You need to modularize your source codes:

- ✓ Asset.h (defines Asset class and its derived classes: Equity, Preferred, & Bond), implementation details of these classes should be in Asset.cpp
- ✓ Position.h and Position.cpp
- ✓ PositionManager.cpp (main driver of the program)

A. Class inheritance [Asset.h & Asset.cpp]
   1. Base class Asset that contains
      a. Private members
         - assetType (e.g. "BOND")
         - currentPrice
         - securityIssuer (e.g. "U.S.Treasury")
         - securitySymbol (e.g. "IBM")
         - pHist – dynamic array to store historical price series
         - nPrices – number of prices in the historical price series
      b. Three parametrized constructors:
         - A parametrized constructor that initializes assetType from input and rest private data members to their default values, e.g. pHist = nullptr
         - A parametrized constructor that initializes (1) assetType, (2) securityIssuer, (3) securitySymbol from input and rest to their default
         - A parametrized constructor that initializes: (1) assetType, (2) securityIssuer, (3) securitySymbol, (4) pHist and nPrices
      c. Overload other special members:
         - Copy constructor (ensure deep copy)
         - Copy assignment (ensure deep copy)
         - Move constructor (ensure change control of resources instead of copying)
         - Move assignment (ensure change control of resources instead of copying)
      d. Public interface functions
         - Getters for all private members
         - Setter for currentPrice
         - Setter for nPrices and pHist
      e. Virtual functions
         - Destructor (make sure to clean the dynamic memory when pHist ≠ nullptr)
         - Get the assetType:
           ```
           virtual string getAssetType();
           ```
         - Get market value for a given number of assets:
           ```
           virtual double getMarketValue(const int& n);
           ```
         - Get market value for a given market price and a number of assets:
           ```
           virtual double getMarketValue(const double& price, const int& n);
           ```
   2. Derive from the base class the following classes:
      a. Equity class (assetType="EQUITY")
         - parametrized constructors (initialize all members + base class properly)
         - override the "getMarketValue" functions from base class

- overload operator<< to steam out (a) assetType, (b)securityIssuer, and (c) securitySymbol

b. Derive Preferred class from Equity class (assetType="PFD")
   - Private members:
     - Dividend per payment
     - Dividend payment frequency
     - Face value
   - parametrized constructors (initialize all members + base class properly)
   - overload operator<< to steam out (a) assetType, (b)securityIssuer, (c) securitySymbol, (d) dividend, and (e) face value

c. Bond class (assetType="BOND")
   - Private members:
     - Coupon rate
     - Coupon payment frequency
     - Par value
   - Parametrized constructors (initialize its own members + base class properly)
   - Override the "getMarketValue" functions from base class (bonds are quoted as percent of par value, so calculate this according to market convection)
   - overload operator<< to steam out (a) assetType, (b)securityIssuer, (c) securitySymbol, (d) Coupon Rate, and (e) Par Value

B. Write the Position class [Position.h & Position.cpp]
   a. Private data members:
      i. Pointer to base class Asset (this is used for polymorphic purpose, not dynamic memory allocation)
      ii. Position size (e.g. number of shares, number of bonds etc.)
      iii. Cost basis (expressed as per unit cost)
      iv. Age of the position (in unit of years)
   b. Public functions
      i. Default constructor (no parameter)
      ii. Parametrized constructors
   c. Public interfaces
      i. Getters and setters
      ii. Current market value of the position (use getMarketValue function from Asset class)
      iii. total dollar cost basis of the position (use getMarketValue function from Asset class)
      iv. annualized price return of the position
   d. overload operator<< to stream out (a) assetType, (b) securitySymbol, (c) current market value of the position, and (d) the annual return of the position since position inception

C. write main function [PositionManager.cpp]:
    a. Read from the attached security data files (equityData.txt, pfdData.txt, bondData.txt) and instantiate appropriate security for each of the entries in these 3 files.
    b. Declare an array of 11 Position. Use the attached portfolio data file, port_data.txt, to fill each of the 11 Positions with these investments
    c. Calculate the market value weighted annualized portfolio return and output the result to an output file
    d. Output each of the 11 securities in the portfolio to the same output file

D. Submit output file along with all of your source code

Congratulations for finishing a great semester!