A. Point class:
    a. Write the class *Point*
        i. Private data members: x, y
        ii. Public member functions (use the keyword *const* where appropriate):
            1. Constructors
                a. Default constructor that initialize the x and y to zero
                b. Parametrized constructor to initialize x and y to user provided values
            2. Setter and getters for the x and y
            3. Interface functions:
                a. distance_to_origin(): returns the distance of point (x,y) to (0,0)
                b. distance_to_point(const Point&p): returns the distance of point (x,y) to point p
                c. overload `operator<<` to out stream the x, y coordinates of Point as (x,y)

    b. In main function, using a while loop
        i. Ask user to input coordinates x and y
        ii. Initialize a point, A, with the user input
        iii.
        iv. Print out the distance of A to origin
        v. Declare a fix-sized array of 5 Point
        vi. Use user-input to assign x, y coordinates to these 5 points
        vii. Lastly, print into an output file the distance of each of the 5 points to point A with a message like this (assume point A is at (1,2), and one of the 5 point is at (4,6))

            Distance from (4,6) to (1,2) is 5

B. Implement and test TimeSeries class
    a. Write the class TimeSeries
        i. Private data members:
            1. *pSeries: the dynamic array to store the time series
            2. seriesLen: the length of the time series
        ii. User-supplied constructors:
            1. Default constructor: initialize the pSeries to nullptr and SeriesLen to 0
            2. Parameterized constructor: TimeSeries(const double* p, int n), where p is a user supplied array of size n
        iii. Setter: setSeries(const double* p, int n)
        iv. Getter:
            1. Write a read-only function *getSeriesLen* for the series length
            2. Write a read-only function *getSeries* that returns the series in a read-only fashion, i.e. the returned pointer CANNOT modify the internally stored series
        v. Interface functions:

1. Write a function, calcAvgReturnVol, to calculate the average return of the stored time series and the volatility of the return series
2. Write a function, findMaxMin, to find the maximum and minimum value and their respective locations in the time series

b. Test your class with provided data series:
   i. Data series, TS_A
      1. Read the provided data, TS_A, into your program
      2. Initialize a TimeSeries object with the data series
   ii. Data series, TS_AMZN
      1. Declare an empty TimeSeries object
      2. Read the provided data, TS_AMZN, into your program
      3. Use the setSeries member function to copy the data series into the above TimeSeries object
   iii. Calculate and print out the average return of TS_A and its return volatility
   iv. Find and print out the maximum and minimum of TS_AMZN series and their locations in the series

C. Compile and run your programs, submit your source code and results on line

Happy coding!