```
function [flag,msg,portfolioMod,accountMod] = sell( ...
    portfolio,exchange,account, ...
    symbol,numShares,commission,timeStamp)

    % This function will "sell" the desired
    % number of shares of the desired stock,
    % referred to by its symbol.  This means
    % the stock might be removed to the list of
    % stocks stored in the portfolio specified.

    % flag = 1 denotes success.
    % flag = 0 denotes failure.

    % Make sure more than zero
    % shares are being sold.
    % Otherwise exit.
    if(numShares <= 0)
        flag = 0;
        msg = 'Invalid number of shares!\n';
        portfolioMod = portfolio;
        accountMod = account;
        return;
    end

    % Check to see if the desired
    % stock is already present in the
    % portfolio.  "strcmp()" will return
    % a matrix of 1s or 0s corresponding
    % to the "symbol" being compared
    % to each element in the cell array
    % "portfolio.stockSymbols".  That
    % matrix is then searched with "find()"
    % for a "1", which will indicate the
    % index of that match in the stockSymbols
    % matrix.
    temp = strcmp(portfolio.stockSymbols,symbol);
    I = find((temp==1),1,'first');
    % Case where the stock is already in the
    % portfolio.
    if(~isempty(I))
        % Check to see if there are enough
        % shares in the portfolio to satisfy
        % the sell order.
        if(portfolio.stockShares(I) >= numShares)
            portfolio.stockShares(I) ...
                = portfolio.stockShares(I) - numShares;
            if(portfolio.stockShares(I) == 0)
                % If the zero shares of the
                % stock are currently owned,
                % DON'T remove it from the
                % portfolio.  This way its
```

```
                % data will be saved and
                % it can be bought again
                % under the same conditions
                % for buying if the price goes
                % down.

% % Remove the stock from the portfolio.
% portfolio.stockSymbols(I) = {};
% portfolio.stockShares(I) = [];

            end
        else
            % Error.  Can't sell more shares
            % than are in the portfolio.
            flag = 0;
            msg = 'Not enough shares to sell.';
            portfolioMod = portfolio;
            accountMod = account;
            return;
        end
    else
        % Error.  Can't sell a stock
        % that is not in the portfolio.
        flag = 0;
        msg = 'Stock not in portfolio.';
        portfolioMod = portfolio;
        accountMod = account;
        return;
    end

    % Record the date of the sell as the
    % last day of trading to date.
    portfolio.lastTradeDay_year = timeStamp(1);
    portfolio.lastTradeDay_month = timeStamp(2);
    portfolio.lastTradeDay_day = timeStamp(3);

    % Record information about the transaction.

    % Find the next empty row in the
    % transaction history list.
    nextTransNum = (size(portfolio.transactions,1) + 1);
    % Add the transaction information
    % to the list.
    portfolio.transactions{nextTransNum,1} = 'SELL';
    portfolio.transactions{nextTransNum,2} = timeStamp(1);
    portfolio.transactions{nextTransNum,3} = timeStamp(2);
    portfolio.transactions{nextTransNum,4} = timeStamp(3);
    portfolio.transactions{nextTransNum,5} = timeStamp(4);
    portfolio.transactions{nextTransNum,6} = timeStamp(5);
    portfolio.transactions{nextTransNum,7} = timeStamp(6);
    portfolio.transactions{nextTransNum,8} = symbol;
```

```
    [flag,tempStock] = getStockData_exchange(exchange,symbol);
    if(flag == 0)
        % Error.  Stock not found in exchange.
        flag = 0;
        msg = 'Stock not in exchange.';
        portfolioMod = portfolio;
        accountMod = account;
        return;
    else
        portfolio.transactions{nextTransNum,9} ...
            = tempStock.currentPrice;
    end
    portfolio.transactions{nextTransNum,10} = numShares;
    portfolio.transactions{nextTransNum,11} ...
        = (tempStock.currentPrice * numShares);

    % Update the portfolio with newly
    % calculated values.  a, b, and c are
    % dummies, just care about getting the
    % updated portfolio back from the function.
    [a,b,c,portfolio] = calcInvestment(portfolio,exchange);

    % Update the investment account
    % by adding/subtracting from
    % the balance.
    nextIndex = (length(account.year) + 1);
    account.year(nextIndex) = timeStamp(1);
    account.month(nextIndex) = timeStamp(2);
    account.day(nextIndex) = timeStamp(3);
    account.balance(nextIndex) =...
        (account.balance(nextIndex-1) ...
        + (tempStock.currentPrice * numShares) ...
        - commission);

    % Make sure to return the newly
    % modified portfolio struct
    % and account struct.
    portfolioMod = portfolio;
    accountMod = account;

    flag = 1;
    msg = 'Success!';
    return;

end
```