Due date  Week 14

Please note: this is a draft assignment specification. Please notify any problems in the specification.

1 The project

The project is to construct a specification, refinement and implementation of a function management system for a network of restaurants.

1.1 Brief Overview

The function management system maintains information on restaurants, bookings, menus, ordering and payment for a network of restaurants. The system will need to keep maintain the following:

- Information on restaurants;
- Reservations per restaurant;
- Table Management information: tables, walkins, and arriving parties, track tables status) for each restaurant;
- Menus: optional menu information per reservation;
- Menusitems: for individual ordering by customers at restaurant;
- Payments.

2 Requirements

2.1 Concepts to be Modelled

The system must model the following:

**Restaurants** a restaurant is represented by a name and a unique identifier.
Capacity each restaurant has some number of tables; each table seats some number of guests.

Reservations made by customers for a number of guests at a particular restaurant.

Menus each restaurant has a number of menus that may be associated with availability and may be requested for a reservation.

Walkins casual guests may walk in off the street without a reservation if there are tables available.

Payment guest must pay before exiting the restaurant.

2.2 Operations

status, restaurantid ← AddRestaurant(restaurant): add restaurant name to the network. Restaurant names must be unique.

status ← RemoveRestaurant(restaurantid): remove restaurant name from the network.

status ← AddTables(restaurantid, capacity, number) add a number of tables, each with same capacity to restaurant.

status ← RemoveTables(restaurantid, capacity, number) remove number of tables, each with the same capacity from restaurant.

status, menu ← NewMenu(restaurantid, menuname, price) add a new menu (menuname) with cost price per head to the collection for restaurant. Receive a unique identifier menu.

status ← UpdateMenu(restaurantid, menu, price) update price for menu at restaurant.

status, menuitemid ← NewMenuItem(restaurant, menuitem, price) add a new menuitem with cost price per head to the collection for restaurant. Receive a unique identifier menuitemid.

status ← UpdateMenuItem(restaurantid, menuitemid, price) update price for menuitemid at restaurant.

status ← AddBaseMenu(restaurantid, menu) add a menu to restaurant for all days.

status ← AddMenu(restaurantid, menu, date) add a particular menu on a particular date for restaurant.

status ← DeleteMenu(restaurantid, menu, date) delete a particular menu on a particular date for restaurant.

status ← CloseRestaurant(restaurantid, date) close restaurant on date.

status, reserveid ← MakeReservation(restaurant, number, date): make a reservation for restaurant for number of guests on date.

status ← SelectMenu(restaurant, reserveid, number): select a menu for reserveid for count people.

status ← CancelReservation(restaurant, reserveid): cancel a reservation under reserveid for restaurantid.

status, tableid ← Checkin(restaurant, reserveid) checkin at restaurant for reservation with reserveid.
status, tableid ← Walkin(restaurant, number, date) casual walk in at restaurant for number of people on date.

status ← Order(restaurant, tableid, menuitemid, count) place order for count of menuitemid at tableid in restaurant.

status ← DeliverFood(restaurant, tableid, menuitemid, count) deliver count of menuitemid to tableid in restaurant.

status, cost ← GetBill(restaurant, tableid, date) get bill, for tableid at restaurant on date.

status ← PayBill(restaurant, tableid, date) pay bill, for tableid at restaurant on date.

2.3 Modelling Sets, Constants and Definitions
Context machines should be used for sets, constants and definitions.
You will need to model date, and time.

3 What you should do
1. Develop “low-level” machines with preconditioned, fragile operations.
2. Animate the specifications and develop animation scripts that illustrate the checking of the various conditions in the requirements.
3. Discharge the proof obligations.
4. Develop robust specifications.
5. Discharge the proof obligations.
6. Develop refinements.
7. Discharge proof obligations.
8. Develop implementations.
9. Discharge as many proof obligations as possible.

4 The Challenge
The challenge of this assignment is to resist any temptation to rush to implementation, and to work your way through the refinements, implementations and associated proof obligations. This way you will experience the novelty of generating code that must be correct, modulo the correctness of the specification.
Good luck!