```
import requests
from time import sleep

s = requests.Session()
s.headers.update({'X-API-key': 'UE4DD4XK'}) # API Key from YOUR RIT Client

MAX_LONG_EXPOSUE = 25000
MAX_SHOT_EXPOSUE = -25000
ORDER_LIMIT = 250

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_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 is_buyer_market(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 over_protection(limit):
    securities = s.get(u_securities).json()
    threshold = get_position() / limit

    if threshold > 0.95:
        resp = s.post(u_cancel, params = {'all': 1})
        for item in securities:
            position = item['position']
            if position > 0.0:
                resp = s.post(u_orders, params = {'ticker': item['ticker'], 'type': 'MARKET', 'quantity':
abs(position) * 0.2, 'action': 'SELL'})
                print(str(item['ticker']) + 'Long Position Exploded, performing short')
            elif position < 0.0:
                resp = s.post(u_orders, params = {'ticker': item['ticker'], 'type': 'MARKET', 'quantity':
abs(position) * 0.2, 'action': 'BUY'})
                print(str(item['ticker']) + 'Short Position Exploded, performing long')


###################### STRAT FUNCTION ######################
def main():
    tick, status = get_tick()
    ticker_list = get_ticker_list()

    while status == 'ACTIVE':
        for i in range(3):
            ticker_symbol = ticker_list[i]
            position = get_position()
```

```
            best_bid_price, best_ask_price = get_bid_ask(ticker_symbol)

            if is_buyer_market(ticker_symbol):
                resp = s.post('http://localhost:9999/v1/orders', params = {'ticker': ticker_symbol, 'type':
'LIMIT', 'quantity': ORDER_LIMIT, 'price': best_bid_price, 'action': 'BUY'})
                print('Longing ' + str(ticker_symbol))
            else:
                resp = s.post('http://localhost:9999/v1/orders', params = {'ticker': ticker_symbol, 'type':
'LIMIT', 'quantity': ORDER_LIMIT, 'price': best_ask_price, 'action': 'SELL'})
                print('Shorting ' + str(ticker_symbol))

            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")
```

NOTE: ***Given 1) and 2) it may be helpful to use both Net Position and Gross Position in your trading logic (i.e. IF statements). You will reach your Gross Position limit first (or tied with Net Position limit if trading only 1 stock), but if you trade over the Gross Position limit you will need to know if you are long or short if you want to reduce your Gross Position below the limit; alternatively, you can set different Gross and Net constraints.***

Possible solution(s): revise get_position() to return both Gross and Net positions, while incorporating both positions in your IF statements, for example - "if gross_position < 25000 and net_position < 24400:" then enter an order to buy 500 shares. If you have similar logic for your sell orders, you will not exceed the Gross Position limit, while still allowing the other trade direction to trade if you exceed the Net Position constraint in the IF statement.

- **DO more Trials, Run, Track , Edit → Record results on Gains. Run more trials on the bid-ask price margin changes, order limit quantity, separate stock tickers for order submitting and looping, ……. etc.**
- **Sleep time won't change substantially. :|**
- **Maybe think about why the professor's coding could keep making continuous monotone profits? Why it is not volatile? WHY does our code's performance follow a normal distribution centralized on a low mean P&L position???**