Vending Machine Specification

This specification is being developed for the St. Louis Lambda Lounge language shootout (May 2009). The goal is to create a specification specific enough for people to write semantically similar programs and ambiguous enough to allow people to solve the problem in a way idiomatic to the implementation language.

The Problem

The goal of this program is to model a vending machine and the state it must maintain during it's operation. How exactly the actions on the machine are driven is left intentionally vague and is up to the implementor.

The machine works like all vending machines: it takes money then gives you stuff. The vending machine accepts money in the form of nickels, dimes, quarters, and paper dollars. Just for fun, let's leave the set of items being vended open but you must at least have 3 primary items that cost $.65, $1.00, and $1.50. The user may hit a "coin return" button to get back the money you've entered so far. If you put more money in than the item's price, you get change back.

The Specification

The valid set of actions on the vending machine are:

- NICKEL, DIME, QUARTER, DOLLAR - insert money
- COIN RETURN - returns all inserted money
- SERVICE - a service person opens the machine and sets the available change and items
- GET-A, GET-B, GET-C - select item A ($.65), B ($1), or C ($1.50)

The valid set of responses from the vending machine are:

- NICKEL, DIME, QUARTER - return coin
- A, B, C - vend item A, B, or C

The vending machine must track the following state:

- available items - each item has a count, a price, and a selector (A,B,or C)
- available change - # of nickels, dimes, quarters, and dollars available
- currently inserted money

I think that most of the behavior is straightforward but there are some fun possible error conditions that you might want to try to deal with, such as what happens when the user selects an item and needs change that is not available. In general, I think it's best to return change with the fewest coins possible as well.

Example Traces

Here's are some possible event traces:

Example 1: Buy B with exact change
Q, Q, Q, Q, GET-B
-> B

Example 2: Start adding change but hit coin return to get change back
Q, Q, COIN-RETURN
-> Q, Q

Example 3: Buy A without exact change (return $.35)
DOLLAR, GET-A
-> A, Q, D

It would be cool to see these implemented as tests in your implementation!