25. The following program has several syntax errors as well as style inconsistencies. Correct the syntax errors and identify the style violations.

```
namespace Chapter2
{
  class converter
  {
  static void main(
{
              CONST int inches = 12;
              int x = 100; y = 10;
              float z = 22.45;
              double ans;
              ans=inches+z*x%y;
              System.write("The result is {f2:0} " + "ans");
}
}
```

PROGRAMMING EXERCISES

For each of the exercises, be sure to include appropriate comments, choose meaningful identifiers, and use proper indentations in your source code.

1. Write a program that converts a temperature given in Celsius into Fahrenheit. Test the program by performing a compile-time initialization of 32 for the original Celsius value. Display the values number aligned. The original temperature should show no digits after the decimal. One position following the decimal should be printed for the converted value. Be sure to provide labels for both values. Go into your source code and change the initialization value to 0. Rerun the application. Select additional test values and rerun the application.

2. Design an application that converts miles into feet and its equivalent metric kilometer measurement. Declare and initialize miles to 4.5. Show your miles and kilometers formatted with two positions to the right of the decimal. Feet should both be shown with no positions to the right of the decimal with comma separators. Be sure to provide labels for values and number align them. Once you get that portion running, go into your source code and change the initialization value for miles. Rerun the application and make sure that your values are still number aligned.

3. Write a program that computes the average of five exam scores. Declare and perform a compile-time initialization with the five exam values. Declare integer memory locations for the exam values. Use an integer

constant to define the number of scores. Print all scores. The average value should be formatted with two digits to the right of the decimal. Rerun the application with different values. Be sure to desk check your results.

4. Write a program that prints the number of quarters, dimes, nickels, and pennies that a customer should get back as change. Declare and initialize all memory locations as integers. On output, show the original change amount as a monetary amount, with two positions to the right of the decimal. Run your program once by performing a compile-time initialization using 92 cents for the value to be converted. Go into your source code and change the 92 to 27. Rerun the application. Be sure to desk check your solutions.

5. Write a program that shows the formatted retail price of shirts when there is a 15% markdown. Test the program by performing a compile-time initialization with an item labeled "Open Collar Running Shirt," which has a wholesale price of $41.00. How much savings is expected with the markdown? Display appropriately labeled retail and markdown values for the shirt. Once you get that running, go back into your source code, add lines of code that will reassign the memory location's values for a Razorback T-Shirt, which has a retail price of $36.00. Add additional lines of code, which will display the new item's information along with the previous item. What happens if the markdown goes to 20% or 10%?

6. Write a program that calculates and prints the take-home pay for a commissioned sales employee. Perform a compile-time initialization and store the name of Joshua Montain in a variable called employeeName. For practice working with the decimal data type, declare all monetary values as decimal. Employees earn 7% of their total sales as a commission. Employees pay federal tax rate of 18%. All employees contribute 10% of their earnings to a retirement program and pay an additional 6% of their earnings to Social Security. If Joshua's sales this month were $161,432, how much money will he take home? Produce a formatted report with your values labeled and number aligned showing the amount for each of the computed items and the sales commission percentage rate. Also show the total deductions. The final take home pay and total sales figure used for the calculations should be formatted with currency. All other values should have comma separators, no dollar sign, and display two positions to the right of the decimal. Select appropriate constants. After you finish displaying Joshua's data, change his sales to 1.3 million and rerun the application.

7. Write a tip calculating applications that can be used to determine what the tip and final charges would be given a total bill charge. Display the tip for 15% and 20% along with totals for each of the percentages. On output, show the original bill charge and each of the tip calculations with a final amount. Be sure to provide labels and number align the values.

8. Write a program that computes a weighted average giving the following weights.

   Homework: 10%
   Projects: 35%
   Quizzes: 10%
   Exams: 30%
   Final Exam: 15%

   Do a compile-time initialization with the following values: Homework: 97; Projects: 82; Quizzes: 60; Exams: 75; Final Exam 80. Display all values, including the weights, appropriately labeled and formatted. Rerun the application with different values.

9. Write a program that computes the amount of money the computer club will receive from the proceeds of their granola sales. They sell the granola bars for $1.50 per bar. Purchases for the granola are in cases, with each case having 100 bars. Each case costs $100.00. They are required to give the student government association 10% of their earnings. Display their proceeds, showing the amount given to the student government association. Show all the values formatted with currency. How much money would they make if they sold 29 cases?

10. In countries using the metric system, many products are sold by grams and kilograms as opposed to pounds and ounces. Write an application that converts grams to pounds and will display the price of the product by pound. Test your application by doing a compile-time initialization of Poutine, a common Canadian dish, made with French fries, which sells for $1.29 per 100 grams. Display both the metric and customary U.S. units. Be sure to provide labels for all values. Once you get that portion running, go into your source code and rerun the application using additional items, such as haricot verts, which are a type of green beans sold at 0.75 per 100 grams.