21. Convert the following `do. . .while` loop into a `for` loop and a `while` loop. Did the logic change? If so, explain.

```
int counter = 100;
do
{
        WriteLine(counter);
        counter--;
}
while (counter > 0);
```

22. Write a `for` loop to display every third number beginning with 10 and continuing through 100.

23. Write a sentinel-controlled `while` loop that allows any number of temperatures to be entered. The average temperature should be calculated and displayed.

24. Create a loop body that generates random numbers between 25 and 75. Write a state-controlled loop that adds all these randomly generated numbers until a value larger than 60 is generated. When the loop stops, display the number of acceptable generated values and the sum of those values.

25. Desk run or trace the following code segment, showing every value that goes into each variable.

```
for (i = 0; i < 3; i++)
     for (j = 4; j > 0; j--)
          WriteLine ("{0}\t{1}", i, j);
```

PROGRAMMING EXERCISES

1. Create an application that contains a loop to be used for input validation. Valid entries are positive integers less than 100. Test your program with values both less than and greater than the acceptable range as well as non-numeric data. When the user is finished inputting data, display the number of valid and invalid entries entered. For invalid values, identify how many of those values were outside the range and the number of non-numeric invalid values entered.

2. Write an application that will enable a vendor to see what earnings he can expect to make based on what percentage he marks up an item. Allow the user to input the wholesale item price. In a tabular form, show the retail price of the item marked up at 5%, 6%, 7%, 8%, 9%, and 10%.

3. Write a program that generates 1000 random numbers between 0 and 100,000. Display the number of odd values generated as well as the smallest and the largest of values. Output should be displayed in a Windows message box.

4. Write a program to allow multiple sets of scores to be averaged. Valid entries must be numeric and in the range of 0 through 100. Calculate the average of the scores entered. Allow any number of scores to be entered per data set but assume that there will be at least one score entered. Use a sentinel-controlled loop variable to terminate the loop. After values are entered and the average is calculated, test the average to determine whether an A, B, C, D, or F should be recorded. The scoring rubric is as follows:

   A→90–100; B→80–89; C→70–79; D→60–69; F < 60.

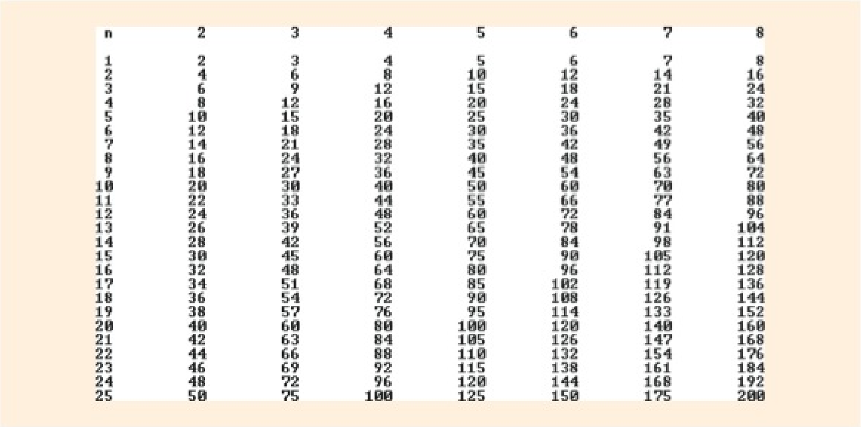
5. Create an application that can be used to calculate the total amount due for purchases. Allow any number of items to be entered. Determine the total due including sales tax and shipping. Sales tax of 7.75% is charged against the total purchases. Shipping charges can be determined based on the number of items purchased. Use the following chart to determine the shipping charge.

fewer than 3 items	$3.50
3 to 6 items	$5.00
7 to 10 items	$7.00
11 to 15 items	$9.00
more than 15 items	$10.00

   Display an itemized summary containing the total purchase charge, number of items purchased, sales tax amount, shipping charge, and grand total.

6. Write a program that allows the user to input any number of hexadecimal characters. Sum the values and display the sum as a hexadecimal value. Treat each single inputted character as a separate value. Within the loop, convert each character entered into its decimal equivalent. Display the original hex value and the corresponding decimal value. For example, if the user inputs F, 15 would be displayed as the decimal equivalent. Create a state-controlled loop structure. After all values are entered, display the sum of values entered in both hexadecimal and decimal notation.

7. Write a program that produces a multiplication table with 25 rows of computations. Allow the user to input the first and last base values for the multiplication table. Display a column in the table beginning with the first base inputted value. The last column should be the ending base value entered. The first row should be for 1 times the beginning base, 1 times the (beginning base value + 1), through 1 times the ending base value. The last row should be for 25 times the beginning base, 25 times the (beginning base value + 1), through 25 times the ending base value. Base values can range from 2 to 8. Display an error message if an invalid base is entered. Display an aesthetically formatted multiplication table. An example of output produced when 2 and 8 are entered appears in Figure 6-22.

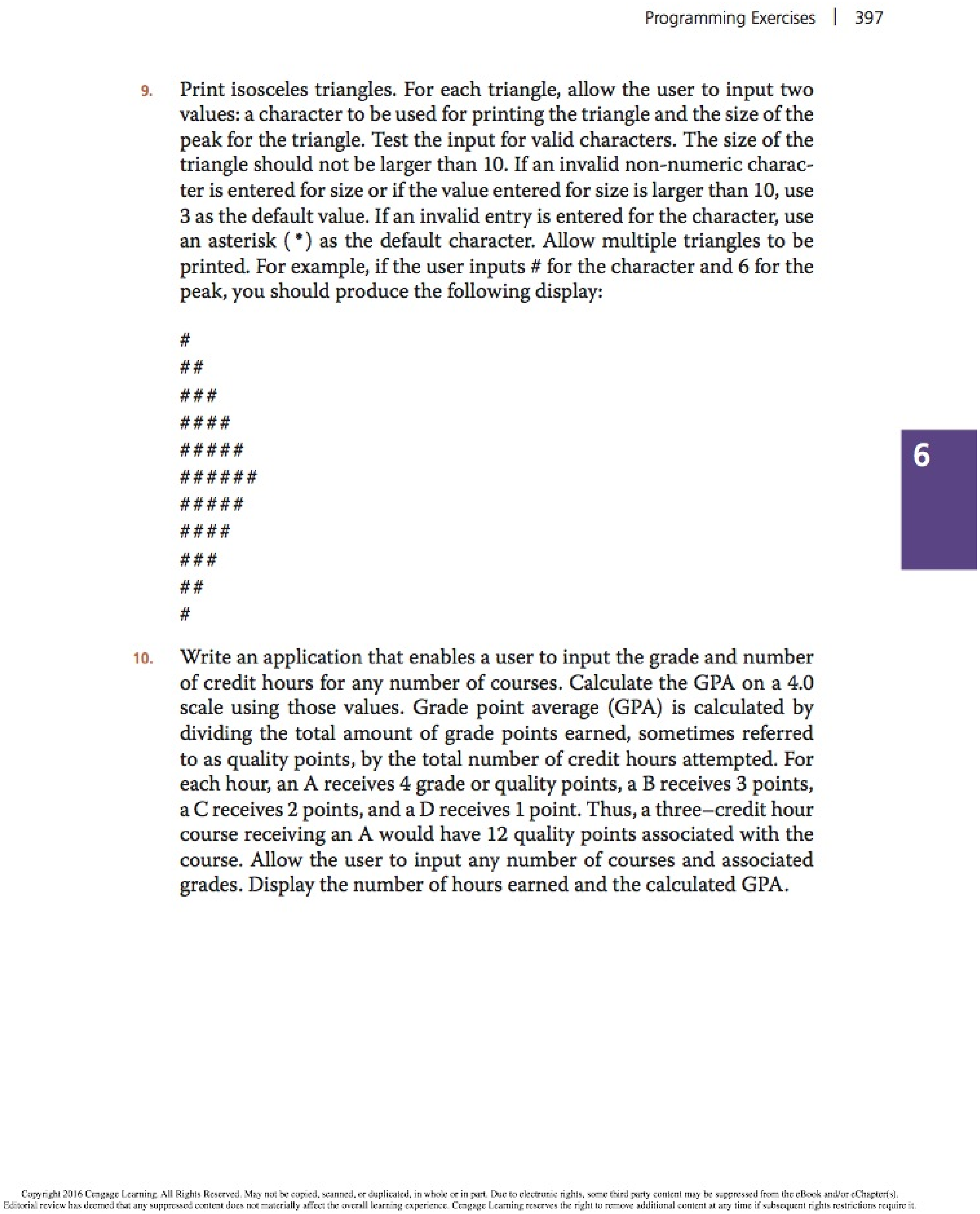


```
n          2          3          4          5          6          7          8

1          2          3          4          5          6          7          8
2          4          6          8         10         12         14         16
3          6          9         12         15         18         21         24
4          8         12         16         20         24         28         32
5         10         15         20         25         30         35         40
6         12         18         24         30         36         42         48
7         14         21         28         35         42         49         56
8         16         24         32         40         48         56         64
9         18         27         36         45         54         63         72
10        20         30         40         50         60         70         80
11        22         33         44         55         66         77         88
12        24         36         48         60         72         84         96
13        26         39         52         65         78         91        104
14        28         42         56         70         84         98        112
15        30         45         60         75         90        105        120
16        32         48         64         80         96        112        128
17        34         51         68         85        102        119        136
18        36         54         72         90        108        126        144
19        38         57         76         95        114        133        152
20        40         60         80        100        120        140        160
21        42         63         84        105        126        147        168
22        44         66         88        110        132        154        176
23        46         69         92        115        138        161        184
24        48         72         96        120        144        168        192
25        50         75        100        125        150        175        200
```

FIGURE 6-22 Output when 2 and 8 are entered

8. Write an application that can be used to determine if three line segments can form a triangle. Prompt the user for the length of three line segments as integers. If non-numeric characters are entered, re-prompt the user for new values. If the three lines could form a triangle, print the integers and a message indicating that they form a triangle. Use a state-controlled loop to allow users to enter as many different combinations as they want.



9. Print isosceles triangles. For each triangle, allow the user to input two values: a character to be used for printing the triangle and the size of the peak for the triangle. Test the input for valid characters. The size of the triangle should not be larger than 10. If an invalid non-numeric character is entered for size or if the value entered for size is larger than 10, use 3 as the default value. If an invalid entry is entered for the character, use an asterisk (*) as the default character. Allow multiple triangles to be printed. For example, if the user inputs # for the character and 6 for the peak, you should produce the following display:

```
#
##
###
####
#####
######
#####
####
###
##
#
```

10. Write an application that enables a user to input the grade and number of credit hours for any number of courses. Calculate the GPA on a 4.0 scale using those values. Grade point average (GPA) is calculated by dividing the total amount of grade points earned, sometimes referred to as quality points, by the total number of credit hours attempted. For each hour, an A receives 4 grade or quality points, a B receives 3 points, a C receives 2 points, and a D receives 1 point. Thus, a three–credit hour course receiving an A would have 12 quality points associated with the course. Allow the user to input any number of courses and associated grades. Display the number of hours earned and the calculated GPA.