PIC 10B Lectures 1 and 2 Spring 2015
Homework Assignment #5

Due Friday, May 8, 2015 by 5:00pm.

Objectives:

1. To gain some experience implementing a sorting algorithm.
2. To count and compare the number of statements executed by different algorithms used to sort the same array.

Introduction:

Shell sort is a generalization of the insertion sort algorithm. At each stage, shell short uses the insertion sort algorithm to make an array h-sorted for a given step size of h. An array is **h-sorted** if every subarray consisting of all elements whose indices differ by some multiple of h is sorted. Shell sort will h-sort the array for a sequence of h values that decrease to 1. When h is 1, shell sort just performs insertion sort on the entire array.

In this assignment, you will be implementing a shell sort function with prototype

```
void shell_sort(int a[], const int size, int& statements);
```

The Shell Sort h-sequence to use is 9841, 3280, 1093, 364, 121, 40, 13, 3, 1.
In addition, you will be modifying a library of sorting functions by introducing a reference parameter called `statements` for each function This parameter will count the number of statements and comparisons executed by the function. Examples of statements to count are

- variable declarations
- for loop initializations
- if, while, do while, and for condition tests (count both when condition evaluates to true as well as when it evaluates to false)
- updates to counter variables (especially in a for loop) eg i++
- assignments
- statements within a function call eg (a swap call counts as three statements)
- increments of an index eg a[i++] = 3; counts as two statements
- a recursive call (counts as one statement since you will be passing in variable statements into the recursive call to keep track of its statements.)

Finally, you will be completing the implementation of the application AnalyzingAlgorithms.cpp which will perform bubble sort, selection sort, insertion sort , merge sort, and shell sort on the same array of N random integers whose elements range from 1 to 10000. For each sorting algorithm, it will output the original array to a text file, sort the array (while keeping track of the number of

statements), and then output the sorted array to the text file. It will then output the number of statements it counted for that sorting function to a different file that will create a table of statement counts for each algorithm for each value of N.

There will be no output sent to the screen. Instead there will be a text file for each sorting algorithm (eg insertionsort.txt), recording the arrays before and after each sort. There will also be a text file StatementCount.txt holding the table of statement counts and another file AlgorithmAnalysis.txt which will hold the table of ratios T(2N)/T(N) which will help us deduce O(T(N)) for each algorithm.

Here are the descriptions of the functions in AnalyzingAlgorithms.cpp which you will need to implement:

- `void performSort(void (*sort)(int[], const int, int &), int data[], const int N, string filename, ostream &os )`
  Creates an ofstream for the given filename and uses it and print_ array to append the first N elements of given array data to the file. Create a counter variable initialized to 0 and use the given sorting function pointer to sort the first N elements of array data while storing the number of statements made by the sort in the counter variable. Then print out the sorted array to the same file. Finally output to the ostream os the number of statements made by the sort (right aligned and in an column of width BN_COL_WIDTH)

- `void makeRecord()(int N, ostream &os)`
  Creates an array of ints called data and 5 clones of that array (one for each sorting algortihm). Give all 6 arrays the same N random integer values. Use the given ostream os to output the value of N (right aligned and column width N_COL_WIDTH.) Call performSort for each sorting algorithm using the given N and individual file names (such as "insertionsort.txt" for insertion_sort). Then output a newline to finish the record (which will be a row in the statement count table)

- `void sortAndCount(string count_file)`
  Creates an ofstream for the given filename count_file.
  Use the ofstream to print out the header
  "`***Number of Statements for Various Sorting Algorithms***`"
  as well as the table header (see screenshot below for StatementCount.txt). Then use makeRecord to create each row of the table. Close the stream at the end.

- `void analyzeCounts(string count_file, string analysis_file)`
  Create an ifstream to read the table data giving the number of statements for each algorithm (the T(N) values) contained the file with name count_file and use that information to compute and store the ratios of T(2N)/T(N) in another table in the file having name given by `analysis_file`. See the screenshots below for what the output in `analysis_`file may look like.

Directions:

1. Create the project called “Hw5” in a solution called “Homework” using Microsoft Visual Studio 2012. All header (.h) and source (.cpp) files inside the project should contain the following header:

```
/*
   <Your Name>                    PIC 10B Intermediate Programming
   ID: <Your Student ID>          Spring 2015
   Email: <Your Email Address>    Assignment #5
   Section: <Your Section # eg 1A>
   Honesty Pledge:

      I, <Your Name Here>, pledge that this is my own independent work,
   which conforms to the guidelines of academic honesty as described in
   the course syllabus.

   List of known bugs:  <Known bugs, if any>
*/
```

2. Modify the Algorithms.h header file to include the prototype
   `void shell_sort(int a[], const int size, int& statements);`
   and add an `int` reference parameter called `statements` to the other prototypes in the header file. You will also need to add a `statements` reference parameter to the `merge` function definition and to the other sorting function parameter lists in implementation file Algorithms.cpp.

3. Implement the shell_sort function in file Algorithms.cpp.

4. Modify the definitions of the other functions in file Algorithms.cpp so that the reference parameter `statements` actually counts the number of statements its function makes.

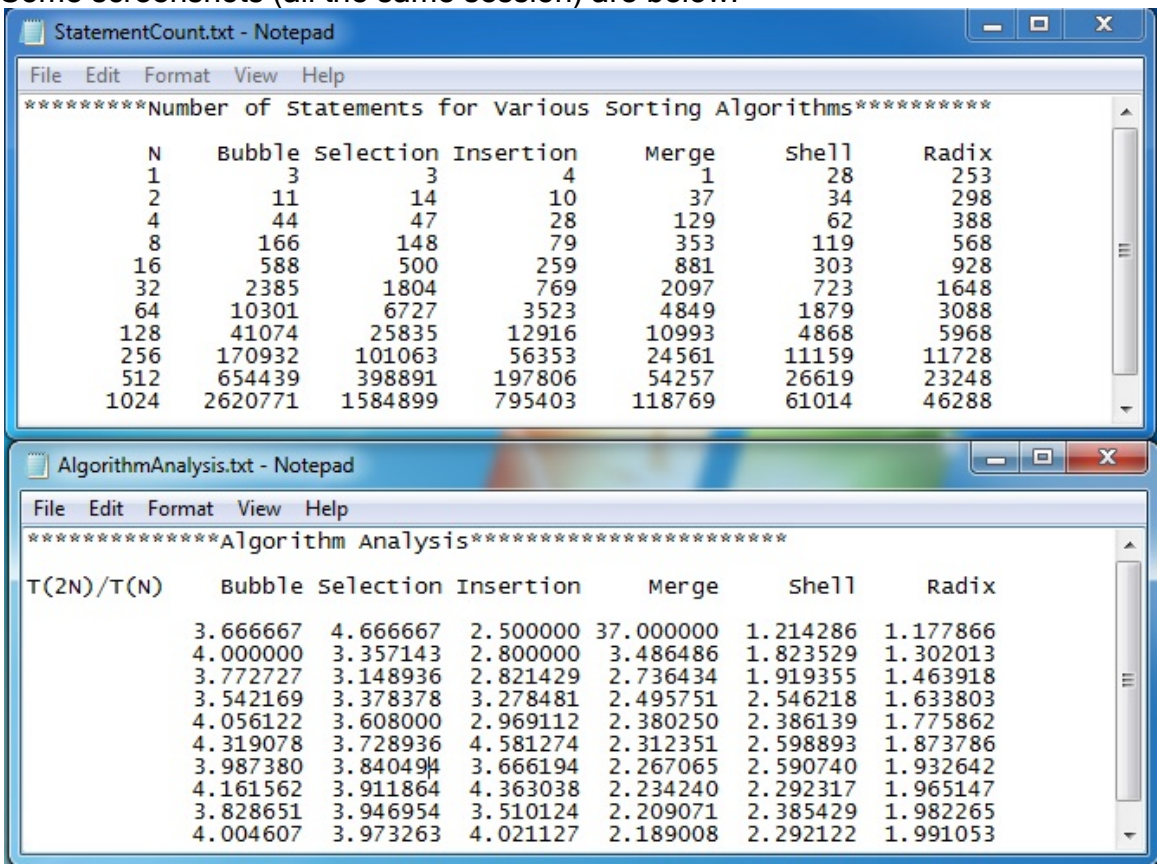
5. Implement the functions below in the application file AnalyzingAlgorithms.cpp
   - `performSort(void (*sort)(int[], const int, int &), int data[], const int N, string filename, ostream &os )`
   - `void makeRecord()(int N, ostream &os)`
   - `void sortAndCount(string count_file)`
   - `void analyzeCounts(string count_file, string analysis_file)`

6. When you have completed your project, be sure to
   - make sure your program compiles in Visual Studio 2012.
   - run your program to make sure it works correctly
   - upload your source code files
     - Algorithms.h
     - Algorithms.cpp

- AnalyzingAlgorithms.cpp

  using the CCLE website. No hardcopies will be collected.
- visually verify that your source code was submitted correctly by clicking on the links to those files on the CCLE page after submission.

Some screenshots (all the same session) are below:



StatementCount.txt - Notepad

File Edit Format View Help

```
*********Number of Statements for Various Sorting Algorithms**********

         N    Bubble Selection Insertion     Merge     Shell     Radix
         1         3         3         4         1        28       253
         2        11        14        10        37        34       298
         4        44        47        28       129        62       388
         8       166       148        79       353       119       568
        16       588       500       259       881       303       928
        32      2385      1804       769      2097       723      1648
        64     10301      6727      3523      4849      1879      3088
       128     41074     25835     12916     10993      4868      5968
       256    170932    101063     56353     24561     11159     11728
       512    654439    398891    197806     54257     26619     23248
      1024   2620771   1584899    795403    118769     61014     46288
```

AlgorithmAnalysis.txt - Notepad

File Edit Format View Help

```
**************Algorithm Analysis*********************

T(2N)/T(N)     Bubble Selection Insertion     Merge     Shell     Radix

             3.666667  4.666667  2.500000 37.000000  1.214286  1.177866
             4.000000  3.357143  2.800000  3.486486  1.823529  1.302013
             3.772727  3.148936  2.821429  2.736434  1.919355  1.463918
             3.542169  3.378378  3.278481  2.495751  2.546218  1.633803
             4.056122  3.608000  2.969112  2.380250  2.386139  1.775862
             4.319078  3.728936  4.581274  2.312351  2.598893  1.873786
             3.987380  3.840494  3.666194  2.267065  2.590740  1.932642
             4.161562  3.911864  4.363038  2.234240  2.292317  1.965147
             3.828651  3.946954  3.510124  2.209071  2.385429  1.982265
             4.004607  3.973263  4.021127  2.189008  2.292122  1.991053
```

Grade Breakdown:

Criteria	Description	Points
Header	Starts every .h and .cpp file	1
Comments	Program well-commented.	1
Algorithms.h	Declarations correct	2
Algorithms.cpp	`shell_sort` implemented correctly	10
	Definitions of other functions modified to count statements	5
AnalyzingAlgorithms.cpp	`performSort`	2
	`makeRecord`	2
	`sortAndCount`	2
	`analyzeCounts`	5
Total		30

A penalty of 5 points will be assessed if your code does not compile using Microsoft Visual Studio 2012.