```
%% MAIN FILE 02:

% Units are $=USD.

% This is the main start-file for the
% random reactive stock simulation.

% Clear Freemat environment.
clear all;
close all;
clc;

% Print a few blank lines.
fprintf(1,'\n');

% Include external files.
% Because of how these files
% are written, variables are
% created with these names.
% Don't reuse these names in
% future code.
fprintf('Loading parameters...\n');
parameters;

% Create a portfolio to keep track of
% the stocks invested in.
fprintf('Creating portfolio...\n');
Portfolio_01 = createEmptyPortfolio('Portfolio_01');

% Create a performance struct to keep
% track of Portfolio_01 data over time.
fprintf('Creating performance database...\n');
Performance_01 = createEmptyPerformance(...
    'Performance_01','Portfolio_01');

% Create an investment account to keep
% track of investment amount and returns.
fprintf('Creating investment account...\n');
Account_01 = createEmptyAccount(...
    'Account_01');

% Set starting time and date for
% the simulation.
timeStamp = clock;
fprintf('Current time is %02d/%02d/%04d %02d:%02d:%02d\n',...
    timeStamp(2),timeStamp(3),timeStamp(1), ...
    timeStamp(4),timeStamp(5),timeStamp(6));

% Set the starting day of the
% simulation to "1".  Rather than
% keeping track of the calendar
% dates, for now the transaction
% dates will simply be "days after
% day zero".
startDate = [0 0 1 0 0 0];

% Day zero (for initializing
% stock exchange).
dayZero = zeros(1,6);

% Initialize investment
% account info.
Account_01.year(1) = dayZero(1);
Account_01.month(1) = dayZero(2);
Account_01.day(1) = dayZero(3);
Account_01.balance(1) = ACCOUNT_BALANCE_INIT;

% Create a stock exchange from where
% stocks can be "purchased" and added
% to the portfolio.  This stock
% exchange will only change when
% the function "updateExchange"
% modifies it.
fprintf('Creating stock exchange...\n');
Exchange_01 = createExchange_04(...
                NAME_TO_SIMULATE, ...
                SYMBOL_TO_SIMULATE, ...
                EXCHANGE_TO_SIMULATE, ...
                dayZero);

fprintf('Buying initial stocks...\n');
% Add initial stocks to the portfolio.
symbolToPurchase = SYMBOL_TO_SIMULATE;
[flag,stock] = getStockData_exchange(...
    Exchange_01,...
    symbolToPurchase);
if(flag == 0)
    % Handle error.
    fprintf('Failed to retrieve stock data from exchange.\n');
    return;
end
sharesToPurchase = floor(...
    INITIAL_PURCHASE_AMOUNT / stock.currentPrice);
% Buy.
[flag,msg,Portfolio_01,Account_01] = buy(...
    Portfolio_01,...
    Exchange_01,...
    Account_01,...
    symbolToPurchase,...
    sharesToPurchase,...
    TRADE_COMMISSION,...
    dayZero);
```

```
if(flag == 0)
    % Error.  Buy failed.
    fprintf(1,'\nError:  Buy failed.\n');
    fprintf(1,msg);
    fprintf(1,'\n');
    return;
end

% RUN SIMULATION:

fprintf('\nSimulation started...\n');

% This loop runs for a specific
% amount of time.  It modifies the
% properties of the stocks in the
% exchange.  Then the portfolio
% automatically decides how to react
% depending on the conditions set forth
% in the "parameters" file.  Each time
% a transaction is made, it is written
% to the portfolio struct and eventually
% exported to a CSV file.

% Duration in the form of
% Years, Months, Days, Hours,
% Minutes, Seconds.  Only
% use durations in DAYS
% until a function is written
% for tracking the actual
% calendar date.
duration = [0 0 SIMULATION_DAYS 0 0 0];
%                incrementTime(duration);
endDate = startDate + duration;
% Adjust endDate so last day is
% equal to the value of the duration,
% assuming day 1 is the first day.
endDate(3) = endDate(3) - 1;

% Uncomment this line to activate
% code to clear previous print
% line in real time.
%       reverseStr = '';

% Record initial performance
% of Portfolio_01.
Performance_01 = updatePerformance(...
    Performance_01,...
    Portfolio_01,...
    dayZero);

% Simulation loop.
```

```
for i = ((startDate(3)-1):(endDate(3)-1))

    % The simulation will have a
    % resolution of 1 day.  That is,
    % the simulated market will only
    % change/update once per day, at
    % which time the program will decide
    % how to react.  The first day of the
    % simulation will be day "0".

    % Keep track of the current day.
    currentDay = zeros(1,6);
    currentDay(3) = (startDate(3) + i);

    % Print feedback to the user to track
    % simulation progress.
    msg = sprintf('Now simulating day:\t%d\tof\t%d...\n',...
        currentDay(3),endDate(3));
    fprintf(1,msg);
% Attempt to clear previous print line
% in real time.  Found code online.
% Uncomment to turn it back on, but
% it doesn't work.
%   fprintf(1,[reverseStr msg]);
%   reverseStr = repmat(sprintf('\b'), 1, length(msg));

    % Update the exchange with simulated
    % market activity.


    % FIX THIS SO THAT STOCK PRICE
    % IS MORE STATIC AND SO THAT IT
    % NEVER GOES BELOW ZERO.


    Exchange_01 = updateExchange(...
        Exchange_01,...
        currentDay);
    % Update the portfolio by executing
    % buy and sell transactions based
    % on the market activity for the
    % last day in question.
    [Portfolio_01,Account_01] = updatePortfolio(...
        Exchange_01,...
        Portfolio_01,...
        Account_01,...
        currentDay,...
        TRADE_COMMISSION,...
        MIN_TRANS_PROFIT,...
        STOCK_AVG_WINDOW);
    % If account hasn't been updated,
    % update it.  This is to ensure
    % that even if there isn't a buy/sell,
    % the account still gets updated so
    % it has continuous data for
    % graphing later against other data.
    if(Account_01.day(end) ~= currentDay(3))
        % Update the investment account
        % by adding/subtracting from
        % the balance.
        nextIndex = (length(Account_01.year) + 1);
        Account_01.year(nextIndex) = currentDay(1);
        Account_01.month(nextIndex) = currentDay(2);
        Account_01.day(nextIndex) = currentDay(3);
        Account_01.balance(nextIndex) =...
            Account_01.balance(nextIndex-1);
    else
        % Do nothing.  The account
        % has already been updated
        % for the current day.
    end
    % Save current portfolio data to
    % the performance struct.
    Performance_01 = updatePerformance(...
        Performance_01,...
        Portfolio_01,...
        currentDay);
end

fprintf(1,'\nSimulation Complete!\n');

% Export data to files.
fprintf(1,'Exporting data to folder "./Simulation_..."\n');
exportData(Exchange_01,...
    Portfolio_01,...
    Performance_01,...
    Account_01,...
    timeStamp);

fprintf(1,'Exiting...\n\n');

return;
```