CSE 373 Homework 6 Project Description

Assigned: Wednesday, May 15, 2002
Due: Wednesday, May 22, 2002
At the start of class

Introduction

For this homework project, you will develop several of the sort routines that we have discussed in class and answer some questions. All the source files that we are providing for this assignment are in a zip file on the web site. You should download and unzip the file. The questions were handed out in class; you can also get a copy from the web site.

Sort. The mainSort program reads the familiar symbol table files. mainSort calls your sort routines to sort the symbol arrays generated from the symbol tables. This process repeats until all the files named on the command line have been processed.

The main program for the homework is supplied in ADT/Sort/mainSort.c, and you are to write the individual sort functions as defined in ADT/include/sort.h. As usual, the functions are based very closely on the discussion in the textbook, however, they are not exactly the same.

Grading

The 7 homework assignments of the quarter will count for a total of 50% of your class grade, and each individual homework assignment will count for about 7% of the total class grade. The grading for this project is as follows.

Questions:	10 points
Implementation:	10 points
Total	20 points

NOTE: You need to turn in several things:

1. the paper copy of your answer sheet
2. a paper copy of the tabular timing data
3. a paper copy of a plot of the timing data
4. a copy of the receipt you got when you did the web turn in
5. do a web turnin of your implementation of sort.c.

Staple the everything together and turn them in on Wednesday.

Directory Structure

All of the homework projects are implementations of one of the Abstract Data Types that we discuss in class. The directory structure of the files you receive is as follows:

ADT/	top level directory
ADT/include	header files
ADT/symbols	example symbol table files
ADT/Sort	directory for the homework project

Program: Sort

The purpose of this program is to read a symbol table file and create an array from it, then sort the array and record data about how long it took. Input is taken from a list of symbol table files provided on the command line, output is to stdout (which can be redirected to a disk file).

This program requires you to implement five sort routines: InsertionSort, ShellSort, HeapSort, MergeSort, and QuickSort. The main program and the header files are supplied. The implementation file "sort.c" is where your code goes.

To do:

1. Create the Sort project just as you have done in the previous homeworks. For the Program arguments entry, use "..\symbols\U-sym10.txt".
2. Note that although the file "sort.c" is provided to you, it is very incomplete, and so the project will not run correctly. All the routines that you will implement are missing an actual implementation.
3. Write the five sort routines, rebuild the project, and run it. Debug until done.
4. If you run the project from Visual C, the output gets displayed in a console window. Once you have the program running so that it sorts the small file correctly, you should run it with the larger examples and store the results. I have supplied a batch file called winrunit.bat that runs your program (Debug\sort.exe) from a command line using five different input files of increasing size. The batch file stores the output in winrundata.txt. To use this batch file, get a command prompt (Start->Programs->Accessories->Command Prompt) and use the cd command to move to the ADT\Sort directory. Then type "winrunit". There is a similar file called runit for running the program in Linux.
5. Review the code and the output files as needed in order to answer the questions in the homework.

The function headers for the routines you need to write are in ADT/include/sort.h. Note that the functions are very similar, but NOT IDENTICAL, to the ones defined in the textbook. The functions are as follows.

File sort.c:

void InsertionSort(ElementType A[],int N, Comparator C);

The basic insertion sort that I have described in class and that is described in the book. Note that you need to use the Comparator C in order to compare elements.

void ShellSort(ElementType A[], int N, Comparator C);

The shell sort routine should be implemented using Hibbard's increments as described in the text. Your can take $\log_2(N)$ in order to get the initial value of k, for use with Hibbard's 2^k-1 formula. Remember that $\log_2(x) = \log(x)/\log(2)$, where the log function is taking the logarithm to the base *e*.

void HeapSort(ElementType A[], int N, Comparator C);

Implement the Heap Sort as given in the book, modified to use our Symbol Comparators.

void MergeSort(ElementType A[], int N, Comparator C);

Implement the recursive Merge Sort as given in the book, modified to use our Symbol Comparators.

void QuickSort(ElementType A[], int N, Comparator C);

Implement the recursive Quick Sort as given in the book, modified to use our Symbol Comparators. Use 20 as the value at which you switch to using Insertion sort for small arrays.

Sample output

The main program produces a data table very similar to figure 7.19 on page 256. For each file named on the command line, it reads in the symbols, then calls each of your sort functions and times them. This takes about 5 seconds per function per file, so don't be alarmed when your program seems to die the first time you run it. When you run the batch file, it names 5 files, and runs each of the 5 sorters against those five files, so the total execution time is about 2 minutes.

A sample output file (rundata.txt or winrundata.txt) is

```
Count,Insert,Shell,Heap,Merge,Quick,Filename
10,0.00001,0.00001,0.00002,0.00002,0.00001,..\symbols\U-sym10.txt
100,0.00081,0.00035,0.00044,0.00038,0.00030,..\symbols\U-sym100.txt
1000,0.09404,0.00723,0.00755,0.00566,0.00513,..\symbols\U-sym1K.txt
10000,,0.14876,0.12285,0.08395,0.08412,..\symbols\U-sym10K.txt
100000,,3.21450,1.81600,1.17980,1.18180,..\symbols\U-sym100K.txt
```

The data is in CSV format, or "comma separated variables". Most spreadsheet programs can read this data very easily, as described below. Your task is to make sure that your functions are sorting the data correctly, and then analyze the timing results.

If the data is not in sorted order after your sort function is done with it, there will be a negative number in the spot where the timing data for that function should be. If the data is sorted correctly, then a floating point number is presented which is the approximate time that it took for your function to sort the data in one pass. The main program calls your function over and over again in order to generate this number, that is why it takes about 5 seconds to generate each of the values.

Reading the data into Excel and Plotting It

These instructions are based on Excel 97. If you have a different version, the menus and toolbars may vary, but the capabilities should be similar. Also, other spreadsheets should be able to draw the same simple charts.

Start the Text Import Wizard by selecting Open from the File menu, then selecting Files of type "Text Files", and selecting the rundata.txt or winrundata.txt file to Open.

Step 1. The file type is Delimited, which should be the default. Start the import at row 1 (also the default).

Step 2. Set the delimiter to be "comma". Do not select "treat consecutive delimiters as one".

Step 3. General format is okay for everything. Click finish.

You should have a table that looks something like this.

Count	Insert	Shell	Heap	Merge	Quick	Filename
10	0.00001	0.00002	0.00003	0.00004	0.00001	../symbols/U-sym10.txt
100	0.0013	0.00058	0.00072	0.00068	0.00042	../symbols/U-sym100.txt
1000	0.16613	0.01348	0.01335	0.01118	0.00747	../symbols/U-sym1K.txt
10000		0.24333	0.21375	0.16733	0.11651	../symbols/U-sym10K.txt
100000		5.62	3.355	2.23333	1.55	../symbols/U-sym100K.txt

Select all the cells in your spreadsheet except the column with the file names in it.

Start the Chart Wizard by clicking on the little bar chart in the main toolbar or by selecting it in the Insert menu.

Step 1 – Chart Type
chart type - XY (Scatter)
chart sub-type – data points connected by lines (not smoothed lines)

Step 2 – Chart Source Data
Data Range Tab: Series in (*) Columns
Series Tab: you should have Series named Insert, Shell, Heap, Merge, Quick

Step 3 – Chart Options
Chart Title: your name, plus whatever else you want
Value (X) Axis: Symbol Count
Value (Y) Axis: Time in Seconds

Step 4 – Chart Location
As new sheet

Click Finish. This will create the plot.

You can use logarithmic axes to more clearly see the data. To change the scaling, you can select the axis (Value (X) Axis or Value (Y) axis) then select format Axis from the format menu or the Chart Toolbar. Select the Scale Tab, then click in the logarithmic scale checkbox to change the scaling.

Depending on your printer, you may want to change the background of the chart to "none". Click in an open area in the chart to select "Plot Area", then right click and select "Format Plot Area." In the Area part of the Patterns tab, select "none".

You may also need to change the data line colors. Click on the data series line that you want to change, then right click and select Format Data Series. In the Line part of the Patterns tab, select a new color.

You should have a plot that looks something like this.