Important Notes:
There are three files that make up the code base for this program. This code outline is a simplified explanation of the various modules and how they work.

Zi.vb is the main file that starts the simulation.
Market_new.vb holds code implementing classes for the single security markets.
Trader.vb implements the trader class.

File: zi.vb
Module: zi
Specific Functions:
(only the substantive functions with non-trivial role in the software (e.g., initializing traders) are covered in this outline)

Subroutine RunSim

Purpose: Starting point for the program
Inputs: None
Code Explanation:

1) Check to see if a zi directory exists on the current computer, if not one is created
2) create files to record simulation data
3) create headers for the various files
4) for each replication
   - -A random gamma is generated
   - -Traders are initialized
   - -Market is run (RunMarket subroutine is called)
   - -data on replication is written to a file

Subroutine GenGammaDelta

Purpose: generates a random gamma value
Inputs: None
Code Explanation:

1) a random value of gamma is generated (uniform distribution between specified lower and upper limits)

File: Market_new.vb
Class: DAMarket (Double Auction Market)
Specific Functions:

Subroutine RunMarket

Purpose: Runs the generic double auction Market (2 state)
Inputs: Accepts an array of n traders
Code Explanation:

1) Defines various local variables to be used in the market
2) For each period
   - a) initialize a counter for the number of trades
   - b) initialize a blank array to hold trade details

c) determine the current state of the world (see GenCurrentState Function)
d) for each trader initialize its CAL (see InitCal Subroutine for details)
e) Within each iteration
- Computer generates a bid or ask (see GetAction subroutine)
- If the action is a bid, then a bid price is generated
- if the action is an ask, then an ask price is generated
- Bid (ask) Prices are compared to the highest (lowest) bid (ask). If they are higher (lower) then the bid (ask) becomes the new best bid (ask).

f) Market is cleared if possible
- If the highest bid is greater than the lowest ask, then a trade occurs at the mid-point of the bid and ask prices. Specifically the following occur:
-The seller of the security receives cash equal to the sale price
-The security is transferred to the buyer
-CAL for the security is updated for all traders
-Trade information is recorded in the data file

g) At the end of a period the end of period routine is called (EndOfPeriod)

Sub GetAction

Purpose: Generates random bid/ask prices for each trader
Inputs: None
Code Explanation: For each trader the following occurs:
-a random numbers is generated
-If the random number generated is higher than 0.5, then the action is an "ask", otherwise it is a bid.

Subroutine EndOfPeriod

Purpose: Performs the end of period cleanup
Inputs: Array of traders, period number, current state, array of trades, number of trades
Code Explanation:
1) Efficiency is calculated for the period (calls calc_eff)
2) Pays out the dividends to each trader
3) if trades occurred in the period, then statistics are calculated
4) Resets the number of tokens and cash back to their starting values
5) records period data to the output file

Function GetRE

Purpose: Calculates the RE equilibrium of a given set of traders which is defined as the maximum of the actual dividends available in the current state assuming that the information has been disseminated to all.
Inputs: Accepts an array of n traders
Returns: A single (real) number
Code Explanation: Take the maximum of the dividends under the current state across all traders if any trader is informed. If no trader is informed, take the maximum of expected dividends. Set RE equilibrium price in the current state to this maximum.

Function GetPI

Purpose: Calculates the PI equilibrium of a given set of traders defined as the maximum across all traders of the expected dividend payoffs (for the uninformed) and dividend payments under the current state (for informed).
Inputs: Accepts an array of n traders
Returns: A single (real) number
Code Explanation: take the maximum across all traders of the expected dividend payoffs (for the uninformed) and dividend payments under the current state (for informed).

Function CalcStats
Purpose: Calculates the applicable statistics for a period (mean, max, min, variance, opening and closing prices, the number of trades that converge to the PI vs. RE equilibria
Inputs: The array of trade data, the number of trades
Returns: Nothing
Code Explanation: Should be self-explanatory

Function CalcMean
Purpose: Determines the mean transaction price for a period
Inputs: Array of trades, number of trades
Returns: A single (real) number
Code Explanation: Self-explanatory

Function Calc_Variance
Purpose: Determines the variance of transaction prices for a period
Inputs: Array of trades, number of trades
Returns: A single (real) number
Code Explanation: Self-explanatory

Function Div_Paid
Purpose: Calculates the amount of dividends paid within a period
Inputs: Array of traders
Returns: A single (real) number
Code Explanation: Calculates the total amount of dividends paid by multiplying the dividends for each state by the number of tokens that each player holds

Subroutine ResetBidAsk
Purpose: resets the bid/ask values back to their initial starting points
Inputs: None
Code Explanation: The currentbid variable is reset to the lowest possible bid value (0) and the currentask variable is reset the maximum possible value (1)

Subroutine InitCal
Purpose: Sets the initial CAL for a player
Inputs: Trader, Period
Code Explanation: In the first period, the CAL must be initialized for each trader. If the trader is an insider, then they are told what the state actually is; therefore, CAL equals dividend payable in the current state. If they are uninformed, then the CAL is set to the expected dividend of the trader. For all other periods, the informed traders are told the actual dividend to be paid.

Function ExpDiv
Purpose: Calculates the expected dividend for a trader
Inputs: A trader
Returns: A single (real) number
Code Explanation: For uninformed players - calculate the sum of the multiple of the dividend for a particular state and the probability of the state. For informed traders, it is the dividend under the current state.

Subroutine Trade
Purpose: Completes the activities necessary for a trade to occur
Inputs: Traders
Code Explanation: First a token is transferred from the seller to the buyer, next cash is transferred to the seller from the buyer. The bid and ask prices are then reset.

Subroutine UpdateTransCAL
Purpose: Adjusts the CAL of a trader after a transaction
Inputs: Trader
Code Explanation: if the trader is not an insider, then the new CAL is determined by multiplying gamma times the transaction price and adding it to 1-gamma times the existing CAL.

Function GenCurrentState
Purpose: Generates a random state for a period
Inputs: None
Returns: An integer
Code Explanation: Generates a random number between 0 and 1. If the random number is greater than the probability of state 0, then the function returns state 1, otherwise state 0
occurs.

Function calc_eff
Purpose: Calculates the efficiency of a period
Inputs: an array of traders
Returns: A single (real) number
Code Explanation: efficiency is determined by dividing the total dividends paid by the total available dividends.

File: trader.vb
Class: trader
Specific Functions:
(only the main functions with non-trivial functions are covered)

Subroutine informTrader
Purpose: Gives information to the traders
Inputs: Current State
Code Explanation:
1) If there are only two states then the trader receives perfect information (i.e., is told exactly which state will occur with 100% probability