

Arrays, 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.

The Program Specification:

Parcel Post

The shipping company Delivering On-time With No Surprises (DOWNS) will ship packages subject to the following constraints:

- The weight of the 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 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

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

For example, a package whose dimensions are 10, 12 and 15 has a length of 15, a girth of 44, and a size of 59.

You need to determine whether certain packages meet these criteria.

Input file descriptions and samples:

This program requires the use of two input files. One contains the postal rates and the other contains the transactions. Your program **must** read the postal rates from a file named **parcel.data** and the transactions from a file named **transact.data** — use of other input file names will generally result in a score of zero.

Postal Rates Input File

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

- The maximum allowable weight of a package
- The maximum allowable length of a package
- The maximum allowable size of a package

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

- A positive integer representing a possible package weight
- The cost of shipping a package of the indicated weight

Use two one-dimensional arrays to store this table. The cost of shipping a package can then be determined by searching the Weight array and using the corresponding element in the Cost array. If the weight of a package falls between two weights in the table, the higher weight should be used to calculate the cost of shipping the package.

Each input line will be terminated by a newline character. You may assume that all of the input values will be logically correct (no negative or missing values, for instance), and that they will be given in the specified order. You may also assume that there will be no more than 25 (weight, cost) pairs.

Here is an example input file:

```
Maximum Weight: 50
Maximum Length: 36
Maximum Size: 72
```

Weight	Cost
2	1.00
4	1.75
6	2.50
8	3.25
10	4.00
12	4.50
14	5.00
16	5.50
18	6.00
20	6.50
22	7.00
24	7.50
26	8.00
28	8.50
30	9.00
35	9.83
40	10.66
45	11.50
50	12.33

Transactions Input File

The first line of this input file contains column labels, which should be ignored.

The remaining lines of the input file will each contain:

- A package number
- The weight of the package
- The first dimension of the package
- The second dimension of the package
- The third dimension of the package

Each input line will be terminated by a newline character. You may assume that all of the input values will be logically correct (no negative or missing values, for instance), and that they will be given in the specified order.

Here is an example input file:

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.

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:  Donald Allison
Parcel Post

Number      Weight    Cost    Too Big    Too Heavy    Too Long
-----
1001         50     12.33
1002         35     *****      x
1003         23      7.50
1004         60     *****           x
1005          1      1.00
1006         18      6.00
1007         51     *****      x           x      x
1008         49     12.33
1009         41     11.50
1010         52     *****      x           x
1011         39     *****           x
1012          6      2.50
-----

Packages processed:  12
Packages rejected:   5

```

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. The table should be terminated by a row of delimiters.

After the table, your output file will display the total number of packages processed and the number of packages that exceeded any of the criteria.

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, 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:

- 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, explaining in one sentence what the function does.
- 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.
- Write and use at least one user-defined function, not counting **main()**. Provide a valid C++ prototype for each function you write, excluding **main()** of course. The function **main()** must be the first function defined in your source code file.

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.