CS 1102

A term. 2013.

Assignment #3

Due: Tuesday September 24, 2013 AD @ 9:00 a.m. via turnin

Assignment guidelines

1. You should use the intermediate student language. You may use any constructs in ISL, unless doing so would contradict the assignment details.
2. You must provide data definitions and templates. As data get more complex, being careful with getting started correctly becomes more important.
3. You should spend time thinking about good test cases for each part. Explain why you chose the test cases you did (a quick note in a comment in the line before the test case will usually suffice). Show us that you've thought through boundary conditions and ways your program could break.
4. For complicated test cases, provide an explanation for why you believe your output is correct.
5. Vocabulary: a "widget" is a word that refers to any device or item you would manufacture.

A warehouse of widgets

For this assignment you will create a system that manages a warehouse of widgets. Each widget has a name, the number in stock, the price of the widget, and the time to manufacture a new widget. Additional widgets can be constructed, if the materials are available. Thus, each widget also contains the necessary ingredients necessary to manufacture a new one. The ingredients take the form of a list of components needed to manufacture a new item, where each component is a widget as well (so it is a recursive data definition).

Create a data definition and template for a Widget.

Basic properties about widgets

Write the following functions:

1. find-widget-longest-name. widget → widget
   This function will examine the widget, as well as all of the subwidgets used to manufacture it, and return the one whose name is the longest
2. find-widget-most-quantity: widget→ widget
   This function will examine the widget, as well as all of the subwidgets used to manufacture it, and return the one with the most in stock
3. After you have written these two functions, create a broader function
   find: widget test-function → widget
   find takes a widget and a test-function, and returns the widget with the highest score on the test function.

4. Rewrite find-widget-longest-name and find-widget-most-quantity in terms of find. Rename your original functions to find-widget-longest-name-original and find-widget –most-quantity-old (to assist with grading).

Placing simple orders for widgets

Now that we have a collection of widgets for sale, we should provide some infrastructure to enable customers can place an order

1. Customers must specify the widget's name, how many are desired, and how long they are willing to wait for the order to complete
2. Create a data definition and template for type Order
3. Customers expect a reply that indicates how many of the specified widget can be provided, the total cost, and how long the order will take to complete
4. Create a data definition and template for the type Reply
5. Write a function: fulfill-order-immediately: order **widget** → reply
   This function checks an order, and produces a reply with how many of the desired widget are currently in stock and what the price will be

Making more widgets

Sometimes customers will want to purchase more items than are currently in stock. In this situation, we are able to construct additional widgets—if the necessary supplies are on hand. Widgets know which subcomponents are necessary to assemble themselves.

1. Write a function: how-many-widgets-assembled
   Order **Widget** → Natural
   Returns how many widgets are able to be produced given all of the raw materials on hand. Note that a widget can only be constructed if *all* of its subcomponents can be assembled. Thus if a widget has two components A and B, and the supplier is able to produce 99 units of A, but 0 units of B, then 0 additional widgets are able to be constructed.
   Implement this either as how many could be made, or max(order size, how many could be made)

2. Write a function: how-long-to-produce
   Widget → Natural
   This function determines how long it will take to produce a new widget. You can assume there are separate factories for assembling each of the subwidgets, but those factories can only create one unit at a time. The time required is indicated in the time property for that widget (look back to the original data definition). Note that times are cumulative. Thus, if a widget takes 4 time units to assemble, and its components A and B take 6 and 9 units, respectively, to assemble, then in total it takes 13 time units to assemble one new widget (assuming A and B have no subcomponents of their own).
3. Write a function: fulfill-order
   Order **Widget** → Reply

A customer places an order indicating how many widgets he wants, and the absolute maximum amount of time he is willing to wait for that order. The Reply indicates how many widgets, up to the maximum requested, are able to be provided in that timeframe.

Hints

1. Conceptual complexity is higher on this assignment than the previous two.
2. Start by sketching out examples and ensuring you understand the data, and how widget production works.
3. Working on the parts in order, and together, will maximize the chance of success on this assignment.