the queue you have decided to automate the entire pipeline of serving Biryani.

## Infrastructure

### Robot Chefs

● From now on, there will be M Robot Chefs present in the kitchen preparing vessels of Biryani.
● Each Robot Chef can prepare a random number (greater than or equal to 1) of Biryani vessels at a time.
● Each Biryani vessel has a capacity to serve P Students.

### Serving Tables

● There will be N Serving Tables present in the mess.
● Each Serving Table has a Serving Container which loads a Biryani vessel prepared by any of our Robot Chefs. Only one vessel of Biryani can be loaded at a time into the Serving Container. The Serving Container can be refilled only when it is empty.
● The Serving Table incorporates the latest state of the art technologies and has automated the process of serving Biryani through multiple slots available on it.
● As long as a serving table has enough portions of Biryani left in the serving container, it makes a random number of slots available for the students to collect a portion of Biryani from the Serving Table. Note that the number of slots should always be less than or equal to the portions left in the Serving Container. Read down for exact limits.
● Thus there is a possibility that once a Serving Table has x available slots and later has greater than x or less than x slots available. After serving x portions (and consequently x students) the portions available in the Serving Container are updated.

### Student

● K Students have registered for the Biryani.

## Pipeline

### Robot Chef

● Each Robot chef takes w seconds (random between 2-5) to prepare r vessels (random between 1-10) of biryani at any particular time. Each vessel has a capacity to feed p students (random between 25-50).
● Once the Robot Chef is done preparing the Biryani Vessels, he invokes the function biryani_ready() and does not return from it until all the biryani vessels he has cooked are loaded into any of the serving containers.
● Once all the biryani vessels are empty, the biryani_ready() function returns and the Robot Chef resumes making another batch of Biryani.

**Serving Tables**

● Each Serving table's serving container is initially empty. It waits for any of the Robot Chefs to load a vessel of Biryani into the serving container. The serving table cannot go into serving mode as long as its container is empty.
● Once the serving container is full, it enters into the serving mode. It invokes a function ready_to_serve_table(int number_of_slots) in which number_of_slots (chosen randomly between 1-10) denotes the number of slots available at the particular serving table at that instant. The function must not return until either all the serving slots of the serving table are full or all the waiting students have been assigned a slot. Note that students can be assigned a slot at any of the serving tables. (If there is no waiting student, then the function returns)

**Students**

● Once a student arrives in the mess, he/she invokes a function wait_for_slot(). This function does not return until a Serving table with a free slot is available, which is ready_to_serve_table method is in progress. Once the function returns the students go to the allocated slot and waits for the slot to serve Biryani.
● Once the student is in the slot, he/she will call the function student_in_slot() to let the Serving Table know that he/she has arrived at the slot.
● Once all the students have been served you are done for the day. To convince the Mess Committee you have decided to build a simulation of your efficient serving pipeline.

**Instructions**

● Each Robot Chef, Serving Table, and Student are threads.
● Stop the simulation when all the students are served.
● A Serving Table is used multiple times for serving. That means if there are still students left to be served, the serving table should invoke ready_to_serve_table().
● Use appropriate small delays for the simulation.
● Prepare a detailed report explaining your implementation and assumptions. Note that the report will be graded.
● The use of semaphores is not allowed. You can use only one mutex lock per Serving Table and Robot Chef.
● Your simulation should allow for multiple students to arrive at a Serving Table simultaneously. It must be possible for several students to have called wait_for_slot() function and the wait_for_slot function returned for each of the students before any of the student calling student_in_slot function
● Your simulation must not result in dead-locks.
● You are allowed to declare more functions if you require it. The goal of this problem is for you to appreciate the use of threads and synchronization techniques. There are no strict restrictions on the output format, as long as the simulation functions smoothly and displays appropriate messages.