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

**Spring Semester 2022**

**Assignment 3**

Due date: See Syllabus

Course weighting 25%

File name: AS1\_98S1.DOC

Last revised: Saturday, 30 April 2022 at 10:06 AM

This assignment requires you to solve three 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. a) Write a program Problem1\_A.cpp that inputs a sequence of several lines of text from the keyboard as C-strings and prints an alphabetical listing of each word in the text and how many times each word occurred without regard for character case (*e.g.*, “Cat” and “cat” are the same word). Ignore empty lines and terminate input by signaling end-of-file from the keyboard.

You may assume there will be no punctuation marks and that exactly one space separates any two words in an input line. You may also assume, but clearly state, the maximum number of characters in a line and the maximum number of input lines.

You may **not** use vectors or the library class string**.**

HINT: Keep a pointer to each word in an array of character pointers or keep the words in a 2-dimensional array of characters (an array of C-strings). Use the qsort() library function to sort the pointer array or the array of C-strings, and then count the number of occurrences of each word.

[30 points]

b) 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]

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.
2. Create a class or a structure to store the data for a tool consisting of a tool number, name, quantity and cost (per unit)

|  |  |  |  |
| --- | --- | --- | --- |
| Tool # | 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. Write a program that maintains a binary random-access file hardware.dat in which each tool is stored in a position given by its tool number. Store a full record, including the tool number.   
   Provide a menu allowing the user to do the following:

* initialize the file hardware.dat to 100 empty records.
* populate the file hardware.dat by reading from a text file
* input data concerning one or more tools fro the keyboard
* list all tools (in a table as in the example on the right; do not list the empty records in between)
* delete a record for a tool
* update *any* information for a tool.

Design the values of the empty record fields so that your program can distinguish an empty record from a record with data for a tool.

Carefully design the user interaction to allow user to perform each of these tasks. Pay special attention to the clarity of the prompts and messages to the user.

For output and input overload the operators << to write on any ostream object and >> to read from any istream object. Use the tool identification number as the record number in the random-access file. Use the table above to start your binary file by reading from a text file.

[40 points]

**Development Requirements**

1. **Constraints**. Coding must use C++ streams for input and output and generate a DOS executable.
2. **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.
3. **Standards**. Your programs must meet the programming standards for this course (attached to Assignment 1).

Delivery:

No paper documentation. All delivery will be on the X: drive, in the directory X:\Dropoff\CS\ganchevg\CS170\Assignment3 in a subdirectory YourName.

* Make subdirectories Code and Documentation.
* In the **Code** directory place all your source code files (.cpp and .h) and any data files (if applicable).
  + Name the source files containing your main programs Problem1\_A.cpp, Proble1\_B.cpp, etc.  
    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.
* In the Documentation directory

1. Place the Grading Sheet (supplied) with sections 1, 2 and 3 completed to show what you have done
2. Make subdirectories Problem1, Problem2, etc. In each subdirectory place

* A structure chart or class diagram showing the modular structure of the corresponding program
* Optionally, structure diagrams or pseudocode showing the algorithm design
* A test report showing:

1. Check-points, with a clearly indicated result (Y or N, typed)
2. Test data and results in a table with three columns:
   * test input (typed)
   * expected results (typed)
   * the actual results of your testing (typed)
3. A brief analysis of any known errors, which the program still produces or a statement that there are no known errors

* Optionally, 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.
2. A feature that is either not shown on the test plan as tested or does not work correctly will be given no credit.
3. Penalty: points will be deducted for a faulty feature that is shown on the test plan as working and for not satisfied non-functional requirements (development requirements, delivery requirements, programming standards).

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

Assignment 3 Grading Sheet Name

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

*(check the boxes below)*

* Problem 1a
* Problem 1 b
* Problem 2 a
* Problem 2 b

2. Extensions

*(list extensions completed)*

3. Documentation submitted

*(check boxes of items attached)*

* Structure charts / class diagrams
* Pseudocode/structure diagrams (optional)
* Test reports
* Source code
* Printed reports (optional)

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 \_\_\_\_