

Decisions and Iteration:**Simple Invoice**

Your solution to this assignment will read a simple order form and produce a customer invoice for that order. This will require reading a relatively simple input file, making decisions about the correctness of the input data, doing simple calculations, and writing a simple output file.

Input specification and sample input data:

The input file for this program is named "Order.txt". A sample input file is given below. The input file begins with four lines that list the name and address of the customer. The address will always consist of three lines, the last of which will contain three tab-separated items: city name, state and zip code. You should treat the customer name and the first two address lines as strings, and you should treat each "token" on the third address line as a separate string. A blank line will follow the third address line.

The next begins with a label which will end with a colon (':'), followed by the number of different items that will be listed in the following table. The next line is blank.

The table will begin with a line of labels, and then a line separating the labels from the order data. After that, there will be lines of data specifying the items which are being ordered. You may assume that the specified number of items will always be listed.

Each order will specify zero or more items, so your program must be designed to handle an arbitrary number of items. Don't forget to handle the case where there are no items. Each of these item order lines will have the following format:

```
<item description><tab><poss. spaces><number of units><spaces><unit price><\n>
```

The item description is just a string, and will never be more than 40 characters long (useful information for formatting your output). There will be a tab character immediately after the description. There will be zero or more spaces following the tab. The number of units must be a positive integer, and will be followed by one or more spaces. The unit price must be a positive decimal number, with two digits following the decimal point. There will be a newline character immediately after the unit price. There may (or may not) be irrelevant data after the last item order line; if so it should not be processed.

```
Hannibal Lecter
Suite B2
Chesapeake Institute
Baltimore      MD      21230

Number of items:  12

Item                                     Units      Price
-----
Chicken thighs, boneless 1 lb           10         2.99
Calimari, rings 16 oz                   -4         5.99
Chicken Wings, 10 oz                    7          2.99
Honey Wheat Bread, 16 oz loaf            5          2.45
Cheddar Singles 12 oz                    0          3.09
Hash Browns, 32 oz                       4          2.29
Homestyle Waffles, 16                    2          2.99
Pure Goat Milk Cheese, 1 lb              8         16.03
White Bread, 16 oz loaf                   1          2.19
Buttermilk Waffles, 16                    5          2.99
Red Kidney Bean                           9          0.79
White Bread, 20 oz loaf                   8          2.55

13:52:43 2/23/2007
FBX 3210 **#
```

The item descriptions are strings of characters, which may contain spaces but not tabs. Note that the appearance of tabs is somewhat unreliable, and so the perfect alignment shown above will not always be achieved in the input file. However, that won't bother your implementation.

It is possible that the customer will specify a number of units that is not positive, or specify a price that is not positive. If that happens, your program must ignore that item — that is, you won't include any of the data for that line in your final report.

What must be calculated:

For each properly ordered item, the program must calculate the total cost of the specified number of units of that item. The program must also calculate the total cost of all the ordered items.

Some orders may qualify for a discount. Specifically, if an order totals more and \$100.00 but less than \$250.00, then the customer will receive a discount of 5% of the amount of the order in excess of \$100.00. If an order totals \$250.00 or more, then the customer receives a 10% discount for the amount of the order in excess of \$250.00, plus a 5% discount for the amount of the order between \$100.00 and \$250.00. For example, if an order totaled \$275.00 then the customer would receive a discount of \$10.00 (5% of \$150.00 plus 10% of \$25.00). The total discount is truncated to an integer (number of cents) if it doesn't come out to an exact integer. The program must determine what discount, if any, the customer should receive, and then calculate the actual total due.

Output specification and sample output data:

The output file must be named "Invoice.txt". An output file, which corresponds to the input data given above, is shown below. The first four lines give the shipping address for the order, as given in the order file. Note the formatting that is used for the last address line; that's why you must read the city name, state and zip code separately. Remember, you must use exactly the same labels shown here. The next line is blank.

The invoice begins with a line of labels and then a line of delimiters separating the labels from the table body. The table body contains one line of output for each correctly ordered item, displaying the item description, units ordered and total price for that many units of that item. The last item data line is followed by another line of delimiters.

The next line shows the total cost of the order, before any discount. The next line shows the discount (0.00 if none), and the next line shows the grand total the customer must pay. The final line shows the total number of units to be shipped.

Ship to:		
Name:	Hannibal Lecter	
Address:	Suite B2	
	Chesapeake Institute	
	Baltimore, MD 21230	
Item	Units	Price

Chicken thighs, boneless 1 lb	10	29.90
Chicken Wings, 10 oz	7	20.93
Honey Wheat Bread, 16 oz loaf	5	12.25
Hash Browns, 32 oz	4	9.16
Homestyle Waffles, 16	2	5.98
Pure Goat Milk Cheese, 1 lb	8	128.24
White Bread, 16 oz loaf	1	2.19
Buttermilk Waffles, 16	5	14.95
Red Kidney Bean	9	7.11
White Bread, 20 oz loaf	8	20.40

	subtotal	251.11
	discount	7.61
	total	243.50
	#units	59

If you have read the *Student Guide to the Curator*, you already know that all the fixed text must be precisely as shown in the sample output. The output should be aligned for easy readability. Additional samples of input and correct output will be available on the course website.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), 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 ten submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and other pertinent information, such as the link to the proper submit page, can be found at:

<http://www.cs.vt.edu/curator/>

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 Curator System.
//
// <Student Name>
```

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