CIS 2541: C++ Language Programming
Program Assignment

Name: _____

Program Assignment	#3 – Arrays, Sorting, Searching, and Pointers
Due Date *(beginning of class)*	04/13/2016
Points	**Stock LookUp** C++ Program header _____/ 5 pts. alignment _____/ 5 pts. file input _____/ 10 pts. pointer memory _____/ 15 pts. functions _____/ 25 pts. accuracy _____/ 10 pts. output formatting _____/ 10 pts. demo _____/ 10 pts. **Total**** _____**/ 90 pts.** ***60 point maximum for code that cannot compile or execute.*

On your own, create a modular C++ program to lookup the trading symbol in a given stock portfolio. The program will use **dynamic** parallel arrays to hold the stock information: trading symbol, company name, number of shares and current price. **Do not use global variables or constants in your solution.**

Stock portfolio information will be read from file and can contain ***any number*** of stocks. The first line of the file contains an integer identifying the remaining number of stocks to read from the file. Each remaining line in the file contains data for a single stock (data types in parentheses): trading symbol (multiple characters), company name (multiple characters with embedded spaces and end delineated by #), number of shares (integer), and current price (real). A sample stock file is given below:

```
5
MCD McDonald's Corporation# 35 99.25
HD The Home Depot, Inc.# 31 115.54
F Ford Motor Co.# 288 16.49
WMT Wal-Mart Stores Inc.# 10 83.33
BA The Boeing Company# 24 151.46
```

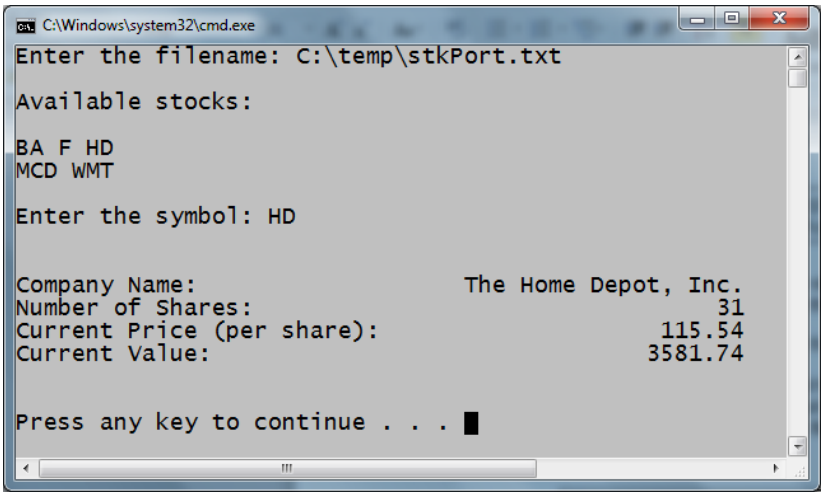
a) Create the source code file **StockLookUp.cpp**. Add a block comment at the top of the file to identify your name, file, date, class, assignment, and short description of the program.

b) Include the following design requirements in your code:

- Prompt and read a filename from the user. If there is an error opening the file, exit the program with an error code of -1.
- If the file is successfully opened for input, use the integer on the first line to **dynamically** create parallel arrays of the appropriate size and data type for each of the stock fields: trading symbol, company name, number of shares, and current price. Use either C++

pointers or C++ 11 smart pointers to reference the parallel arrays. Read the remaining stock data from the opened file into the created parallel arrays. Read until the end of file or the dynamically allocated arrays have been fully populated, whichever comes first. Remember to close the file after reading is complete.

- Create an array of pointers to the trading symbols. Sort ***only*** this array of pointers and display all the sorted trading symbols, 3 to a line, using this array of sorted pointers.
- Prompt the user for a trading symbol to lookup. Perform a **case insensitive** search of the trading symbol array for the user input trading symbol (*do not search the array of sorted pointers to trading symbols*). If the case insensitive trading symbol is found, display the stock information for the found trading symbol. Current value of the stock is defined as number of shares times current price per share. Format your session as given in the following example (previous data file used):



If the trading symbol is not found, display an error message and exit with an error code of -2.

- Create and use the following functions in your solution:
  - ✓ Function to sort array of pointers to trading symbols for subsequent display of sorted trading symbols 3 to a line (Hint: See Section 9.11 for example on sorting array of pointers).
  - ✓ Function to perform case insensitive search of trading symbol array for a given trading symbol and return index of found symbol or -1 otherwise.
  - ✓ Main function to open file, create and manage dynamic arrays, read data from file into arrays, display sorted trading symbols, prompt user for trading symbol to search, and display result of trading symbol search.

c) Use proper alignment and spacing in your source code. Use textbook examples as a guide.

d) Compile the source code until no syntax errors are found. Run the program with a given data file and manually verify the expected and actual results. Debug as appropriate to ensure accuracy.

e) Copy and paste the generated output from the data file as a multi-line comment at the end of the source code file.

f) Attach a copy of the source code file(s) and generated output to this sheet for full possible points as indicated above Upload a copy of ***only*** your source code file in Blackboard under the appropriate program assignment. Multiple source code files must be combined into a single .zip file before upload.

g) Be prepared to give a demo of your code and describe its functionality at the ***beginning*** of class on the listed due date. Note that the demo data file will be *different*, but with the *same layout*, as the one provided with the program assignment.