3. Design the logic, and write the rest of the program using a nested If Then Else statement.

4. Compile the program.

5. Execute the program entering the following as input:
   - Employee's name: Kim Smith
   - Number of shifts: 25
   - Number of transactions: 75
   - Transaction dollar value: 40000.00

6. Your output should be:
   - Employee Name: Kim Smith
   - Employee Bonus: $50

You learn how to insert a decimal point and control the number of places that appear after the decimal point in Chapter 9 of this book.

**The Select Case Statement**

The Select Case statement is similar to a nested If statement because it is also a multipath decision statement. A Select Case statement offers the advantage of being easier for you to read than nested If statements, and a Select Case statement is also easier for you, the programmer, to maintain. You use the Select Case statement in situations when you want to compare an expression with several numeric or string constants.

The syntax for writing a Select Case statement in Visual Basic is as follows:

```
Select Case testexpression
    Case expressionlist1
        statement(s)
    Case expressionlist2
        statement(s)
    Case expressionlist3
        statement(s)
    Case Else
        statement(s)
End Select
```

You begin writing a Select Case statement with the keywords Select Case. Following Select Case, you include an expression that evaluates to a numeric or string value. Cases are then defined within the Select Case statement by using the keyword Case as a label, and including a numeric or string value after this label. For example, you could include an integer constant such as 10 or an arithmetic expression that evaluates to an integer such as 10/2. You could also include a string constant such as "Hello". The computer evaluates the testexpression in the Select Case statement and then compares it to the expressionlist values following the Case labels. If the testexpression and an expressionlist values match, then the computer executes the statement(s) that follow until it encounters the next Case label or the End Select statement. You can use the keyword Else to establish a case for values that do not match any of the expressionlist values following the Case labels.
Decision Statements

The following code sample illustrates the use of the Select Case statement in Visual Basic:

```vbnet
Dim DeptNum As Integer
Dim DeptName As String
DeptName = "Development"
Select Case DeptName
    Case "Marketing"
        DeptNum = 1
    Case "Development"
        DeptNum = 2
    Case "Sales"
        DeptNum = 3
    Case Else
        DeptNum = 0
End Select
System.Console.WriteLine("Department: ", DeptNum)
```

If the test expression matches more than one expression list value, only the statement(s)
following the first match are executed.

In the preceding example, when the program encounters the Select Case statement, the value of the variable named DeptName is "Development". The value "Development" matches the string constant in the second case of the Select Case statement. Therefore, the value 2 is assigned to the Integer variable named DeptNum. A Case label is encountered next, and causes the program to exit from the Select Case statement. The statement following the Select Case statement System.Console.WriteLine("Department: ", DeptNum) executes next.

In Visual Basic, the expression list can be made up of one or more of the following:

- expression(s)
- expression To expression
- Is comparison operator expression

The following code sample illustrates the use of various forms of the expression list in the Select Case statement in Visual Basic:

```vbnet
Dim YearsOfService As Integer
Dim VacationDays As Integer
YearsOfService = 22
Select Case YearsOfService
    Case 1 To 10
        VacationDays = 10
    Case 11 To 14
        VacationDays = 15
    Case 15 To 19
        VacationDays = 20
    Case Is >= 20
        VacationDays = 25
```
Writing Programs that Make Decisions

Case Else
  VacationDays = 0
End Select

In the preceding example, when the program encounters the Select Case statement, the value of the variable named YearsOfService is 22. The program then encounters the first case, which is Case 1 To 10. The value of YearsOfService is not in the range of 1 through 10 so the program then tests the second case, which is Case 11 To 14. The value of YearsOfService is not in the range of 11 through 14, so the program then tests the third case, which is Case 15 To 19. The value of YearsOfService is not in the range of 15 through 19 so the program then tests the fourth case, which is Case Is >= 20. The value of YearsOfService is greater than or equal to 20. Therefore, the value 25 is assigned to the Integer variable named VacationDays. A Case label is encountered next, and causes the program to exit from the Select Case statement. The statement following the Select Case statement System.Console.WriteLine("Vacation Days: " & VacationDays) executes next.

You can also include multiple values in an expression list by including a comma between the values as follows: Case 1, 3, 5.

Exercise 4-4: Using a Select Case Statement

In this exercise, you use what you have learned about the Select Case statement. Study the following code and then answer Questions 1–4.

First, examine the following code:

Dim NumValue As Integer = 10
Dim Answer As Integer = 0
Select Case NumValue
  Case 5
    Answer = Answer + 5
  Case 10
    Answer = Answer + 10
  Case 15
    Answer = Answer + 15
  Case 20
    Answer = Answer + 20
  Case 25
    Answer = Answer + 25
  Case Else
    Answer = 0
End Select
System.Console.WriteLine("Answer: " & Answer)

1. What is the value of answer if the value of NumValue is 10?
2. What is the value of answer if the value of NumValue is 40?
3. What is the value of answer if the value of NumValue is 5?

4. What is the value of answer if the value of NumValue is 17?

Lab 4-4: Using a Select Case Statement

In this lab, you complete a prewritten Visual Basic program that calculates an employee's end-of-year bonus and prints the employee's name, yearly salary, performance rating, and bonus. In this program, bonuses are calculated based on employees' annual salary and their performance rating. The rating system is contained in Table 4-5.

<table>
<thead>
<tr>
<th>Rating</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25% of annual salary</td>
</tr>
<tr>
<td>2</td>
<td>15% of annual salary</td>
</tr>
<tr>
<td>3</td>
<td>10% of annual salary</td>
</tr>
<tr>
<td>4</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 4-5 Employee ratings and bonuses

1. Open the file named EmployeeBonus2.vb using Notepad or the text editor of your choice.
2. Variables have been declared for you, and the input statements and output statements have been written. Read them over carefully before you proceed to the next step.
3. Design the logic and write the rest of the program using a Select Case statement.
4. Compile the program.
5. Execute the program entering the following as input:
   Employee's name: Jeanne Hanson
   Employee's salary: 70000
   Employee's performance rating: 2
6. Confirm that your output matches the following:
   Employee Name: Jeanne Hanson
   Employee Salary: $70000
   Employee Rating: 2
   Employee Bonus: $10500
Using Decision Statements to Make Multiple Comparisons

When you write programs, you must often write statements that include multiple comparisons. For example, you may want to determine that two conditions are True before you decide which path your program will take. In the following sections, you learn how to implement AND logic in a program by using the AND logical operator. You also learn how to implement OR logic using the OR logical operator.

Using AND Logic

When you write Visual Basic programs, you can use the AND operator (And) to make multiple comparisons in a single decision statement. Remember that when using AND logic, all expressions must evaluate to True for the entire expression to be True.

The Visual Basic code that follows illustrates a decision statement that uses the AND operator (And) to implement AND logic:

```vbnet
Dim MedicalPlan As String = "Y"
Dim DentalPlan As String = "Y"
If MedicalPlan = "Y" And DentalPlan = "Y" Then
    System.Console.WriteLine("Employee has medical insurance" & 
                           " and also has dental insurance.")
Else
    System.Console.WriteLine("Employee may have medical insurance " & 
                           "or may have dental insurance, but does " & 
                           "not have both medical and dental " & 
                           "insurance.")
End If
```

In this example, the variables named MedicalPlan and DentalPlan have both been initialized to the string constant "Y". When the expression MedicalPlan = "Y" is evaluated, the result is True. When the expression DentalPlan = "Y" is evaluated, the result is also True. Because both expressions evaluate to True, the entire expression MedicalPlan = "Y" And DentalPlan = "Y" evaluates to True. Because the entire expression is True, the output generated is "Employee has medical insurance and also has dental insurance."

If you initialize either of the variables MedicalPlan or DentalPlan with a value other than "Y", then the expression MedicalPlan = "Y" And DentalPlan = "Y" evaluates to False, and the output generated is "Employee may have medical insurance or may have dental insurance, but does not have both medical and dental insurance."

Using OR Logic

You can use OR logic when you want to make multiple comparisons in a single decision statement. Of course, you must remember when using OR logic that only one expression must evaluate to True for the entire expression to be True.
Using Decision Statements to Make Multiple Comparisons

The Visual Basic code that follows illustrates a decision statement that uses the OR operator (Or) to implement OR logic:

```vbnet
Dim MedicalPlan As String = "Y"
Dim DentalPlan As String = "N"
If MedicalPlan = "Y" Or DentalPlan = "Y"
    System.Console.WriteLine("
        "Employee has medical insurance" & _
        " or dental insurance or both.")
Else
    System.Console.WriteLine("
        "Employee does not have medical insurance and also does not have dental insurance.")
End If
```

In this example, the variable named MedicalPlan is initialized to the string constant "Y", and the variable named DentalPlan is initialized to the string constant "N". When the expression MedicalPlan = "Y" is evaluated, the result is True. When the expression DentalPlan = "Y" is evaluated, the result is False. The expression MedicalPlan = "Y" Or DentalPlan = "Y" evaluates to True because when using OR logic, only one of the expressions must evaluate to True for the entire expression to be True. Because the entire expression is True, the output generated is "Employee has medical insurance or dental insurance or both."

If you initialize both of the variables MedicalPlan and DentalPlan to the string constant "N", then the expression MedicalPlan = "Y" Or DentalPlan = "Y" evaluates to False, and the output generated is "Employee does not have medical insurance and also does not have dental insurance."

Exercise 4-5: Making Multiple Comparisons in Decision Statements

In this exercise, you use what you have learned about OR logic. This example program was written for a marketing research firm that wants to determine if a customer prefers Coke or Coke Zero over some other drink. Study the following code and then answer Questions 1–4.

```vbnet
' CokeOrCokeZero.vb - This program determines if a customer prefers to drink Coke or Coke Zero or some other drink.
Option Explicit On
Option Strict On
Module CokeOrCokeZero
    Sub Main()
        Dim CustomerName As String = "Customer's name"
        Dim Drink As String = "" "Customer's favorite drink"
        ' Work done in the housekeeping() procedure
        CustomerName = InputBox$("Enter customer's name: ")
        Drink = InputBox$("Enter customer's drink preference: ")
        ' Work done in the detailLoop() procedure
        If Drink = "Coke" Or Drink = "Coke Zero" Then
            System.Console.WriteLine("Customer Name: " & _
                CustomerName)
            System.Console.WriteLine("Drink: " & Drink)
        End If
    End Sub
End Module
```
Else
    System.Console.WriteLine(CustomerName & " does not prefer Coke or Coke Zero.")
End If
End Sub
End Module

1. What is the exact output when this program executes if the customer's name is Chas Matson and the drink is Coke?

2. What is the exact output when this program executes if the customer's name is Chas Matson and the drink is Coke Zero?

3. What is the exact output from this program when
   If Drink = "Coke" Or Drink = "Coke Zero"
   is changed to
   If Drink = "Coke" And Drink = "Coke Zero"
   and the customer's name is still Chas Matson and the drink is still Coke?

4. What is the exact output from this program when
   If Drink = "Coke" Or Drink = "Coke Zero"
   is changed to
   If Drink = "Coke" Or Drink = "Coke Zero"
       Or Drink = "coke"
       Or Drink = "coke zero"
   and the customer's name is Chas Matson, and the drink is coke? What does this change allow a user to enter?

Lab 4-5: Making Multiple Comparisons in Decision Statements

In this lab, you complete a partially written Visual Basic program for an airline that offers a 25 percent discount to passengers who are 6 years old or younger and the same discount to
passengers who are 65 years old or older. The program should request a passenger’s name and age, and then print whether the passenger is eligible or not eligible for a discount.

1. Open the file named Airline.vb using Notepad or the text editor of your choice.
2. Variables have been declared and initialized for you, and the input statements have been written. Read them carefully before you proceed to the next step.
3. Design the logic, deciding whether to use AND or OR logic. Write the decision statement to identify when a discount should be offered and when a discount should not be offered.
4. Be sure to include output statements telling whether or not the customer is eligible for a discount.
5. Compile the program.
6. Execute the program, entering the following as input:
   a. Customer Name: Will Moriarty
      Customer Age: 11
      What is the output?

   b. Customer Name: James Chung
      Customer Age: 64
      What is the output?

   c. Customer Name: Darlene Sanchez
      Customer Age: 75
      What is the output?

   d. Customer Name: Ray Sanchez
      Customer Age: 60
      What is the output?

   e. Customer Name: Tommy Sanchez
      Customer Age: 6
      What is the output?

   f. Customer Name: Amy Patel
      Customer Age: 8
      What is the output?