**Missions**

**Common matters to both missions**

* Order list is given differently for each mission Pizza Shop.
* There are no invalid orders (arriving at negative time, taking-outs, etc.).
* No delays (Orders registered immediately, No delay in delivering orders to tables)
* 2000 ms – the duration of one timestamp A
* 200 ms – the duration of one timestamp B
* 4 - timestamps spent by one person to eat one small pizza
* 6 - timestamps spent by one person to eat one large pizza
* 10 – timestamps spent to reassign and move customers to other table
* Set of tables:

{ One table for 8; One table for 20; Four tables, each for 4; Five tables, each for 1 }.

You can choose these tables to use in a pizza shop. Avoid keeping redundant tables.

**Precautions**

* Before going into implementation, you should read all the issues and constraints carefully and proceed with the design.
* Designed and implemented with scalability in mind to control different number of tables.

**Mission 0. A small Pizza Shop (problem\_id = 0)**

John opened a small pizza shop and bought his tables. Now he wants to simulate the pizza shop process using the management system before actually going into the work.

Condition:

* Number of Requests: 10
* Number of ovens: 1
* 8 – timestamps spent by an oven to cook one large pizza
* 5 – timestamps spent by an oven to cook one small pizza

**Mission 1. A large Pizza Shop (problem\_id = 1)**

John is happy that he was successful with his small pizza shop. Now he wants to expand his business to a lager pizza shop.

But he is concerned how he will manage more number of ordering slots, ovens and tables during request peak times. Without efficient technique, his customers will stand in queues for far too long.

Condition:

* Number of Requests: 100
* Number of ovens: 4
* 6 – timestamps spent by an oven to cook one large pizza
* 4 – timestamps spent by an oven to cook one small pizza

Let's make customers happy by developing an efficient queuing and cooking arrangements based on the above!