LCDB: Linked Computer Database

This assignment will modify and extend DCDB. All of the previous functionality of SCDB, CCDB and DCDB will be retained, unless explicitly removed or modified in this specification. There are two major extensions.

First, the DCDB computer data structure will be changed. LCDB will use a **double linked-list** of dynamically allocated nodes to store multiple computer objects. In order to receive full credit, the list must be fully encapsulated using a C++ class, and the list nodes must themselves be implemented using a node class.

Second, in order to make the program slightly more useful, you will add a histogram command. This will create a horizontal histogram of the manufacturer's average computer price currently stored in the list (see input/output file descriptions below). In addition, the program will provide the capability of dumping the list ordered on different fields. This will require slight changes to the input/output specifications, the addition of one new action and the modification of one existing action:

`histogram`

and

`dump <FieldSpecifier>`

LCDB will process command-line arguments in almost the same way as DCDB.

`LCDB <InitialComputerSystemDataFileName>`

or

`LCDB <ComputerSystemDatabaseFileName> <DatabaseActionsFileName>`

The change being that if only one command-line argument `<InitialComputerSystemDataFileName>` is present it will represent an initial computer data file that will be input and automatically saved as correspondingly named `<InitialComputerSystemDataFileName>.cdb` database file. (No actions file will be processed in this case.) The database file format is still unspecified. If two command-line arguments are present the first will represent a previously saved database file.

Input file descriptions:

The actions file will have almost the same syntax as for DCDB. Note that use of hard-coded file names will annoy the person evaluating your program, and you will be charged points for this shortcoming. Sample input files will be posted on the website soon.

Each line of the actions file will contain one of the commands described in the SCDB, CCDB and DCDB specifications, or one of the commands described below. As before, commands are case-sensitive and take a fixed number of arguments. It may be assumed that the command names will be valid and each command will include the correct number of arguments. Command arguments will be tab-delimited.

`histogram`

> A `histogram` command causes the current computer objects, stored in the linked-list database, to be traversed, the average price of each different manufacturers' computers to be determined and a proportional horizontal histogram of the average computer prices to be output to the dump file. The bar representing the computer prices will begin in column 11. It will be preceded by the manufacturer's name in columns 1-9, (truncated if necessary), and a colon in column 10. The maximum length for a bar will be 70 characters. The bar for the manufacturer with the greatest average computer price will be exactly 70 characters, with the other price bars output proportionally.

The characters representing the bar for a manufacturer will be the characters of the manufacturer's name itself, (see the output section below). For ease of histogram production, it may be assumed that no more than 10 different manufacturers will exist in the database at any given time.

`dump <FieldSpecifier>`

This causes the database list of computer records to be printed to the `dbase.txt` output file in ascending order by the specified field. As previously, each computer record output will be numbered starting at zero. However, for LCDB no maximum storage size for the database list will be output. The FieldSpecifier must be one of: `Model, Price`. These represent ordering upon the corresponding fields. If a `dump` instruction includes an invalid FieldSpecifier, the list will not be output. You are free to use whatever sort algorithm you wish.

Output description and sample:

As before, output data resulting from a `dump` command must be written to a file named `dbase.txt` — use of any other output file name **will** result in a large deduction. The format of `dump` output should be the same as previous programs, except that the `[XXX] END` output at the end of database dump will no longer be included. Sample output files will also be posted on the course website shortly.

Given the sample database file contents output dump below, the corresponding histogram command output would be:

Programmer: Dwight Barnette
Linked Computers Database

IDX	Model	Manufacturer	Price	Processor	Speed	RAM (MB)	Drive (MB)	Video (MB)	Cache (K)	CD	Sound	Case
000	688982U	IBM	1099	Pentium III	550	128	13500	16	512K	40X	Digital Aud	Mini
001	688997U	IBM	2499	Pentium III	600	256	9100	0	512K	40X	Digital Aud	Mini
002	689382U	IBM	1599	Pentium III	500	128	13500	16	512K	40X	Digital Aud	N/A
003	C4012	Cyber Power Inc.	989	Pentium III	1000	256	60000	64	N/A	16X	SoundBlaste	ATX
004	C4024	Cyber Power Inc.	1559	Pentium III	850	256	40000	32	256K	16X	AC97 3D WA	ATX
005	CC202	Cyber Power Inc.	2349	Pentium III	866	128	30000	32	N/A	52X	AC97 3D WA	ATX
006	D6720T	HP	699	Pentium III	450	64	6400	0	N/A	0X	N/A	N/A
007	D7969T	HP	999	Pentium III	500	64	8400	8	512K	32X	Digital Aud	Mini
008	MT-359AD	4explorermicro.com	717	Pentium III	1000	64	20000	8	512K	50X	3D Soundpro	ATX
009	MT-359AT	4explorermicro.com	1469	Pentium III	1000	64	60000	32	None	50X	3D Soundpro	ATX
010	MT-359AV	4explorermicro.com	1899	Pentium III	1000	128	60000	32	None	50X	3D Soundpro	ATX

```
                              Computer Price Histogram

IBM       :IBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMIBMI
Cyber Pow:Cyber Power Inc.Cyber Power Inc.Cyber Power Inc.Cyber Power Inc.Cy
HP        :HPHPHPHPHPHPHPHPHPHPHPHPHPHPHPHPHP
4explorer:4explorermicro.com4explorermicro.com4explorermicro.com4
```

Linked List:

You are **required** to use a linked list to store the computer system information. Your implementation of this list will be examined. In order to receive full credit, you must implement the nodes using a well-designed node class and the linked list itself must be encapsulated using a well-designed list class. In addition, the computer system data must be encapsulated so that the data they store is logically separated from the node pointers.

Note this mimics the behavior of a C++ STL (Standard Template Library) list object. You are specifically forbidden to use any C++ STL list objects in this program, or any other sort of predefined dynamic/static list type. You may base your implementation on the linked list implementations shown in the textbook, or those shown in the notes and lectures. (The lecture notes implementation is recommended, as the Deitel implementation node class is an instantiation of a template class. If you elect to base your implementation on the Deitel code you must modify it **not** to use templates.) You may not use linked list code from any other source. Advanced C++ OOP constructs and concepts, (e.g., inheritance, virtual

functions, templates, etc.), not covered in CS 1704 are also **not** to be used in this program. Violating these restrictions would remove major points of this assignment and will certainly result in a large deduction.

Programming Standards:

You'll be expected to observe good programming/documentation standards. All the requirements for documentation and coding given in the DCDB specification are still in effect. In addition:

Documentation:

- You must describe the purpose of each of your classes in a header comment that precedes the class declaration.
- You must document each data member and function member of your classes, both in the header file containing the class declaration and in the corresponding source file containing the implementation. The header file does not have to contain full documentation for each function, but the purpose of each function should be described there.

Coding:

- You must separate the interface of each of your classes from its implementation by placing each class declaration (interface) in its own header file and the implementation of that class in a corresponding source file. The name of the class should be used as the name of the header and source files.
- You must protect access to your data by making **all** data members of your classes private.
- When a node is removed from your linked list, you must dispose of it properly by using `delete`.
- When you discard your computer system database list to load an existing one from a file, you must `delete` every node in the old list so that your program does not waste memory.
- Memory leaks that might affect the execution or functionality of your program will be penalized. In fact, you should avoid memory leaks altogether.

Interim Design:

You will produce an interim design for LCDB, and represent that design in a modular structure chart. The structure chart must indicate your design plans for LCDB. It is expected that your final design will differ from the interim design. Nevertheless, your interim design should be relatively complete. If the interim design is incomplete or if the differences between your interim design and your final code are excessive, you will be penalized. That means that you should take the production of the interim design seriously, but **not** that you should avoid changes that would improve your final implementation of LCDB. You must submit this interim design, to the Curator System, no later than midnight **Wednesday, Nov. 14**. Submit a MS Word.doc file. Do **NOT** compress, (zip), the interim design submission! The file must contain no bit-map drawings, only vector-structured graphic drawings will be accepted. The file size must be no more than 300K.

Testing:

Obviously, you should be certain that your program produces correct output on all given sample data. However, verifying that your program produces correct results on a few test cases does not constitute a satisfactory testing regimen. You could make up and try additional input files as well; of course, you'll have to determine what the correct output would be.

Deliverables:

Your final project submission must include the following (and absolutely no other files):

- all source code (`*.cpp` and `*.h` files) comprising your project
- The MS Visual C++ project files (.dsp and .dsw). Do **NOT** submit the debug/release project subdirectories or an executable.exe file.
- a revised modular structure chart reflecting the final design of your project, at the time of submission; this must in the same MS Word format as the interim chart.

- one set of input files, named `AreaData.txt` and `Actions.txt`, and the corresponding final dump `dbase.txt`, and the corresponding saved database file(s)`.cdb`.
- a brief ASCII text readme file, named `readme.txt`, with any special execution instructions

Submissions will be archived, but not scored, by the Curator System. You will submit your project as an archive file in a zipped archive containing the items listed above. (The shareware program WinZip is very easy to use and is available from the Computing Services website: `http://www.ucs.vt.edu/`)

Note that omitting files from your archive is a classic error. Once you've created your project archive, copy the file to a new location, unzip it, attempt a build and then test the resulting executable. Submitting an incomplete copy of your project may delay its evaluation and **will** result in a substantial loss of points. In particular, if you omit a source file necessary to compile your program, you **will** be allowed to supply that file; however, we will then apply a **late penalty** corresponding to the date that you have provided a complete copy for evaluation. There will be **no exceptions** to this penalty. **Also note** that including unnecessary files is also a classic error. Visual C++ users: do **not** zip up the **debug** subdirectory!

Submitting your project archive:

You will submit your project archive to the Curator System, as described above. All submissions for `LCDB` must be made by midnight Monday Dec. 3rd. There will be **NO late submissions** accepted for `LCDB`. `LCDB` will be subjected to runtime testing by the TAs, who will also score your implementation for adherence to the specified programming standards. Demonstration time slot signup sheet will be made available in the CS lab. An announcement will be posted when they are available. Students will only be allowed to schedule and perform one demonstration. TA/student demonstration assignments will be posted on the course Web site. You will be allowed to make up to five submissions of `LCDB` to the Curator. Note well: **your last submission will be graded.** There are no exceptions to this policy! Changes made to code during a demonstration will be heavily penalized.

Pledge:

Each of your project submissions to the Curator System must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment of the `cpp` file containing `main()`:

```
// On my honor:
//
// - I have not discussed the C++ language code in my program with
//   anyone other than my instructor or the teaching assistants
//   assigned to this course.
//
// - I have not used C++ language code obtained from another student,
//   or any other unauthorized source, either modified or unmodified.
//
// - If any C++ language code or documentation used in my program
//   was obtained from another source, such as a text book or course
//   notes, that has been clearly noted with a proper citation in
//   the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
//   interfere with the normal operation of the Curator Server.
```

Failure to include this pledge in a submission is a violation of the Honor Code.