```
import requests
from time import sleep

s = requests.Session()
s.headers.update({'X-API-key': '31K6K3GF'}) # API Key from YOUR RIT Client 66666

MAX_LONG_EXPOSUE = 25000
MAX_SHOT_EXPOSUE = -25000
ORDER_LIMIT = 800

u_case = 'http://localhost:9999/v1/case'
u_book = 'http://localhost:9999/v1/securities/book'
u_securities = 'http://localhost:9999/v1/securities'
u_tas = 'http://localhost:9999/v1/securities/tas'
u_orders = 'http://localhost:9999/v1/orders'
u_news = 'http://localhost:9999/v1/news'
u_cancel = 'http://localhost:9999/v1/commands/cancel'

#################### CONSTANTS & INITIAL ####################

def get_tick():
    resp = s.get(u_case)
    if resp.ok:
        case = resp.json()
        return case['tick'], case['status']
def get_book(ticker):
    payload = {'ticker': ticker}
    resp = s.get (u_book, params = payload)
    if resp.ok:
        book = resp.json()
        return book

def get_bid_ask(ticker):

    book = get_book(ticker)

    bid_side_book = book['bids']
    ask_side_book = book['asks']

    bid_prices_book = [item['price'] for item in bid_side_book] #grabs all of bids in the book
    ask_prices_book = [item['price'] for item in ask_side_book] #grabs all of asks in the book

    best_bid_price = bid_prices_book[0]
    best_ask_price = ask_prices_book[0]
    return best_bid_price, best_ask_price

def get_position():
    resp = s.get (u_securities)
    if resp.ok:
        book = resp.json()
        return abs(book[0]['position']) + abs(book[1]['position']) + abs(book[2]['position'])

def get_time_sales(ticker):
    payload = {'ticker': ticker, 'limit': 1}
    resp = s.get (u_tas, params = payload)
    if resp.ok:
        book = resp.json()
        time_sales_book = [item["quantity"] for item in book]
        return time_sales_book

def get_news():
    resp = s.get (u_news)
    if resp.ok:
        news_query = resp.json()

        return news_query

def get_open_orders():
    resp = s.get (u_orders)
    if resp.ok:
        orders = resp.json()
        buy_orders = [item for item in orders if item["action"] == "BUY"]
        sell_orders = [item for item in orders if item["action"] == "SELL"]
        return buy_orders, sell_orders

def get_order_status(order_id):
    resp = s.get ('http://localhost:9999/v1/orders' + '/' + str(order_id))
    if resp.ok:
        order = resp.json()
        return order['status']

def get_cancelled(ticker):
    payload = {'ticker': ticker}
    resp = s.get('http://localhost:9999/v1/commands/cancel', params=payload)
    if resp.ok:
        cancell = resp.json()
        return cancell
```

```
def view_cancelled_orders(ticker):
    cancelled_orders = get_cancelled(ticker)  # Call your function
    print("Cancelled Orders:", cancelled_orders)  # Print the response

if __name__ == '__main__':
    ticker_symbol = 'AC'  # Replace with the desired ticker symbol
    view_cancelled_orders(ticker_symbol)

def get_ticker_list(): #return a full list of the securities on the market
    resp = s.get(u_securities)
    securities = resp.json()
    ticker_list = [item['ticker'] for item in securities]
    return ticker_list
##################### DEFAULT FUNCTIONS #####################

def get_trade_vol(ticker):
    book = get_book(ticker)

    bid_vol = 0.0
    ask_vol = 0.0

    for item in book['bids']: #pull the total bid volume of the ticker
        bid_vol += item['quantity'] - item['quantity_filled']

    for item in book['asks']: #pull the total bid volume of the ticker
        ask_vol += item['quantity'] - item['quantity_filled']

    return bid_vol, ask_vol
    #print('The bid vol for ' + ticker + ' is ' + str(bid_vol) + ' and the ask vol is ' + str(ask_vol))

def is_super_buyer(bvol, avol): #Check if its a strong buyer's market
    return bvol / avol <= 0.70

def is_super_seller(bvol, avol): #Check if its a strong seller's market
    return avol / bvol <= 0.70

def is_competitive(bp, ap): #Check if the bid ask spread is narrow
    return 0.03 >= abs(bp - ap)

def over_protection(limit):
    securities = s.get(u_securities).json()
    threshold = get_position() / limit
    if threshold > 0.50: #If the total position reaches 95% of the limit
        resp = s.post(u_cancel, params = {'all': 1})
        for item in securities:
            position = item['position']
            if position > 0.0: #Dump long position if the underlaying ticker has positive position
                resp = s.post(u_orders, params = {'ticker': item['ticker'], 'type': 'MARKET', 'quantity':
abs(position), 'action': 'SELL'})
                print(str(item['ticker']) + ' long Position Exploded, short dumping')
            elif position < 0.0: #Dump short position if the underlaying ticker has negative position
                resp = s.post(u_orders, params = {'ticker': item['ticker'], 'type': 'MARKET', 'quantity':
abs(position), 'action': 'BUY'})
                print(str(item['ticker']) + ' short Position Exploded, long dumping')


##################### STRAT FUNCTION #####################
def main():
    tick, status = get_tick()
    ticker_list = ['AC', 'RY', 'CNR']
    price_history = {ticker: [] for ticker in ticker_list} # To store recent prices

    while status == 'ACTIVE':
        for i in range(3):
            position = get_position()
            best_bid_price, best_ask_price = get_bid_ask(ticker_list[i])
            bid_vol, ask_vol = get_trade_vol(ticker_list[i])

            # Append current best prices to history
            price_history[ticker_list[i]]. append(best_bid_price)

            #Check the length of price history and trim it to the last 15 entries

            if len(price_history[ticker_list[i]]) > 15:
                price_history[ticker_list[i]].pop(0)

            # Check for a decreasing trend; ensure we have enough data (at least 2 points)
            if len(price_history[ticker_list[i]]) > 1:  # Only proceed if we have at least 2 prices
                if all(price_history[ticker_list[i]][j] > price_history[ticker_list[i]][j + 1]
                       for j in range(len(price_history[ticker_list[i]]) - 1)):
                    print(f"{ticker_list[i]} is in a decreasing trend.")
                    # Cancel any open sell orders if a downward trend is detected
                    buy_orders, sell_orders = get_open_orders()
                    for order in sell_orders:
                        cancel_resp = s.post(u_cancel, params={'order_id': order["id"]})
                        if cancel_resp.ok:
```

```
                        print(f'Cancelled sell order: {order["id"]}')
                    continue  # Skip this iteration if in a downtrend


            # if is_super_buyer(bid_vol, ask_vol):
            #     #Take advantage when it is a buyer's market
            #     resp = s.post(u_orders, params = {'ticker': ticker_symbol, 'type': 'LIMIT', 'quantity':
ORDER_LIMIT * 1.5, 'price': best_bid_price - 0.60, 'action': 'BUY'})
            #     print('A SUPER BUYERS MARKET FOR ' + str (ticker_symbol) + '! Longing ' +
str(ORDER_LIMIT * 1.3) + ' at ' + str(best_bid_price))
            # elif is_super_seller(bid_vol, ask_vol):
            #     #Take advantage when it is a seller's market
            #     resp = s.post(u_orders, params = {'ticker': ticker_symbol, 'type': 'LIMIT', 'quantity':
ORDER_LIMIT * 1.5, 'price': best_ask_price + 0.60, 'action': 'SELL'})
            #     print('A SUPER BUYERS MARKET FOR ' + str (ticker_symbol) + '! Shorting ' +
str(ORDER_LIMIT * 1.3) + ' at ' + str(best_ask_price))
            if is_competitive(best_bid_price, best_ask_price):
                #To fix the problem of 0 trade in RY when spread is extremely narrow
                resp = s.post(u_orders, params = {'ticker': ticker_list[i], 'type': 'LIMIT', 'quantity':
ORDER_LIMIT * 0.9, 'price': best_bid_price - 0.01, 'action': 'BUY'})
                resp = s.post(u_orders, params = {'ticker': ticker_list[i], 'type': 'LIMIT', 'quantity':
ORDER_LIMIT * 0.9, 'price': best_ask_price + 0.01, 'action': 'SELL'})
                print('Low spread for ' + str(ticker_list[i]) + ' long & shorting at a reduced price')
            else:
                resp = s.post(u_orders, params = {'ticker': ticker_list[i], 'type': 'LIMIT', 'quantity':
ORDER_LIMIT, 'price': best_bid_price - 0.02, 'action': 'BUY'})
                resp = s.post(u_orders, params = {'ticker': ticker_list[i], 'type': 'LIMIT', 'quantity':
ORDER_LIMIT, 'price': best_ask_price + 0.02, 'action': 'SELL'})
                print('Its nothing time, longing and shorting ' + str(ticker_list[i]) + ' at best b&a price')

            sleep (0.5)

            over_protection(MAX_LONG_EXPOSUE)

        tick, status = get_tick()

if __name__ == '__main__':
    main()

##################### MAIN FUNCTIONS #####################

    # position = get_position()

    # time_and_sales = get_time_sales('AC')
    # news = get_news()

    # best_bid_price, best_ask_price = get_bid_ask('AC')
    # resp = s.post('http://localhost:9999/v1/orders', params = {'ticker': 'AC', 'type': 'LIMIT',
'quantity': 500, 'price': best_bid_price - 5, 'action': 'BUY'})
    # resp = s.post('http://localhost:9999/v1/orders', params = {'ticker': 'AC', 'type': 'LIMIT',
'quantity': 500, 'price': best_ask_price + 5, 'action': 'SELL'})

    # best_bid_price, best_ask_price = get_bid_ask('RY')
    # resp = s.post('http://localhost:9999/v1/orders', params = {'ticker': 'RY', 'type': 'LIMIT',
'quantity': 500, 'price': best_bid_price - 5, 'action': 'BUY'})
    # resp = s.post('http://localhost:9999/v1/orders', params = {'ticker': 'RY', 'type': 'LIMIT',
'quantity': 500, 'price': best_ask_price + 5, 'action': 'SELL'})

    # resp.json()
    # order_id = resp.json()['order_id']

    # sleep(5)

    # resp = s.delete('http://localhost:9999/v1/orders/' + str(order_id))

    # resp = s.post('http://localhost:9999/v1/commands/cancel', params = {'all': 1})

    # resp = s.post('http://localhost:9999/v1/commands/cancel', params = {'ticker': 'AC'})

    # resp = s.post('http://localhost:9999/v1/commands/cancel', params = {'query': 'Volume > 0'})


    # tick, status = get_tick()
    # x = 0

    # while status == 'ACTIVE':

    #     print("While loop..." + str(x))
    #     x = x + 1
    #     tick, status = get_tick()


    # for i in range(9, 24):

    #     print(i)
    # i = 0
    # while i < 10:

    #     print(i)
    #     i = i + 1

    # x = 0
    # y = 10

    # if x > 0:

    #     print("x > 0")

    # elif x <= 0:

    #     print("x <= 0")

    # if y != 10:

    #     print("y does not equal 10")

    # if x > 0 and y > 0:

    #     print ("x and y are greater than 0")

    # if x > 0 or y > 0:

    #     print ("x or y are greater than 0")
```