The project is to construct a specification, refinement and implementation of a warehouse management system for managing the stocking and supplying of products stored in (possibly multiple) warehouses.

1.1 Brief Overview

The warehouse management system maintains information on products stored in warehouses. The system will need to keep maintain the following:

- Description of products.
- Quantity of each product in stock.
- Prices of products.
- Location of product in the warehouse(s).
- Expiry date of products.
- Suppliers for products.
- Orders to suppliers and customers.

2 Requirements

2.1 Concepts to be Modelled

The system must model the following:

Products supplied by the company.
Location places where stock is stored. Physically these can be bins or shelves within warehouses. Abstractly there is no need to distinguish: there is simply a set of locations.

Quantity, expiry date and location of stock. It is important that stock of a product with the same expiry date should be kept in the same location.

Capacity of warehouse locations for each product.

Price of products. There could be varying rates based on quantity, expiry date or customer.

Suppliers of products.

Orders for stock from suppliers. Orders will be for a quantity of product with a minimum expiry date.

Customers that purchase products.

Orders for products from customers. Orders will be for a quantity of product with a minimum expiry date.

Invoices for supply of orders.

2.2 Operations

1. status, product \(\leftarrow\) NewProduct(description): add a new product with description.

2. status \(\leftarrow\) SetPrice(product, price): set or update the price of a product. Question: what do we do about expiry date?

3. status, location \(\leftarrow\) NewLocation: allocate a new location in the warehouse(s).

4. status \(\leftarrow\) SetLocationCapacity(location, product, capacity): set the capacity of location for product.

5. status \(\leftarrow\) StoreProduct(product, quantity, expiry): store new stock of a product with expiry date.

6. status, reservation \(\leftarrow\) ReserveProduct(product, quantity, expiry, customer): reserve a quantity of product with minimum expiry for customer and return the reservation identifier. The oldest product should be reserved and there may not be enough in stock.

7. status \(\leftarrow\) UnReserve(reservation): unreserve product with reservation code.

8. status \(\leftarrow\) Dispatch(reservation): dispatch product reserved with reservation identifier.

9. status, supplier \(\leftarrow\) NewSupplier(name): establish a new supplier whose name is name.

10. status \(\leftarrow\) NewSupplierProduct(supplier, product): establish supplier as a source of product.

11. status \(\leftarrow\) SetReorderLevel(product, quantity): set reorder level for product to level. The reorder level is the stock quantity at which a product needs to be re-ordered.

12. status \(\leftarrow\) OrderProduct(supplier, product, quantity, expiry): order quantity of product with minimum expiry date from supplier.

13. status, quantity \(\leftarrow\) AvailableStorage(product, expiry): return the quantity of product with expiry date that can be stored in free space in the warehouse.
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!