**Act.m**

%function takes in a single trade and returns whether it bought or sold

function [sold, bought] = act(trade1)

sold = 0;

bought = 0;

%switch case checks which moving average is in question and retrieves

%relevant historical information

switch trade1.movingAvg

case '1 min'

%trade1.paramas.Symbol = ticker symbol

%trade1.movingAvg = moving average

data = IBMatlab('action','history', 'symbol',trade1.paramas.Symbol,'barSize',trade1.movingAvg, 'useRTH',1,'duration', '420 S');

case '1 hour'

data = IBMatlab('action','history', 'symbol',trade1.paramas.Symbol,'barSize',trade1.movingAvg, 'useRTH',1,'duration', '21 D');

case '1 day'

data = IBMatlab('action','history', 'symbol',trade1.paramas.Symbol,'barSize',trade1.movingAvg, 'useRTH',1,'duration', '2 Y');

case '1 W'

data = IBMatlab('action','history', 'symbol',trade1.paramas.Symbol,'barSize',trade1.movingAvg, 'useRTH',1,'duration', '10 Y');

end

%Gets the current information about the stock in question

current = IBMatlab('action', 'query', 'symbol', trade1.paramas.Symbol);

%sends the highs, lows and closes from the historical data to EMA() which

%returns the averages for each candle

averages = EMA(data.high,data.low,data.close);

disp(trade1.paramas.Action);

%outer if statement checks to see whether you are trying to buy or sell

if strcmp(trade1.paramas.Action, 'buy')

%disp statements just to print out the prices that are being compared

disp('in buy')

disp('bid ');

disp(current.bidPrice);

disp('averages ');

disp(averages(1));

%if the current market price if above the most recent average then buy

if current.bidPrice > averages(1)

disp('bought');

%IBMatlab statement is just taking the trade parameters which has

%all the info needed to buy

IBMatlab(trade1.paramas);

%sets boolean value of bought

bought = 1;

else

disp('did not buy');

end

else

%program will go in here if the action is 'sell'

disp('bid ');

disp(current.bidPrice);

disp('averages ');

disp(averages(1));

%if market value if less than the most recent candle average then sell

if current.bidPrice < averages(1)

disp('sold');

IBMatlab(trade1.paramas);

%sets boolean value of sold

sold = 1;

else

disp('didnt sell');

end

end

end

**AWS.m**

%This function takes the allTrades vector as input and returns a new one

function newTrades = AWS(allTrades)

%It loops through every trade in the vector.

for i=1:length(allTrades)

%Boolean values to tell AWS whether or not the trade bought or sold

sold = 0;

bought = 0;

disp(allTrades(i).paramas.Action);

%Passes current trade to act() and act() tells us whether it bought or

%sold

[sold, bought] = act(allTrades(i));

%if act() sold then delete the current trade from allTrades

if sold == 1

allTrades(i) = [];

end

%if act() bought then change the 'buy' parameter to a 'sell' so next

%time it is checking the trade, it is trying to sell.

if bought == 1

disp('in change');

allTrades(i).paramas.Action = 'sell';

end

end

%returns the new vector

newTrades = allTrades;

end

**EMA.m**

%function takes in the highs, lows and close values and returns average of

%each candle

function [average] = EMA(highs, lows, close)

%initialize the empty totals vector

totals = [];

%for loop runs through all values passed to it

for i=1: length(highs)

%statement does (high + low + close) / 3 and adds it to the totals

%vector

totals = [totals,(highs(i) + lows(i) + close(i))/3];

end

%uses matlab function movavg to get the exponential moving average for each

%candle. I had to rotate the vector because that's how the movavg function

%wants it.

totals = rot90(totals);

average = movavg(totals,'exponential',length(highs));

end

**getGlobalIt.m**

function r = getGlobalT()

global T

r = T;

end

**setGlobalIT.m**

function setGlobalT(info)

global T

T = info;

end

**trade.m**

classdef trade

%class for storing the trades in a single object each that

%can easily be stored and retrieved with AWS

properties

paramas %Parameters for IBMatlab

movingAvg %The moving average it is based on

end

methods

% function obj = trade(inputArg1,inputArg2)

% %TRADE Construct an instance of this class

% % Detailed explanation goes here

% obj.params = inputArg1;

% obj.movingAvg = inputAvg2;

% end

%

% function outputArg = method1(obj,inputArg)

% %METHOD1 Summary of this method goes here

% % Detailed explanation goes here

% outputArg = obj.Property1 + inputArg;

% end

end

end