Assignment 4 Report

Shaunak Badani
20171004

1. Robot chef problem :

Following are the global variables declared :

Data type	Variable name	Purpose
Long long int [array of 1000 elements]	robot_biryani_vessels	The ith element of the array stores the number of biryani vessels prepared by the ith robot.
Long long int	no_of_students	No of students who will come in an orderly fashion to satisfy their hunger pangs with biryani.
Long long int	no_of_tables	No of serving tables available.
Long long int	no_of_chefs	No of robot chefs available to make biryani.
Long long int	time_consume_biryani	Amount of time every student takes to eat biryani.
Long long int	wait_time_student	The time gap between 2 students entering in line.
Tab_info [struct]	tables	Tab_info is a struct containing two integers : p -> feeding capacity of the serving container on the serving table. no_of_slots -> No of slots available for that particular table. Hence these two integers are recorded for every table in the mess.

The program starts with initializing the pipeline.

The *initializePipeline()* method creates threads for the simulation. *No_of_chefs* threads for the chefs, *No_of_tables* threads for the tables and *No_of_students* threads for the students.

Assumptions :

- The student will not be allotted a table where there is no biryani, as opposed to the clarification mentioned later on Moodle :
- "Student X has gone to table Y"
- "Student X has been served biryani"
- If we had to print these two statements in my simulation, they would be printed one after the other, as the student is not allotted to the table where there is no biryani.

Robot Chef Implementation :

- Every robot chef has its own thread. The thread starts with the execution of the *make_biryani* function. This function does the following :
    - Waits for previous biryani vessels prepared to be emptied into serving tables.
    - Gives some rest time before preparing the next batch of biryani.
    - Makes biryani within 2 - 5 seconds.
    - Prepares 1 - 10 biryani vessels in the time interval mentioned.
    - After that is done, it invokes the *biryani_ready* function.
    - This function runs in the particular thread until the no of students becomes 0, i.e until all students are served.
- The *biryani_ready* function updates the no of vessels prepared in the *robot_biryani_vessels* array, and returns.

Serving Tables Implementation :

- Every serving table has its own thread. The thread starts with the execution of *wait_for_biryani* function. This function does the following :
    - Looks for biryani vessels in the *robot_biryani_vessels* array.
    - If a biryani vessel is found, it allocates a certain feeding capacity (i.e. no of students it can serve) between 25 - 50.
    - Updates the *tables* array with the feeding capacity of that particular table.
    - Calls *ready_to_serve_table* function.
- The *ready_to_serve_table* function selects 1 - feeding_capacity no of slots, and assigns it to the particular table.

Student Implementation :

- Every student has its own thread. It invokes *wait_for_slot* function which does the following :
    - Looks for empty slots with tables which have non empty biryani vessels.
    - If such a table is found, the *student_in_slot* function is called.
- The *student_in_slot* function does the following :
    - Allots portion of biryani to student.
    - Waits for *time_consume_biryani* seconds during which the student eats biryani.
    - After that time, the student leaves that slot to make space for other hungry students standing in the circular Kadamb mess line.