L7 Assignment

Based on L5 project, now we start to make some upgrades by using

- Use smart pointer to remove all raw pointers
- Factory pattern to create trades

Step 1:
Implement factory class to create different type of trades. Example code as below.

```
#pragma once

using namespace std;
// Abstract creator class
class TradeFactory
{
public:
    virtual shared_ptr<Trade> createTrade(const string &type, const Date &tradeDate, const Date
&expiryDate) = 0;
    virtual ~TradeFactory()
    {
        cout << "trade factor is destroyed" << endl;
    }
};

// Concrete creator class - create option type of trades
class LinearTradeFactory : public TradeFactory
{
public:
    shared_ptr<Trade> createTrade(const string& type, const Date& tradeDate, const Date& expiryDate)
override
    {
        if (type == "swap")
        {
            return make_shared<Swap>(tradeDate, expiryDate);
        }
        else if (type == "bond")
        {
            return make_shared<Bond>(tradeDate, expiryDate);
        }
        else
            return nullptr;
    }
};

// Concrete creator class - create option type of trades
class OptionTradeFactory : public TradeFactory
{
public:
    shared_ptr<Trade> createTrade(const string &type, const Date &tradeDate, const Date &expiryDate)
override
    {
        if (type == "european")
        {
            return make_shared<EuropeanOption>(tradeDate, expiryDate);
        }
        else if (type == "american")
        {
            return make_shared<AmericanOption>(tradeDate, expiryDate);
        }
```

```
        else
            return nullptr;
    }
};
```

Step2: Add constructor needed in swap, bond, European and American trade type.

```
Swap(Date start, Date end) : Trade("SwapTrade", start)
    {
        /*
        add constructor details
        */
    }
```

Step 3: Add setter function for all trade class, since the above trade factory construction function only set the start date and end date of trades. For example:

```
inline void setStrike(double _strike)
    {
        strike = _strike;
    };
    inline void setOptionType(OptionType type)
    {
        optType = type;
    };
```

Step 4: adapt main function accordingly.

```
vector<shared_ptr<Trade>> myPortfolio;
    auto lFactory = make_unique<LinearTradeFactory>();
    auto oFactory = make_unique<OptionTradeFactory>();
    Date tradeDate = Date(2023, 12, 31);
    Date expiryDate = Date(2025, 12, 31);
    auto swap = lFactory->createTrade("swap", tradeDate, expiryDate);
    auto bond = lFactory->createTrade("bond", tradeDate, expiryDate);
    auto eoption = oFactory->createTrade("european", tradeDate, expiryDate);
    auto aoption = oFactory->createTrade("american", tradeDate, expiryDate);

    auto eOpt = dynamic_cast<EuropeanOption>(eoption);
    eOpt->setStrike(100);

    myPortfolio.push_back(swap);
    myPortfolio.push_back(bond);
    myPortfolio.push_back(eoption);
    myPortfolio.push_back(aoption);
```