Part 1 (4 pts)

1. Figure 7.3 on Page 500 defines the Function New, which creates objects from classes, and takes in an initial message.
(a) Write a function New2, which takes a class as input, and creates an object without an initial message.

(b) Write a function New3, which takes a class as input, and creates an object where the initial message in always the zero-arity record init.
(Hint, write the function for (a) and (b) in terms of New)

2. Figure 7.6 on Page 510 defines the class Account, and page 511 defines a subclass LoggedAccount.
(a) The class Counter from Page 498 is redefined on Page 499 without syntactic support. Do the same with the Account.

(b) Implement the LogObj used in LoggedAccount, where LogObj is an instance of the class LogTransfer. LogTransfer is a class with a single attribute log, which is a list containing the transfer values in order, and a single method addEntry, which takes transfer(Amt) as input, adds Amt to the end of log, and displays Amt with a Browse statement.

(c) Test the complete code and run several batchTransfer calls for an instance of Account and an instance of LoggedAccount. What is the behavior of batchTransfer for both objects? Answer this in terms of dynamic versus static binding as described on Page 512.

3. Using flexible argument lists with variable reference to method head from Page 505, extend the class Counter from Page 498 in the following way: change the attribute val to a record of counters (memory cells) with feature names from a … m, i.e,. val(a:(memory cell) b:(memory cell) … m(memory cell)). Change the method inc, to a method which takes in a flexible argument list that starts at a. i.e. inc(a:A …)=M. This method should loop through the features of M which correspond to the features of val, and increment the corresponding memory cells with e.g., M.a (the value corresponding to the feature in M).

Example:
Count is a Counter object with val(a:0 b:0 … m:0)
{Count inc(a:5 d:10)}
Count now has val(a:5 b:0 c:0 d:10 … m:0)

4. Using the wrapping methods described on Page 522-3 for TraceNew and TraceNew2, write a function that wraps an input object inside the class AttrCount. That is, write a function from objects to object, where AttrCount has the same structure as TraceNew2, but also contains a single attribute count, an additional method browseCount (Browsing the value of count), and modifies the otherwise case to increment the value of count for each time a Message is passed to the wrapped object.

Part 2 (4 pts)

Convert the C++ code lab12Market.cpp on Piazza into Oz. Run several tests and include them in your assignment file, to show that the code executes properly.

Important concepts:

- Static member variables -> in lab example
- Protected methods -> Page 515-6
- Private methods -> Page 514
- All instances of 'this->' can be ignored.