**CS 170 Computer Science 1: C++**

**Fall Semester 2008**

**Assignment 3**

Due Monday December 8th by 10:00 a.m.

Course weighting 25%

File name:

Last revised: Wednesday, 10 December 2008 at 05:13 AM

This assignment requires you to solve four programming problems, and to implement your solution in C++. You will be assessed by your final delivery. This is an individual assignment. No collaboration is permitted.

Problems:

1. Write a program PROBLEM1\_A.cpp that inputs **several** lines of text from the keyboard and prints an alphabetical listing of each word in the text and how many times it occurred. Ignore any case differences (e.g., “Cat” and “cat” are the same word). Use the qsort() function to sort the words. Terminate input by signaling end-of-file from the keyboard. As a minimum requirement, you may assume there will be no punctuation marks and that exactly one space will separate words.

[30 points]

Write a second version of this program, PROBLEM1\_B.cpp that will take its input from a sequential file rather than from the keyboard.

[5 points]

|  |  |  |  |
| --- | --- | --- | --- |
| Record # | Tool name | Quantity | Cost |
| 3 | Electric Sander | 7 | 57.98 |
| 17 | Hammer | 76 | 11.99 |
| 24 | Jig Saw | 21 | 11.00 |
| 39 | Lawn Mover | 3 | 79.50 |
| 56 | Power Saw | 18 | 99.99 |
| 68 | Screw-driver | 106 | 6.99 |
| 77 | Sledge Hammer | 11 | 21.50 |
| 83 | Wrench | 34 | 7.50 |

1. You are the owner of a hardware store and need to keep inventory that can tell you what different tools you have, how many of each you have on hand and the cost of each one. Write a program that initializes the random-access file “hardware.dat” to one hundred empty records, lets you input the data concerning each tool, enables you to list all your tools, lets you delete a record for a tool that you no longer have and lets you update *any* information in the file. The tool identification number should be the record number. Use the table on the right to start your file.

[30 points]

1. a) A gym club keeps information on the name, height, date of joining, and fitness score out of 100 for each member. Define a gymMember structure to hold the data. Overload the stream insertion operator << so that it can be used with this structure. Place all data type definitions and the operator declaration in a header file gymMember.h. Place the operator definition in a file gymMember.cpp. Write a main program Problem3a.cpp that will create an array of gymMember variables. You can populate the array by initialization, by reading from the keyboard or by reading from a file. Then your program should print the array on the screen in a well formatted table.

[15 points]

b) Write another main program Problem3b.cpp that will create a dynamic array of gymMember variables. You can populate the array by initialization, by reading from the keyboard or by reading from a file. Then your program should print the array on the screen in a well formatted table.

[5 points]

1. This question is optional, for extra credit. It asks you to solve the same problem as in question 3, but using classes rather than structures.   
     
   A gym club keeps information on the name, height, date of joining, and fitness score out of 100 for each member. Define a gymMember class to hold the data. Overload the stream insertion operator << so that it can be used with this class. Provide an argument-less constructor, and a constructor with arguments for name, height, date of joining, and fitness score. Make all data members private and provide appropriate accessor and mutator member functions. Place all data type definitions and the class declaration in a header file gymMember.h. Place the class implementation, including the << operator implementation in a file gymMember.cpp. Write a main program Problem4.cpp that will create a dynamic array of gymMember objects. Populate all array elements - at least one array element using constructor arguments, and at least one array element using mutator functions. Then your program should print the array on the screen in a well formatted table.

[15 points]

1. **Development Requirements**
2. **Constraints**. Coding must use C++ streams for input and output and generate a DOS executable.
3. **Dependencies**. You are encouraged to use global constants, but your program must not declare any global variables, whether of simple data types, structures, arrays, or file streams.
4. **Standards**. Your programs must meet the programming standards for this course (attached to Assignment 1).

Delivery

All your source code files (.cpp and .h) and any data files (if applicable) must be placed on a floppy disk in a directory \*YourName*\Ass3. (for example, a: \PeterThomas\Ass3). You should name the source files containing your main programs PROBLEM1.CPP, PROBLEM2.CPP and PROBLEM3.CPP.  
NOTE: If for any of the problems these files are missing, (or incorrectly named or in the wrong directory) or the program will not compile and link correctly, I will not be able to grade your work for this problem. Please double-check this.

You should also submit in a folder with a standard cover sheet (file CoverSheet.doc in the course directory) with all fields filled in correctly, the following in this order:

1. Grading guide (supplied) with sections 1, 2 and 3 completed to show what you have done
2. For each problem, in this order:

* A structure chart showing the modular structure of your program
* Optionally, structure diagrams or pseudocode showing the algorithms design

1. A test plan showing:
2. Check-points, with a clearly indicated result (Y or N)
3. Test data and results in a table with three columns:
   * test input (printed)
   * expected results (printed)
   * the actual results of your testing written in by hand
4. A brief analysis of any known errors, which the program still produces

* A printed copy of your source code. Use a fixed-width font to preserve indentation. The first page should list files used by the program (source, header, library and data, if applicable).

1. Sample copies of any printed reports produced by your program (if applicable).

.

Grading Schedule

The assignment will be graded on an A to F scale of grades.

Work which barely meets the minimum requirements and either has problems with usability or readability or does not meet the programming standards will be graded D- to C. Work which shows a useable solution with all the minimum requirements and meets the programming standards will be graded C+ to B+. Work which in addition demonstrates initiative in design and implementation as evidenced by superior user interaction, additional functionality, robustness and reliability will be graded A- to A.

Grading Notes:

1. To get credit for a program feature, it must be coded, tested and documented correctly according to the given standards and be working in all respects.

## A feature that is either not shown on the test plan as tested or does not work correctly will be given no credit. Penalty: points will be deducted for a faulty feature that is shown on the test plan as working.

1. The grading criteria include functionality, non-functional requirements, documentation and development requirements as indicated on the grading guide. Pay special attention to function/method design.

Assignment 3 Grading Guide Name

1. Minimum Requirements *(write your full name)*

*(check the boxes below)*

* Problem 1
* Problem 2
* Problem 3
* Problem 4

2. Extensions

*(list extensions completed)*

Problem 1

* I added a system that lists the words according to the first letter of the word.
* A heading is printed for each letter that begins any word in the text.
* A program title and description was added.

Problem 2

* A program title and description was added.
* Users can delete all records at once
* Records that do not contain data, are not displayed.
* Multiple menus to enhance the users experience where added. Such as: the main menu, the record edit menu, and the edit record data menu.
* The program creates a file hardware.dat and initializes it to the preset records, if hardware.dat does not exist.

Problem 3

* A program title and description are outputted.

Problem 4

* A program title and description are outputted.

3. Documentation submitted

*(check boxes of items attached)*

* Structure charts
* Pseudocode/structure diagrams
* Test plans
* Source code
* Printed reports

4. Non-functional requirements

- screen layout

- user interaction

- data validation

5. Development Requirements

- program organization

- dependencies

- data structures

- function/method design

- program layout

- internal documentation

Grade \_\_\_\_