CSCE 310 Data Structures & Algorithms

Fall 2004
Steve Goddard
Homework 4, November 2, 2004
(Total of 100 points)

Chapters 5 and 6

Due: 9:00pm Tuesday, November 16, 2004

Exercises From the Book (50 points):

5.2 #3 (10 pts)
5.3 #7 (10 pts)
6.3 #7.a (5 pts)
6.4 #7 (5 pts)

Additional problems:

A) (10 pts) Define and write the pseudo code for the BST predecessor ADT.

B) (10 pts) Write the iterative (loop) version of the recursive BST Search pseudo code presented in class.

Programming Exercise (50 points):

Search, Insert, Delete, do it again, then Sort!

Your assignment, should you decide to accept it, is to create a program that reads company names from a file, stores them into a data structure, iteratively and interactively inserts and deletes various companies from the set, and prints out an alphabetically sorted list of the company names. You choose the algorithms and data structures to meet the requirements. However, you must justify your selection of the algorithms and data structures. (Hint: typically balanced search trees are used for this type of problem. Why?)

The program must meet the following requirements:

a. Your C++ executable program will be named `DynamicSet`.

b. It will accept one input parameter on the command line: the name of the ASCII input file of companies to analyze in the input file. The calling format is

```
DynamicSet <inputFile>
```

`<inputFile>` is the name of the file that contains names of companies. The first five lines of a sample test file (retrieved from the course web page) are:

```
WALTER INDUSTRIES INC
AGWAY INC
TEXTRON INC
CARLISLE COS INC
LEUCADIA NATIONAL CORP
```

For example, executing your program with the command:

```
DynamicSet InputFile
```

means that the file `InputFile` will be read from the file and the data stored as a dynamic set.

c. After reading the file, the program will interactively prompt for and accept commands from standard input to either insert or delete companies and their associated data from the set of entries. The set of commands accepted will be:

```
insert CompanyName

delete CompanyName

quit
```

   Only one entry can be inserted or deleted per command.

d. Upon termination of the interactive portion (i.e., the command quit is entered), your program will then print to standard output an alphabetically sorted list of the company names. The output should be of the following form (left justified):

```
**Sorted List of Companies**
<companyName1>
<companyName2>
      …
<companyNameN>
```

e. The coding standard must be followed.

f. Create and turn in a Makefile that will build your executable from your source file(s).

You may download a test input file from the course Web page. Its URL is
http://cse.unl.edu/~goddard/Courses/CSCE310J/Assignments/CompanyNames.csv
The test input file is based on data downloaded from the Compustat Web site for 9,050 companies.

You may use code from the Internet or any textbook. However, you must cite the source if you did not create it. If you use existing code, you must also include pseudo code for the algorithm in your report. If you implement pseudo code from the book, you simply to need to cite the figure and page number, and identify any deviations from the code. You are responsible for understanding the code you use and the basic algorithm.

Analysis: Analyze the Best, Worst, and Average case time complexity for the insert command, the delete command, and generating the final alphabetical list of company names, assuming the input file may be in any order (e.g, the input file may or may not be sorted by company name).

The analysis report should also describe how your program works, what design choices (if any) were made and why, and any known defects. Defend your choice of algorithms for insertion, deletion, and generating the sorted list of entries in the set. You may also discuss enhancements you think should be made, but were beyond the scope of the assignment.

Notes: In order to make this assignment gradable, an autograding script may be used. This means you must follow the formatting EXACTLY for your program to output results that the grader will recognize as correct. This means your program should not output extra spaces after commas, extra spaces at the end of lines, or extra lines at the end of the file. Such extraneous characters will result in the loss of points.

Deliverables:
You should turn in your analysis document, all code, and a makefile to build your program into the web-handin. You should also submit a hard copy of your analysis to me. I should be able to type `make` to make your program.

This problem will be scored at follows:

Program correctness (as defined)	35%	(18 pts)
Quality of design/readability	20%	(10 pts)
In-line documentation/coding standard	15%	(7 pts)
Design and analysis document	25%	(12 pts)
Thoroughness of test cases	5%	(3 pts)