Summary

Section 4.1 Introduction

- Before writing a program to solve a problem, you must have a thorough understanding of the problem and a carefully planned approach to solving it. You must also understand the building blocks that are available and employ proven program-construction techniques.

Section 4.2 Algorithms

- Any computing problem can be solved by executing a series of actions (p. 102) in a specific order.
- A procedure for solving a problem in terms of the actions to execute and the order in which they execute is called an algorithm (p. 102).
- Specifying the order in which statements execute in a program is called program control (p. 102).

Section 4.3 Pseudocode

- Pseudocode (p. 103) is an informal language that helps you develop algorithms without having to worry about the strict details of Java language syntax.
- The pseudocode we use in this book is similar to everyday English—it's convenient and user friendly, but it's not an actual computer programming language. You may, of course, use your own native language(s) to develop your own pseudocode.
- Pseudocode helps you "think out" a program before attempting to write it in a programming language, such as Java.
- Carefully prepared pseudocode can easily be converted to a corresponding Java program.

Section 4.4 Control Structures

- Normally, statements in a program are executed one after the other in the order in which they're written. This process is called sequential execution (p. 103).
- Various Java statements enable you to specify that the next statement to execute is not necessarily the next one in sequence. This is called transfer of control (p. 103).
- Bohm and Jacopini demonstrated that all programs could be written in terms of only three control structures (p. 103)—the sequence structure, the selection structure and the repetition structure.
- The term "control structures" comes from the field of computer science. The *Java Language Specification* refers to "control structures" as "control statements" (p. 104).
- The sequence structure is built into Java. Unless directed otherwise, the computer executes Java statements one after the other in the order in which they're written—that is, in sequence.
- Anywhere a single action may be placed, several actions may be placed in sequence.
- Activity diagrams (p. 104) are part of the UML. An activity diagram models the workflow (p. 104; also called the activity) of a portion of a software system.
- Activity diagrams are composed of symbols (p. 104)—such as action-state symbols, diamonds and small circles—that are connected by transition arrows, which represent the flow of the activity.
- Action states (p. 104) contain action expressions that specify particular actions to perform.
- The arrows in an activity diagram represent transitions, which indicate the order in which the actions represented by the action states occur.
- The solid circle located at the top of an activity diagram represents the activity's initial state (p. 104)—the beginning of the workflow before the program performs the modeled actions.
- The solid circle surrounded by a hollow circle that appears at the bottom of the diagram represents the final state (p. 104)—the end of the workflow after the program performs its actions.

- Rectangles with their upper-right corners folded over are UML notes (p. 104)—explanatory remarks that describe the purpose of symbols in the diagram.
- Java has three types of selection statements (p. 105).
- The `if` single-selection statement (p. 105) selects or ignores one or more actions.
- The `if…else` double-selection statement selects between two actions or groups of actions.
- The `switch` statement is called a multiple-selection statement (p. 105) because it selects among many different actions or groups of actions.
- Java provides the `while`, `do…while` and `for` repetition (also called iteration or looping) statements that enable programs to perform statements repeatedly as long as a loop-continuation condition remains true.
- The `while` and `for` statements perform the action(s) in their bodies zero or more times—if the loop-continuation condition (p. 105) is initially false, the action(s) will not execute. The `do…while` statement performs the action(s) in its body one or more times.
- The words `if`, `else`, `switch`, `while`, `do` and `for` are Java keywords. Keywords cannot be used as identifiers, such as variable names.
- Every program is formed by combining as many sequence, selection and repetition statements (p. 105) as is appropriate for the algorithm the program implements.
- Single-entry/single-exit control statements (p. 105) are attached to one another by connecting the exit point of one to the entry point of the next. This is known as control-statement stacking.
- A control statement may also be nested (p. 105) inside another control statement.

Section 4.5 `if` Single-Selection Statement

- Programs use selection statements to choose among alternative courses of action.
- The single-selection `if` statement's activity diagram contains the diamond symbol, which indicates that a decision is to be made. The workflow follows a path determined by the symbol's associated guard conditions (p. 106). If a guard condition is true, the workflow enters the action state to which the corresponding transition arrow points.
- The `if` statement is a single-entry/single-exit control statement.

Section 4.6 `if…else` Double-Selection Statement

- The `if` single-selection statement performs an indicated action only when the condition is `true`.
- The `if…else` double-selection (p. 105) statement performs one action when the condition is true and another action when the condition is false.
- A program can test multiple cases with nested `if…else` statements (p. 107).
- The conditional operator (p. 110; `?:`) is Java's only ternary operator—it takes three operands. Together, the operands and the `?:` symbol form a conditional expression (p. 110).
- The Java compiler associates an `else` with the immediately preceding `if` unless told to do otherwise by the placement of braces.
- The `if` statement expects one statement in its body. To include several statements in the body of an `if` (or the body of an `else` for an `if…else` statement), enclose the statements in braces.
- A block (p. 110) of statements can be placed anywhere that a single statement can be placed.
- A logic error (p. 110) has its effect at execution time. A fatal logic error (p. 110) causes a program to fail and terminate prematurely. A nonfatal logic error (p. 110) allows a program to continue executing, but causes it to produce incorrect results.

- Just as a block can be placed anywhere a single statement can be placed, you can also use an empty statement, represented by placing a semicolon (`;`) where a statement would normally be.

Section 4.8 `while` Repetition Statement

- The `while` repetition statement (p. 114) allows you to specify that a program should repeat an action while some condition remains true.
- The UML's merge (p. 114) symbol joins two flows of activity into one.
- The decision and merge symbols can be distinguished by the number of incoming and outgoing transition arrows. A decision symbol has one transition arrow pointing to the diamond and two or more transition arrows pointing out from the diamond to indicate possible transitions from that point. Each transition arrow pointing out of a decision symbol has a guard condition. A merge symbol has two or more transition arrows pointing to the diamond and only one transition arrow pointing from the diamond, to indicate multiple activity flows merging to continue the activity. None of the transition arrows associated with a merge symbol has a guard condition.

Section 4.9 Formulating Algorithms: Counter-Controlled Repetition

- Counter-controlled repetition (p. 115) uses a variable called a counter (or control variable) to control the number of times a set of statements execute.
- Counter-controlled repetition is often called definite repetition (p. 115), because the number of repetitions is known before the loop begins executing.
- A total (p. 115) is a variable used to accumulate the sum of several values. Variables used to store totals are normally initialized to zero before being used in a program.
- A local variable's declaration must appear before the variable is used in that method. A local variable cannot be accessed outside the method in which it's declared.
- Dividing two integers results in integer division—the calculation's fractional part is truncated.

Section 4.10 Formulating Algorithms: Sentinel-Controlled Repetition

- In sentinel-controlled repetition (p. 119), a special value called a sentinel value (also called a signal value, a dummy value or a flag value) is used to indicate "end of data entry."
- A sentinel value must be chosen that cannot be confused with an acceptable input value.
- Top-down, stepwise refinement (p. 120) is essential to the development of well-structured programs.
- Division by zero is a logic error.
- To perform a floating-point calculation with integer values, cast one of the integers to type `double`.
- Java knows how to evaluate only arithmetic expressions in which the operands' types are identical. To ensure this, Java performs an operation called promotion on selected operands.
- The unary cast operator is formed by placing parentheses around the name of a type.

Section 4.12 Compound Assignment Operators

- The compound assignment operators (p. 131) abbreviate assignment expressions. Statements of the form

  variable = *variable* *operator* *expression*;

  where *operator* is one of the binary operators `+`, `-`, `*`, `/` or `%`, can be written in the form

  variable *operator*= *expression*;

- The `+=` operator adds the value of the expression on the right of the operator to the value of the variable on the left of the operator and stores the result in the variable on the left of the operator.

Section 4.13 Increment and Decrement Operators

- The unary increment operator, `++`, and the unary decrement operator, `--`, add 1 to or subtract 1 from the value of a numeric variable (p. 131).
- An increment or decrement operator that's prefixed (p. 131) to a variable is the prefix increment or prefix decrement operator, respectively. An increment or decrement operator that's postfixed (p. 131) to a variable is the postfix increment or postfix decrement operator, respectively.
- Using the prefix increment or decrement operator to add or subtract 1 is known as preincrementing or predecrementing, respectively.
- Preincrementing or predecrementing a variable causes the variable to be incremented or decremented by 1; then the new value of the variable is used in the expression in which it appears.
- Using the postfix increment or decrement operator to add or subtract 1 is known as postincrementing or postdecrementing, respectively.
- Postincrementing or postdecrementing the variable causes its value to be used in the expression in which it appears; then the variable's value is incremented or decremented by 1.
- When incrementing or decrementing a variable in a statement by itself, the prefix and postfix increment have the same effect, and the prefix and postfix decrement have the same effect.

Section 4.14 Primitive Types

- Java requires all variables to have a type. Thus, Java is referred to as a strongly typed language (p. 134).
- Java uses Unicode characters and IEEE 754 floating-point numbers.

Self-Review Exercises

4.1 Fill in the blanks in each of the following statements:

a) All programs can be written in terms of three types of control structures: ________, ________ and ________.
b) The ________ statement is used to execute one action when a condition is true and another when that condition is false.
c) Repeating a set of instructions a specific number of times is called ________ repetition.
d) When it's not known in advance how many times a set of statements will be repeated, a(n) ________ value can be used to terminate the repetition.
e) The ________ structure is built into Java; by default, statements execute in the order they appear.
f) Instance variables of types `char`, `byte`, `short`, `int`, `long`, `float` and `double` are all given the value ________ by default.
g) Java is a(n) ________ language; it requires all variables to have a type.
h) If the increment operator is ________ to a variable, first the variable is incremented by 1, then its new value is used in the expression.

4.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) An algorithm is a procedure for solving a problem in terms of the actions to execute and the order in which they execute.
b) A set of statements contained within a pair of parentheses is called a block.
c) A selection statement specifies that an action is to be repeated while some condition remains true.
d) A nested control statement appears in the body of another control statement.
e) Java provides the arithmetic compound assignment operators `+=`, `-=`, `*=`, `/=` and `%=` for abbreviating assignment expressions.

f) The primitive types (`boolean`, `char`, `byte`, `short`, `int`, `long`, `float` and `double`) are portable across only Windows platforms.
g) Specifying the order in which statements execute in a program is called program control.
h) The unary cast operator `(double)` creates a temporary integer copy of its operand.
i) Instance variables of type `boolean` are given the value `true` by default.
j) Pseudocode helps you think out a program before attempting to write it in a programming language.

4.3 Write four different Java statements that each add 1 to integer variable `x`.

4.4 Write Java statements to accomplish each of the following tasks:
a) Use one statement to assign the sum of `x` and `y` to `z`, then increment `x` by 1.
b) Test whether variable `count` is greater than 10. If it is, print `"Count is greater than 10"`.
c) Use one statement to decrement the variable `x` by 1, then subtract it from variable `total` and store the result in variable `total`.
d) Calculate the remainder after `q` is divided by `divisor`, and assign the result to `q`. Write this statement in two different ways.

4.5 Write a Java statement to accomplish each of the following tasks:
a) Declare variables `sum` of type `int` and initialize it to `0`.
b) Declare variables `x` of type `int` and initialize it to `1`.
c) Add variable `x` to variable `sum`, and assign the result to variable `sum`.
d) Print `"The sum is: "`, followed by the value of variable `sum`.

4.6 Combine the statements that you wrote in Exercise 4.5 into a Java application that calculates and prints the sum of the integers from 1 to 10. Use a `while` statement to loop through the calculation and increment statements. The loop should terminate when the value of `x` becomes 11.

4.7 Determine the value of the variables in the statement `product *= x++;` after the calculation is performed. Assume that all variables are type `int` and initially have the value 5.

4.8 Identify and correct the errors in each of the following sets of code:
a)
```
while (c <= 5)
{
   product *= c;
   ++c;
```
b)
```
if (gender == 1)
   System.out.println("Woman");
else;
   System.out.println("Man");
```

4.9 What is wrong with the following `while` statement?

```
while (z >= 0)
   sum += z;
```

Answers to Self-Review Exercises

4.1 a) sequence, selection, repetition. b) `if…else`. c) counter-controlled (or definite). d) sentinel, signal, flag or dummy. e) sequence. f) `0` (zero). g) strongly typed. h) prefixed.

4.2 a) True. b) False. A set of statements contained within a pair of braces (`{` and `}`) is called a block. c) False. A repetition statement specifies that an action is to be repeated while some condition remains true. d) True. e) True. f) False. The primitive types (`boolean`, `char`, `byte`, `short`, `int`, `long`, `float` and `double`) are portable across all computer platforms that support Java. g) True. h) False. The unary cast operator `(double)` creates a temporary floating-point copy of its operand. i) False. Instance variables of type `boolean` are given the value `false` by default. j) True.

4.3

```
x = x + 1;
x += 1;
++x;
x++;
```

4.4

a) `z = x++ + y;`

b)
```
if (count > 10)
   System.out.println("Count is greater than 10");
```

c) `total -= --x;`

d)
```
q %= divisor;
q = q % divisor;
```

4.5

a) `int sum = 0;`

b) `int x = 1;`

c) `sum += x;` or `sum = sum + x;`

d) `System.out.printf("The sum is: %d%n", sum);`

4.6 The program is as follows:

```java
// Exercise 4.6: Calculate.java
// Calculate the sum of the integers from 1 to 10
public class Calculate
{
   public static void main(String[] args)
   {
      int sum = 0;
      int x = 1;

      while (x <= 10) // while x is less than or equal to 10
      {
         sum += x; // add x to sum
         ++x; // increment x
      }

      System.out.printf("The sum is: %d%n", sum);
   }
} // end class Calculate
```

```
The sum is: 55
```

4.7 `product = 25, x = 6`

4.8

a) Error: The closing right brace of the `while` statement's body is missing.
Correction: Add a closing right brace after the statement `++c;`.

b) Error: The semicolon after `else` results in a logic error. The second output statement will always be executed.
Correction: Remove the semicolon after `else`.

4.9 The value of the variable `z` is never changed in the `while` statement. Therefore, if the loop-continuation condition `(z >= 0)` is true, an infinite loop is created. To prevent an infinite loop from occurring, `z` must be decremented so that it eventually becomes less than 0.

Exercises

4.10 Compare and contrast the `if` single-selection statement and the `while` repetition statement. How are these two statements similar? How are they different?

4.11 Explain what happens when a Java program attempts to divide one integer by another. What happens to the fractional part of the calculation? How can you avoid that outcome?

4.12 Describe the two ways in which control statements can be combined.

4.13 What type of repetition would be appropriate for calculating the sum of the first 100 positive integers? What type would be appropriate for calculating the sum of an arbitrary number of positive integers? Briefly describe how each of these tasks could be performed.

4.14 What is the difference between preincrementing and postincrementing a variable?

4.15 Identify and correct the errors in each of the following pieces of code. [*Note:* There may be more than one error in each piece of code.]

a)
```
if (age >= 65);
   System.out.println("Age is greater than or equal to 65");
else
   System.out.println("Age is less than 65)";
```

b)
```
int x = 1, total;
while (x <= 10)
{
   total += x;
   ++x;
}
```

c)
```
while (x <= 100)
   total += x;
   ++x;
```

d)
```
while (y > 0)
{
   System.out.println(y);
   ++y;
```

4.16 What does the following program print?

```
// Exercise 4.16: Mystery.java
public class Mystery
{
   public static void main(String[] args)
   {
      int x = 1;
      int total = 0;

      while (x <= 10)
      {
         int y = x * x;
         System.out.println(y);
         total += y;
         ++x;
      }

      System.out.printf("Total is %d%n", total);
   }
} // end class Mystery
```

For Exercise 4.17 through Exercise 4.20, perform each of the following steps:

a) Read the problem statement.
b) Formulate the algorithm using pseudocode and top-down, stepwise refinement.
c) Write a Java program.

d) Test, debug and execute the Java program.
e) Process three complete sets of data.

4.17 *(Gas Mileage)* Drivers are concerned with the mileage their automobiles get. One driver has kept track of several trips by recording the miles driven and gallons used for each tankful. Develop a Java application that will input the miles driven and gallons used (both as integers) for each trip. The program should calculate and display the miles per gallon obtained for each trip and print the combined miles per gallon obtained for all trips up to this point. All averaging calculations should produce floating-point results. Use class `Scanner` and sentinel-controlled repetition to obtain the data from the user.

4.18 *(Credit Limit Calculator)* Develop a Java application that determines whether any of several department-store customers has exceeded the credit limit on a charge account. For each customer, the following facts are available:

a) account number
b) balance at the beginning of the month
c) total of all items charged by the customer this month
d) total of all credits applied to the customer's account this month
e) allowed credit limit.

The program should input all these facts as integers, calculate the new balance (= *beginning balance* + *charges* – *credits*), display the new balance and determine whether the new balance exceeds the customer's credit limit. For those customers whose credit limit is exceeded, the program should display the message `"Credit limit exceeded"`.

4.19 *(Sales Commission Calculator)* A large company pays its salespeople on a commission basis. The salespeople receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5,000 worth of merchandise in a week receives $200 plus 9% of $5000, or a total of $650. You've been supplied with a list of the items sold by each salesperson. The values of these items are as follows:

Item	Value
1	239.99
2	129.75
3	99.95
4	350.89

Develop a Java application that inputs one salesperson's items sold for last week and calculates and displays that salesperson's earnings. There's no limit to the number of items that can be sold.

4.20 *(Salary Calculator)* Develop a Java application that determines the gross pay for each of three employees. The company pays straight time for the first 40 hours worked by each employee and time and a half for all hours worked in excess of 40. You're given a list of the employees, their number of hours worked last week and their hourly rates. Your program should input this information for each employee, then determine and display the employee's gross pay. Use class `Scanner` to input the data.

4.21 *(Find the Largest Number)* The process of finding the largest value is used frequently in computer applications. For example, a program that determines the winner of a sales contest would input the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a pseudocode program, then a Java application that inputs a series of 10 integers and determines and prints the largest integer. Your program should use at least the following three variables:

a) `counter`: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed).
b) `number`: The integer most recently input by the user.
c) `largest`: The largest number found so far.

4.22 *(Tabular Output)* Write a Java application that uses looping to print the following table of values:

```
N       10*N    100*N   1000*N

1       10      100     1000
2       20      200     2000
3       30      300     3000
4       40      400     4000
5       50      500     5000
```

4.23 *(Find the Two Largest Numbers)* Using an approach similar to that for Exercise 4.21, find the *two* largest values of the 10 values entered. [*Note:* You may input each number only once.]

4.24 *(Validating User Input)* Modify the program in Fig. 4.12 to validate its inputs. For any input, if the value entered is other than 1 or 2, keep looping until the user enters a correct value.

4.25 What does the following program print?

```java
// Exercise 4.25: Mystery2.java
public class Mystery2
{
   public static void main(String[] args)
   {
      int count = 1;

      while (count <= 10)
      {
         System.out.println(count % 2 == 1 ? "****" : "++++++++");
         ++count;
      }
   }
} // end class Mystery2
```

4.26 What does the following program print?

```java
// Exercise 4.26: Mystery3.java
public class Mystery3
{
   public static void main(String[] args)
   {
      int row = 10;

      while (row >= 1)
      {
         int column = 1;

         while (column <= 10)
         {
            System.out.print(row % 2 == 1 ? "<" : ">");
            ++column;
         }

         --row;
         System.out.println();
      }
   }
} // end class Mystery3
```

4.27 *(Dangling-`else` Problem)* Determine the output for each of the given sets of code when `x` is `9` and `y` is `11` and when `x` is `11` and `y` is `9`. The compiler ignores the indentation in a Java program. Also, the Java compiler always associates an `else` with the immediately preceding `if` unless told to do otherwise by the placement of braces (`{}`). On first glance, you may not be sure which `if` a particular `else` matches—this situation is referred to as the "dangling-`else` problem." We've eliminated the indentation from the following code to make the problem more challenging. [*Hint:* Apply the indentation conventions you've learned.]

a)
```
if (x < 10)
if (y > 10)
System.out.println("*****");
else
System.out.println("#####");
System.out.println("$$$$$");
```

b)
```
if (x < 10)
{
if (y > 10)
System.out.println("*****");
}
else
{
System.out.println("#####");
System.out.println("$$$$$");
}
```

4.28 *(Another Dangling-`else` Problem)* Modify the given code to produce the output shown in each part of the problem. Use proper indentation techniques. Make no changes other than inserting braces and changing the indentation of the code. The compiler ignores indentation in a Java program. We've eliminated the indentation from the given code to make the problem more challenging. [*Note:* It's possible that no modification is necessary for some of the parts.]

```
if (y == 8)
if (x == 5)
System.out.println("@@@@@");
else
System.out.println("#####");
System.out.println("$$$$$");
System.out.println("&&&&&");
```

a) Assuming that `x = 5` and `y = 8`, the following output is produced:

```
@@@@@
$$$$$
&&&&&
```

b) Assuming that `x = 5` and `y = 8`, the following output is produced:

```
@@@@@
```

c) Assuming that `x = 5` and `y = 8`, the following output is produced:

```
@@@@@
```

d) Assuming that `x = 5` and `y = 7`, the following output is produced. [*Note:* The last three output statements after the `else` are all part of a block.]

```
#####
$$$$$
&&&&&
```

4.29 *(Square of Asterisks)* Write an application that prompts the user to enter the size of the side of a square, then displays a hollow square of that size made of asterisks. Your program should work for squares of all side lengths between 1 and 20.

4.30 *(Palindromes)* A palindrome is a sequence of characters that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write an application that reads in a five-digit integer and determines whether it's a palindrome. If the number is not five digits long, display an error message and allow the user to enter a new value.

4.31 *(Printing the Decimal Equivalent of a Binary Number)* Write an application that inputs an integer containing only 0s and 1s (i.e., a binary integer) and prints its decimal equivalent. [*Hint:* Use the remainder and division operators to pick off the binary number's digits one at a time, from right to left. In the decimal number system, the rightmost digit has a positional value of 1 and the next digit to the left a positional value of 10, then 100, then 1000, and so on. The decimal number 234 can be interpreted as 4 * 1 + 3 * 10 + 2 * 100. In the binary number system, the rightmost digit has a positional value of 1, the next digit to the left a positional value of 2, then 4, then 8, and so on. The decimal equivalent of binary 1101 is 1 * 1 + 0 * 2 + 1 * 4 + 1 * 8, or 1 + 0 + 4 + 8 or, 13.]

4.32 *(Checkerboard Pattern of Asterisks)* Write an application that uses only the output statements

```
System.out.print("* ");
System.out.print(" ");
System.out.println();
```

to display the checkerboard pattern that follows. A `System.out.println` method call with no arguments causes the program to output a single newline character. [*Hint:* Repetition statements are required.]

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

4.33 *(Multiples of 2 with an Infinite Loop)* Write an application that keeps displaying in the command window the multiples of the integer 2—namely, 2, 4, 8, 16, 32, 64, and so on. Your loop should not terminate (i.e., it should create an infinite loop). What happens when you run this program?

4.34 *(What's Wrong with This Code?)* What is wrong with the following statement? Provide the correct statement to add one to the sum of `x` and `y`.

```
System.out.println(++(x + y));
```

4.35 *(Sides of a Triangle)* Write an application that reads three nonzero values entered by the user and determines and prints whether they could represent the sides of a triangle.

4.36 *(Sides of a Right Triangle)* Write an application that reads three nonzero integers and determines and prints whether they could represent the sides of a right triangle.

4.37 *(Factorial)* The factorial of a nonnegative integer *n* is written as *n*! (pronounced "*n* factorial") and is defined as follows:

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1 \quad \text{(for values of } n \text{ greater than or equal to 1)}$$

and

$$n! = 1 \quad (\text{for } n = 0)$$

For example, $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$, which is 120.

a) Write an application that reads a nonnegative integer and computes and prints its factorial.

b) Write an application that estimates the value of the mathematical constant *e* by using the following formula. Allow the user to enter the number of terms to calculate.

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

c) Write an application that computes the value of e^x by using the following formula. Allow the user to enter the number of terms to calculate.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

Making a Difference

4.38 ***(Enforcing Privacy with Cryptography)*** The explosive growth of Internet communications and data storage on Internet-connected computers has greatly increased privacy concerns. The field of cryptography is concerned with coding data to make it difficult (and hopefully—with the most advanced schemes—impossible) for unauthorized users to read. In this exercise you'll investigate a simple scheme for encrypting and decrypting data. A company that wants to send data over the Internet has asked you to write a program that will encrypt it so that it may be transmitted more securely. All the data is transmitted as four-digit integers. Your application should read a four-digit integer entered by the user and encrypt it as follows: Replace each digit with the result of adding 7 to the digit and getting the remainder after dividing the new value by 10. Then swap the first digit with the third, and swap the second digit with the fourth. Then print the encrypted integer. Write a separate application that inputs an encrypted four-digit integer and decrypts it (by reversing the encryption scheme) to form the original number. [*Optional reading project:* Research "public key cryptography" in general and the PGP (Pretty Good Privacy) specific public key scheme. You may also want to investigate the RSA scheme, which is widely used in industrial-strength applications.]

4.39 ***(World Population Growth)*** World population has grown considerably over the centuries. Continued growth could eventually challenge the limits of breathable air, drinkable water, arable cropland and other limited resources. There's evidence that growth has been slowing in recent years and that world population could peak some time this century, then start to decline.

For this exercise, research world population growth issues online. *Be sure to investigate various viewpoints.* Get estimates for the current world population and its growth rate (the percentage by which it's likely to increase this year). Write a program that calculates world population growth each year for the next 75 years, *using the simplifying assumption that the current growth rate will stay constant.* Print the results in a table. The first column should display the year from year 1 to year 75. The second column should display the anticipated world population at the end of that year. The third column should display the numerical increase in the world population that would occur that year. Using your results, determine the year in which the population would be double what it is today, if this year's growth rate were to persist.