

Parsing Delimited Input

This assignment is primarily about parsing character and numeric input data that is delimited; i.e., the input consists of logical tokens that are separated by specified characters.

The input file contains raw data from a (hypothetical) roommate preferences survey. In particular, an individual may indicate a preference in one or more of nine categories (named: Wild, Pious, SportsFan, Studious, Smoker, NightOwl, Vegan, Geek, Greek). If a preference is indicated, the individual must provide a numerical rating in the range 1 to 5, and also provide an indication of the importance of the category as a numerical rating in the range 1 to 10.

You must design and implement a C++ program that will parse an input file of the form described below and produce an output file of the specified form. This assignment will be graded automatically by the Curator System (which will be used to collect most of your work in this course). Read the *Student Guide to the Curator System* at the Curator website (<http://ei.cs.vt.edu/~eags/Curator.html>) for submission information.

Note: this program must be implemented as a single C++ source file.

Input file description and sample:

Your program **must** read its input from a file named `parse.in` — use of another input file name will result in a score of zero. The first three lines of the input file are a text header and should be ignored. Each remaining line of the input file will be a survey response record, conforming to the following format:

```
<ID><tab><Name> { <tab><Response Record> } + <newline>
```

where

```
<ID>          xxx-xx-xxxx, where each x is a digit from 0-9
<Name>       a sequence of 1 to 20 characters, possibly containing punctuation and spaces (but not tabs)
<tab>       a single tab character
<newline>    a single newline character
```

and a `<Response Record>` has the form `<Category>:<Importance>:<Rating>`, and

```
<Category>   a sequence of 1 to 10 characters with no internal whitespace from the list given above
<Importance> a positive integer in the range 1-10
<Rating>     a positive integer in the range 1-5
```

Note that the fields of a response record are delimited by colons (:).

You may assume that the input file will contain tabs and colons as described. Do not assume that all the category labels that occur in the input file will be from the list given above (case sensitive); if they are not, your program should ignore that response record. You also may not assume that the integer values will be in the specified ranges; if your program encounters an out-of-range integer value, the program should ignore that response record. Each individual will always have at least one valid response record.

Roommate Preferences Survey Data						
Sequence Number: 4390120						
000-00-0001	Hokie, Joe	Pious:1:1	Sports:5:5	Wild:5:1	Smoker:5:1	Geek:10:5
000-00-0002	Hoo, Haskell	Wild:10:5	Smoker:5:5	Greek:10:5	Geek:10:1	SportsFan:5:5

There will be at least one survey response record; the maximum number of input lines is unspecified. Your program must be written so that it will detect when it's out of input and terminate correctly.

Output description and sample:

A sample output file produced from the sample input file above is shown below. The first three lines of your output should identify you by name, identify the assignment you're submitting, and your section, as shown. The fourth line should be blank.

Next your output file will contain a sequence of tables, one for each survey response record given in the input file. Delimit each table by a line of 25 plus signs, as shown. Each table should list the respondent's name, on a line by itself. This should be followed by a listing of the categories for which the respondent provided information, listed in alphabetical order, and for each category, the product of the importance and rating given for that category.

Your program must write its output data to a file named `parse.out` — use of any other output file name will result in a score of zero.

```

Programmer:  Bill McQuain
CS 2704 Homework Assignment 1
2:00 TTh Section

+++++
Hokie, Joe
    Geek          50
    Pious         1
    Smoker        5
    Wild          5
+++++
+++++
Hoo, Haskell
    Geek          10
    Greek         50
    Smoker        25
    SportsFan     25
    Wild          50
+++++

```

You are not required to use this exact horizontal spacing, but your output must satisfy the following requirements:

- You must arrange your output in neatly aligned columns, with a label identifying the contents of each column. Use spaces, not tabs to align your output. Note that while the Curator doesn't deduct points for horizontal alignment, the ability to properly align output data in a tabular format will be required for some of the programming projects in this course.
- You must print a newline at the end of each line, including the line of pluses marking the end of the table.

Grading:

You will submit this assignment to the Curator System (read the *Student Guide* mentioned above), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. If you do not get a perfect score, analyze the problem carefully and test your fix before submitting again. The highest score you achieve will be counted.

Programming Standards:

While we won't be evaluating your source code on this assignment for internal documentation or programming style, you should still observe good practice. Here's a brief description of some of the things we'd normally expect:

Documentation:

- You must include the honor pledge (below) in your program header comment.
- Your header comment must describe what your program does.
- You must include a comment explaining the purpose of every variable or named constant you use in your program.
- You must use meaningful identifier names that suggest the meaning or purpose of the constant, variable, function, etc.
- Precede every major block of your code with a comment explaining its purpose.
- Precede every function you write with a header comment. This should explain in one sentence what the function does, then describe the logical purpose of each parameter (if any), describe the return value (if any), and state reasonable pre- and post-conditions.
- You must use indentation and blank lines to make control structures like loops and if-else statements more readable.

Coding:

- Use named constants instead of variables where appropriate.
- Use an struct or class variables to organize related heterogeneous data.
- Make good use of user-defined functions in your design and implementation. The body of `main()` should contain no more than 20 executable statements and the bodies of the other functions you write should each contain no more than 40 executable statements. An executable statement is any statement other than a constant or variable declaration, function prototype or comment. Blank lines do not count.
- The definition of `main()` must be the first function definition in your source file. You may use file-scoped function prototypes and you may use file-scoped constants. You may also make the `typedef` statement for your structured variable type file-scoped (in fact you must do this).
- You may not use file-scoped variables of any kind.
- Function parameters should be passed appropriately. Use pass-by-reference only when the called function needs to modify the parameter. Pass any large objects, such as array parameters, by constant reference (using `const`) when pass-by-reference is not needed.
- Use string objects to store character data, not `char` arrays.

Pledge:

Each of your program submissions 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 for your program:

```
// 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 Automated Grader.
```

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