

Reading to Input Failure, Decisions, Functions

This programming assignment uses many of the ideas presented in sections 3 through 8 of the course notes, so you are advised to read those notes carefully as well as the following program specification. This program will be worth 12% of your final grade and will be broken into two parts – 70% for correctness (EAGS score) and 30% for style.

The Program Specification:	Parcel Post
-----------------------------------	--------------------

The shipping company Dead-Ex (formed by former Grateful Dead groupies) will ship packages subject to the following constraints:

- The weight of a package must not exceed a specified maximum weight.
- The length of the package must not exceed a specified maximum length. The length of a package is defined to be the length of its longest dimension.
- The size of the package must not exceed a specified maximum size. The size of a package is defined to be the length of the package plus the girth of the package. The girth of a package is the circumference of the package around its two smallest sides and can be calculated using the formula:

$$\text{Girth} = 2 * (\text{Dimension1} + \text{Dimension2} + \text{Dimension3} - \text{Length})$$

You need to determine whether certain packages meet these criteria. For example, suppose the maximum weight allowed is 40, the maximum length is 36, and the maximum size is 72. Consider a package with weight 35, and dimensions 12, 16 and 35. The length of the package is 35. The girth of the package is $2*(12 + 16 + 35 - 35) = 2*28 = 56$. So the size of the package is $56 + 35 = 91$, which is too large.

Postal Rates

The cost to ship a package is dependant upon its weight and size. The rates are as follows:

- For packages that weigh less than 10 pounds, the rate is a flat \$1.00
- For packages between 10 and 19 pounds, the rate is \$0.15 per pound
- For packages between 20 and 29 pounds, the rate is \$0.20 per pound
- For packages that weigh 30 pounds or more, the rate is \$0.25 per pound

Additionally, if the package size is greater than the maximum standard size, an additional 10% surcharge will be added to the cost of shipping.

Input file descriptions and samples:

Your program **must** read the postal rates from a file named `parcel.data` — use of other input file names will generally result in a score of zero.

In the file, a newline character will terminate each input line. You may assume that all of the input values will be logically correct (no negative or missing values, for instance), that they will be in the specified ranges, and that they will be given in the specified order. Unless indicated otherwise, you may also assume that all data values will be in the range 1 to 999.

Input File

The first three lines of this input file are preceded by an appropriate label and specify:

- A positive integer representing the maximum standard size for a package (above this size, there is a 10% surcharge to the cost of shipping)
- A positive integer representing the maximum allowable size of a package
- A positive integer representing the maximum allowable weight of a package
- A positive integer representing the maximum allowable length of a package

The fourth line is blank. The fifth line contains column labels, which should be ignored. The remaining lines of the input file will each contain:

- A positive integer representing the package number in the range of 1000 to 9999
- A positive integer representing the weight of the package
- A positive integer representing the first dimension of the package
- A positive integer representing the second dimension of the package
- A positive integer representing the third dimension of the package

There is no guaranteed limit on the number of transactions.

Here is an example input file:

Maximum Standard Size:	60			
Maximum Size:	72			
Maximum Weight:	50			
Maximum Length:	36			
Number	Weight	Dim1	Dim2	Dim3
1001	50	36	1	1
1002	35	12	16	35
1003	23	7	13	20
1004	60	1	1	1
1005	1	6	6	2
1006	18	10	10	12
1007	51	27	49	15
1008	49	31	10	10
1009	41	9	36	9
1010	52	9	10	36
1011	39	38	8	9
1012	6	3	17	32

What to Calculate:

Your program must output a table showing all of the transactions read from the input file. The table format is shown in the output section below. In particular, for each package, you must determine if it is too big, too heavy, too long, or any combination of these and indicate your conclusions as shown in the example output file below. If the package can be mailed, you must determine the cost of mailing it.

After processing all the packages, you should print out the total number of packages that were rejected for each reason, as well as the number of packages that were process. Remember that a package can be rejected for more than one reason.

Output description and sample:

Your program must write its output data to a file named `package.out` — use of any other output file name **will** result in a score of zero. A sample output file produced from the sample input files above is shown below:

```

Programmer:  Chris Knestrick
Parcel Post

Number      Weight    Cost    Too Big    Too Heavy    Too Long
-----
1001         50     12.50
1002         35     *****      x
1003         23      4.60
1004         60     *****                x
1005          1      1.00
1006         18      2.70
1007         51     *****      x          x          x
1008         49     13.48
1009         41     11.28
1010         52     *****      x          x
1011         39     *****                x
1012          6      1.10
-----
Totals:                3          3          2

Packages shipped:    7

```

As usual, the first line of your output should identify you by name, as shown. The second line should include the title “Parcel Post” only. The third line should be blank. The fourth and fifth lines should contain the specified column labels and a row of delimiters to mark the top of the table.

Next your output file will contain a table, with a line of output for each package. Each line should contain the package number and the package weight. If none of the criteria for a package are exceeded, print the cost of shipping the package; otherwise print “*****” in the Cost column and print a lower-case ‘x’ in the appropriate column(s) to indicate which of the criteria are exceeded. Notice that more than one column can be marked.

After the last line of the table, print a line of delimiters and then display the number of packages that exceeded each of the criteria, as well as the total number successfully processed.

You are not required to use the exact horizontal spacing shown in the example above, but your output must satisfy the following requirements:

- You must use the specified header and column labels, and print a row of delimiters before and after the table body, as shown.
- You must arrange your output in neatly aligned columns. Use spaces and `setw()`, not tabs to align your output.
- You must use the same ordering of the columns as shown here, and print the Cost with precision two.

Programming Standards:

You'll be expected to observe good programming/documentation standards. All the discussions in class about formatting, structure, and commenting your code should be followed.

Documentation:

- You must include the honor pledge 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), the name of the calling function(s), the name of any functions that are called, 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 `double` variables for all cost values, and `int` variables for all other input data.
- You must make good use of user-defined functions in your design and implementation. To encourage this, the body of `main()` must contain no more than 20 executable statements and the bodies of the other functions you write must 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.
- You must write at least four functions, besides `main()`. For reference, my solution uses seven. For each function that is missing, 25 points will be deducted from your **final** grade, not just style. This means that if your program receives 100 from the EAGS but you didn't include any functions, your final grade will be a 0.
- You must write at least one `void` function, at least one `bool` function, at least one function of type `int`, and at least one function of type `double`.
- 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 not use any global variables – failure to follow this guideline will result in major deductions.**
- Function parameters should be passed appropriately. Use pass-by-reference only when the called function needs to modify the parameter.

Your submission that receives the highest score will be graded for adherence to these requirements, whether it is your last submission or not. If two or more of your submissions are tied for highest, the earliest of those will be graded. Therefore: implement and comment your C++ source code with these requirements in mind from the beginning rather than planning to clean up and add comments later.

Testing:

Obviously, you should be certain that your program produces the output given above when you use the given input file. However, verifying that your program produces correct results on a single test case does not constitute a satisfactory testing regimen.

At minimum, you should test your program on **all** the posted input/output examples given along with this specification. The same program that will be used to test your solution generated those input/output examples. You could make up and try additional input files as well; of course, you'll have to determine by hand what the correct output would be.

Pledge:

Each of your submissions to the NAG 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.