f)
```
Const MAX_CLASS_SIZE As Integer = 22
Dim currentSize As Integer
currentSize = Val(Me.txtSize.Text)
Me.lblEmptySeats.Text = MAX_CLASS_SIZE
```

13. What value is assigned to taxRate in the following statement if `seven percent` is typed in the txtTaxRate text box?

```
taxRate = Val(Me.txtTaxRate.Text)
```

14. Write each equation as a Visual Basic assignment statement, assuming π is a constant named PI:

a) $A = lw$ (geometry)

b) $P = \dfrac{R - C}{N}$ (business)

c) $A = \dfrac{h(b_1 + b_2)}{N}$ (geometry)

d) $V = \dfrac{4}{3}\pi r^3$ (geometry)

e) $A = \dfrac{F + S + T}{3}$ (algebra)

f) $P = \dfrac{5F}{4d^2}$ (physics)

g) $A = P + Pr\,t$ (business)

h) $M = \dfrac{Pr(1+r)^n}{(1+r)^n - 1}$ (algebra)

i) $x = \dfrac{-b\sqrt{b^2 - 4ac}}{2a}$ (algebra)

15. Rewrite the equations in parts (a) through (h) of question 14 to solve for the variable listed below and then write each equation as a Visual Basic assignment statement:
a) l
b) R
c) b1
d) r
e) T
f) F
g) P
h) P

True/False

16. Determine if each of the following statements is true or false. If false, explain why.
a) A variable must be declared before it is used.
b) Multiple variables with different data types can be declared in a single statement.
c) A runtime error results in the termination of program execution.
d) The Val() function converts text box data to a string value.
e) An `Integer` variable uses 4 bytes of memory to store its value.
f) `Boolean` variables are initialized to `True`.
g) In the variable assignment statement `x = 4.5`, where x is an `Integer` variable, x is assigned the value `4`.
h) Modulus division returns the remainder resulting from division.
i) In the variable assignment statement `x = 14 Mod 4 * 2`, where x is a `Double` variable, x is assigned the value `4`.
j) Constant declarations must be placed in the beginning of a program.
k) Visual Basic sees no difference between the variables `NEW` and `New`.
l) Keywords make good variable identifiers.
m) `Private` is a keyword.
n) The Visual Basic IDE informs the programmer of a syntax error.
o) Errors that violate the rules of Visual Basic are called semantic errors.
p) Run-time errors are always detected by the compiler.
q) A breakpoint temporarily stops the execution of a program, but pressing the right-arrow key continues execution.

Exercise 1 ————————————————ObjectHeight

The height of an object at any given time dropped from a starting height of 100 meters is given by the equation `h=100-4.9*t²` where `t` is the time in seconds. Create an ObjectHeight application that prompts the user for a time less than 4.5 seconds and then displays the height of the object at that time when Height is clicked. The application interface should look similar to:
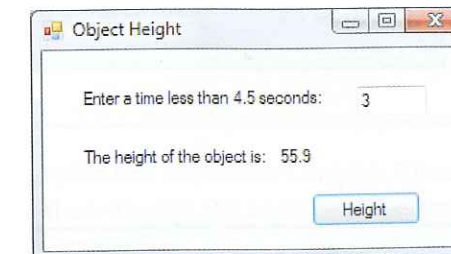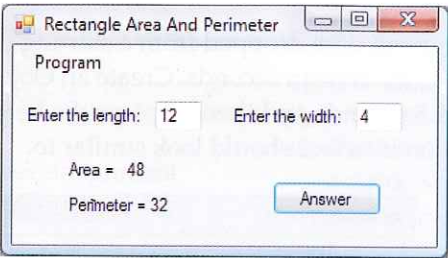

Exercise 2 ————————————————TemperatureConversion

a) Create a TemperatureConversion application that prompts the user for a temperature in degrees Fahrenheit and then displays the temperature in degrees Celsius when Celsius is clicked. Use the formula `C=5/9(F-32)` to make the conversion. Test the program with values `212`, `98.6`, `50`, `32`, and `-40`. The application interface should look similar to:
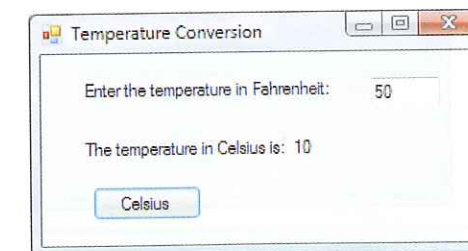
b) Modify the TemperatureConversion application to include a Program menu with Celsius, Fahrenheit, and Exit commands. Use the formula in part (a) above to determine the formula for converting from Celsius to Fahrenheit. Modify the prompt to display `Enter the temperature:` Modify the temperature label appropriately. Remove the Celsius button and corresponding code and size the form appropriately.

Exercise 3 ♻ — RectangleAreaAndPerimeter

Modify the RectangleAreaAndPerimeter application created in Chapter 2 Exercise 8 to prompt the user for the length and width of the rectangle and then display the area and perimeter of the rectangle when Answer is clicked. The application interface should look similar to:
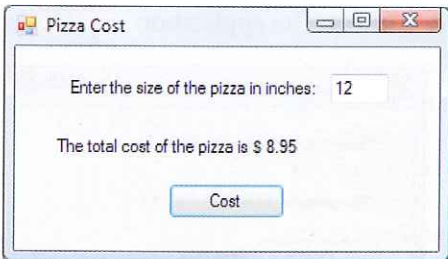
```
Rectangle Area And Perimeter
┌ Program ──────────────────────────┐
│  Enter the length:  12    Enter the width:  4  │
│                                   │
│          Area =  48               │
│                                   │
│          Perimeter = 32     [ Answer ]  │
└───────────────────────────────────┘
```

Exercise 4 — PizzaCost
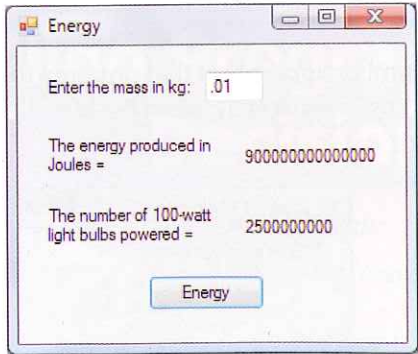
The cost of making a pizza at a local shop is as follows:

- Labor cost is $0.75 per pizza, regardless of size
- Rent cost is $1.00 per pizza, regardless of size
- Materials is $0.05*diameter*diameter (diameter is measured in inches)

Create a PizzaCost application that prompts the user for the size of a pizza and then displays the cost of making the pizza when Cost is clicked. The application interface should look similar to:
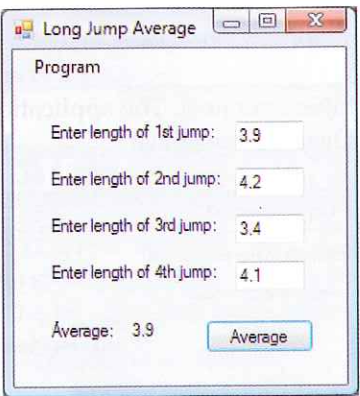
```
Pizza Cost
┌────────────────────────────────────┐
│  Enter the size of the pizza in inches:  12  │
│                                    │
│  The total cost of the pizza is $ 8.95  │
│                                    │
│              [ Cost ]              │
└────────────────────────────────────┘
```

Exercise 5 — Energy

Einstein's famous formula, $e=mc^2$, gives the amount of energy released by the complete conversion of matter of mass m into energy e. If m represents the mass in kilograms and c represents the speed of light in meters per second ($3.0 * 10^8$ m/s), then the result is in the energy units Joules. It takes 360000 Joules to light a 100-watt light bulb for an hour. Create an Energy application that prompts the user for a mass in kilograms and then displays the energy and the number of light bulbs that could be powered when Energy is clicked. The application interface should look similar to:
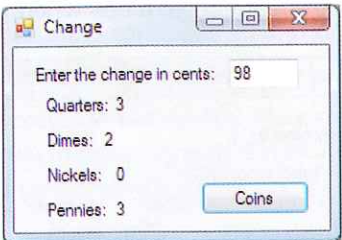
```
Energy
┌────────────────────────────────────┐
│  Enter the mass in kg:   .01        │
│                                    │
│  The energy produced in             │
│  Joules =              900000000000000  │
│                                    │
│  The number of 100-watt             │
│  light bulbs powered =    2500000000    │
│                                    │
│              [ Energy ]             │
└────────────────────────────────────┘
```

Exercise 6 ♻ — LongJumpAverage

Modify the LongJumpAverage application created in Chapter 2 Exercise 9 to prompt the user for the lengths of four long jumps and then display the average jump length when Average is clicked. The application interface should look similar to:

```
Long Jump Average
┌ Program ──────────────────────────┐
│  Enter length of 1st jump:  3.9     │
│  Enter length of 2nd jump:  4.2     │
│  Enter length of 3rd jump:  3.4     │
│  Enter length of 4th jump:  4.1     │
│                                   │
│  Average:  3.9       [ Average ]  │
└───────────────────────────────────┘
```

Exercise 7 — Change

Create a Change application that prompts the user for an amount less than 100 and then displays the minimum number of coins necessary to make the change when Coins is clicked. The change can be made up of quarters, dimes, nickels, and pennies. The application interface should look similar to:

```
Change
┌────────────────────────────────────┐
│  Enter the change in cents:  98     │
│  Quarters:  3                       │
│  Dimes:  2                          │
│  Nickels:  0                        │
│  Pennies:  3           [ Coins ]    │
└────────────────────────────────────┘
```

Exercise 8 — DigitsOfANumber

a) Create a DigitsOfANumber application that prompts the user for a two-digit number and then displays the digits separately when Digits is clicked. The application interface should look similar to:
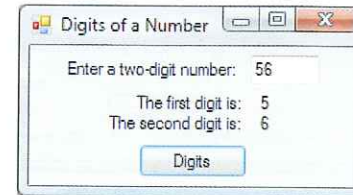
b) Modify the DigitsOfANumber application to include a Program menu with a Digits command.

Exercise 9 — TimeConversion

Create a TimeConversion application that prompts the user for a time in minutes and then displays the time in seconds or hour:minute format depending on the radio button clicked. Be sure to consider times where the number of minutes left over is less than 10. For example, 184 minutes in hour:minute format is 3:04 (*Hint*: use the modulus operator). The application interface should look similar to the following after clicking Minutes to hour:minute format:
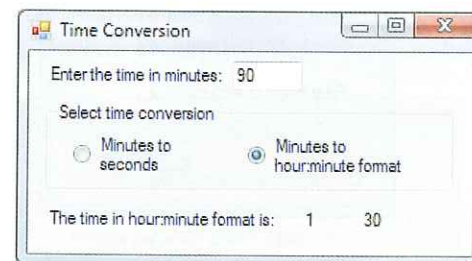
Exercise 10 — International

Create an International application that displays the phrase "My Name is" in English, Spanish, or French. Each radio button should display a tool tip. A *tool tip* displays information to the user about the purpose of an interface object and appears in a box when the mouse hovers on an object:
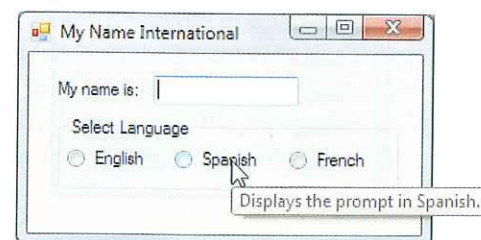
To add a ToolTip component, click the ToolTip control in the Toolbox and then click the form. The ToolTip component is shown in the component tray at the bottom of the Design window. The ToolTip property for each object should be set to the text that is to appear when the mouse hovers over that object.

Chapter 4
Controlling Program Flow with Decision Structures

Key Concepts

Controlling the flow of a program
Generating random numbers
Writing algorithms
Writing compound Boolean expressions
Using message boxes and counters
Selecting appropriate test data
Creating and modifying problem solutions

Case Study

Pizza Order application

The If...Then Statement

conditional control structure

The If...Then statement is a *conditional control structure*, also called a *decision structure*, which executes a set of statements when a condition is true. The If...Then statement takes the form:

```
If condition Then
    statements
End If
```

TIP The condition of an If...Then statement should never make an equality comparison between floating point numbers because of the possibility of roundoff error.

For example, in the following If...Then statement guess = 7 is the condition, and there is one statement that will execute when this condition is true:

```
If guess = 7 Then
    Me.lblMessage.Text = "You guessed it!"
End If
```

In the condition, the equal sign (=) is used as a relational operator to determine if the value of guess is equal to 7. If equal, then the Text property of lblMessage is changed. If not, program flow continues to the next statement after the End If.

Boolean expression
relational operators

The condition of an If...Then statement is a *Boolean expression*, which evaluates to either True or False. *Relational operators* can be used to form Boolean expressions. There are six relational operators:

TIP The equal sign (=) is used as a relational operator as well as an assignment operator.

Operator	Meaning
=	equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
<>	not equal to