4.25 The following statements use conditional expressions. Rewrite each with an `if/else` statement.

A) `j = k > 90 ? 57 : 12;`

B) `factor = x >= 10 ? y * 22 : y * 35;`

C) `total += count == 1 ? sales : count * sales;`

D) `cout << (((num % 2) == 0) ? "Even\n" : "Odd\n");`

4.26 What will the following program display?

```
#include <iostream>
using namespace std;

int main()
{
   const int UPPER = 8, LOWER = 2;
   int num1, num2, num3 = 12, num4 = 3;

   num1 = num3 < num4 ? UPPER : LOWER;
   num2 = num4 > UPPER ? num3 : LOWER;
   cout << num1 << " " <<  num2 << endl;
   return 0;
}
```

## 4.14 The `switch` Statement

**CONCEPT:** The **`switch`** statement lets the value of a variable or expression determine where the program will branch.

A branch occurs when one part of a program causes another part to execute. The `if/else if` statement allows your program to branch into one of several possible paths. It performs a series of tests (usually relational) and branches when one of these tests is true. The `switch` statement is a similar mechanism. It, however, tests the value of an integer expression and then uses that value to determine which set of statements to branch to. Here is the format of the `switch` statement:

```
switch (IntegerExpression)
{
   case ConstantExpression:
        // place one or more
        // statements here

   case ConstantExpression:
        // place one or more
        // statements here

   // case statements may be repeated as many
   // times as necessary

   default:
        // place one or more
        // statements here
}
```

The first line of the statement starts with the word `switch`, followed by an integer expression inside parentheses. This can be either of the following:

- a variable of any of the integer data types (including `char`)
- an expression whose value is of any of the integer data types

On the next line is the beginning of a block containing several `case` statements. Each `case` statement is formatted in the following manner:

```
case ConstantExpression:
     // place one or more
     // statements here
```

After the word `case` is a constant expression (which must be of an integer type), followed by a colon. The constant expression may be an integer literal or an integer named constant. The `case` statement marks the beginning of a section of statements. The program branches to these statements if the value of the `switch` expression matches that of the `case` expression.

**WARNING!** The expression of each `case` statement in the block must be unique.

**NOTE:** The expression following the word `case` must be an integer literal or constant. It cannot be a variable, and it cannot be an expression such as `x < 22` or `n == 50`.

An optional `default` section comes after all the `case` statements. The program branches to this section if none of the `case` expressions match the `switch` expression. So, it functions like a trailing `else` in an `if/else if` statement.

Program 4-23 shows how a simple `switch` statement works.

**Program 4-23**

```
1  // The switch statement in this program tells the user something
2  // he or she already knows: the data just entered!
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char choice;
9
10    cout << "Enter A, B, or C: ";
11    cin >> choice;
12    switch (choice)
13    {
14       case 'A': cout << "You entered A.\n";
15                 break;
16       case 'B': cout << "You entered B.\n";
17                 break;
18       case 'C': cout << "You entered C.\n";
19                 break;
20       default:  cout << "You did not enter A, B, or C!\n";
21    }
22    return 0;
23 }
```

(program continues)

**Program 4-23** *(continued)*

**Program Output with Example Input Shown in Bold**

```
Enter A, B, or C: B [Enter]
You entered B.
```

**Program Output with Example Input Shown in Bold**

```
Enter A, B, or C: F [Enter]
You did not enter A, B, or C!
```

The first `case` statement is `case 'A':`, the second is `case 'B':`, and the third is `case 'C':`. These statements mark where the program is to branch to if the variable `choice` contains the values `'A'`, `'B'`, or `'C'`. (Remember, character variables and literals are considered integers.) The `default` section is branched to if the user enters anything other than A, B, or C.

Notice the `break` statements that are in the `case 'A'`, `case 'B'`, and `case 'C'` sections.

```
switch (choice)
{
   case 'A': cout << "You entered A.\n";
             break;  <----------
   case 'B': cout << "You entered B.\n";
             break;  <----------
   case 'C': cout << "You entered C.\n";
             break;  <----------
   default:  cout << "You did not enter A, B, or C!\n";
}
```

The `case` statements show the program where to start executing in the block and the `break` statements show the program where to stop. Without the `break` statements, the program would execute all of the lines from the matching `case` statement to the end of the block.

**NOTE:** The `default` section (or the last `case` section, if there is no `default`) does not need a `break` statement. Some programmers prefer to put one there anyway, for consistency.

Program 4-24 is a modification of Program 4-23, without the `break` statements.

**Program 4-24**

```
1  // The switch statement in this program tells the user something
2  // he or she already knows: the data just entered!
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char choice;
9
```

```
10      cout << "Enter A, B, or C: ";
11      cin >> choice;
12      // The following switch is
13      // missing its break statements!
14      switch (choice)
15      {
16         case 'A': cout << "You entered A.\n";
17         case 'B': cout << "You entered B.\n";
18         case 'C': cout << "You entered C.\n";
19         default:  cout << "You did not enter A, B, or C!\n";
20      }
21      return 0;
22   }
```

**Program Output with Example Input Shown in Bold**

```
Enter A, B, or C: A [Enter]
You entered A.
You entered B.
You entered C.
You did not enter A, B, or C!
```

**Program Output with Example Input Shown in Bold**

```
Enter A, B, or C: C [Enter]
You entered C.
You did not enter A, B, or C!
```

Without the `break` statement, the program "falls through" all of the statements below the one with the matching `case` expression. Sometimes this is what you want. Program 4-25 lists the features of three TV models a customer may choose from. The Model 100 has remote control. The Model 200 has remote control and stereo sound. The Model 300 has remote control, stereo sound, and picture-in-a-picture capability. The program uses a `switch` statement with carefully omitted `break`s to print the features of the selected model.

**Program 4-25**

```
1  // This program is carefully constructed to use the "fall through"
2  // feature of the switch statement.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int modelNum;  // Model number
9
10    // Get a model number from the user.
11    cout << "Our TVs come in three models:\n";
12    cout << "The 100, 200, and 300. Which do you want? ";
13    cin >> modelNum;
14
```

*(program continues)*

**Program 4-25** *(continued)*

```
15      // Display the model's features.
16      cout << "That model has the following features:\n";
17      switch (modelNum)
18      {
19         case 300: cout << "\tPicture-in-a-picture.\n";
20         case 200: cout << "\tStereo sound.\n";
21         case 100: cout << "\tRemote control.\n";
22                   break;
23         default:  cout << "You can only choose the 100,";
24                   cout << "200, or 300.\n";
25      }
26      return 0;
27   }
```

**Program Output with Example Input Shown in Bold**

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 100 [Enter]
That model has the following features:
   Remote control.
```

**Program Output with Example Input Shown in Bold**

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 200 [Enter]
That model has the following features:
   Stereo sound.
   Remote control.
```

**Program Output with Example Input Shown in Bold**

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 300 [Enter]
That model has the following features:
   Picture-in-a-picture.
   Stereo sound.
   Remote control.
```

**Program Output with Example Input Shown in Bold**

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 500 [Enter]
That model has the following features:
You can only choose the 100, 200, or 300.
```

Another example of how useful this "fall through" capability can be is when you want the program to branch to the same set of statements for multiple `case` expressions. For instance, Program 4-26 asks the user to select a grade of pet food. The available choices are A, B, and C. The `switch` statement will recognize either upper or lowercase letters.

**Program 4-26**

```
1   // The switch statement in this program uses the "fall through"
2   // feature to catch both uppercase and lowercase letters entered
3   // by the user.
4   #include <iostream>
5   using namespace std;
6
```

```
 7  int main()
 8  {
 9     char feedGrade;
10
11     // Get the desired grade of feed.
12     cout << "Our pet food is available in three grades:\n";
13     cout << "A, B, and C. Which do you want pricing for? ";
14     cin >> feedGrade;
15
16     // Display the price.
17     switch(feedGrade)
18     {
19        case 'a':
20        case 'A': cout << "30 cents per pound.\n";
21                  break;
22        case 'b':
23        case 'B': cout << "20 cents per pound.\n";
24                  break;
25        case 'c':
26        case 'C': cout << "15 cents per pound.\n";
27                  break;
28        default:  cout << "That is an invalid choice.\n";
29     }
30     return 0;
31  }
```

**Program Output with Example Input Shown in Bold**

```
Our pet food is available in three grades:
A, B, and C. Which do you want pricing for? b [Enter]
20 cents per pound.
```

**Program Output with Example Input Shown in Bold**

```
Our pet food is available in three grades:
A, B, and C. Which do you want pricing for? B [Enter]
20 cents per pound.
```

When the user enters `'a'` the corresponding `case` has no statements associated with it, so the program falls through to the next `case`, which corresponds with `'A'`.

```
case 'a':
case 'A':  cout << "30 cents per pound.\n";
           break;
```

The same technique is used for `'b'` and `'c'`.

## Using `switch` in Menu Systems

The `switch` statement is a natural mechanism for building menu systems. Recall that Program 4-18 gives a menu to select which health club package the user wishes to purchase. The program uses `if/else if` statements to determine which package the user has selected and displays the calculated charges. Program 4-27 is a modification of that program, using a `switch` statement instead of `if/else if`.

**Program 4-27**

```
// This program uses a switch statement to determine
// the item selected from a menu.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int choice;       // To hold a menu choice
   int months;       // To hold the number of months
   double charges;   // To hold the monthly charges

   // Constants for membership rates
   const double ADULT = 40.0,
                CHILD = 20.0,
                SENIOR = 30.0;

   // Constants for menu choices
   const int ADULT_CHOICE = 1,
             CHILD_CHOICE = 2,
             SENIOR_CHOICE = 3,
             QUIT_CHOICE = 4;

   // Display the menu and get a choice.
   cout << "\t\tHealth Club Membership Menu\n\n"
        << "1.  Standard Adult Membership\n"
        << "2.  Child Membership\n"
        << "3.  Senior Citizen Membership\n"
        << "4.  Quit the Program\n\n"
        << "Enter your choice: ";
   cin >> choice;

   // Set the numeric output formatting.
   cout << fixed << showpoint << setprecision(2);

   // Respond to the user's menu selection.
   switch (choice)
   {
      case ADULT_CHOICE:
         cout << "For how many months? ";
         cin >> months;
         charges = months * ADULT;
         cout << "The total charges are $" << charges << endl;
         break;

      case CHILD_CHOICE:
         cout << "For how many months? ";
         cin >> months;
         charges = months * CHILD;
         cout << "The total charges are $" << charges << endl;
         break;

```

```
53          case SENIOR_CHOICE:
54             cout << "For how many months? ";
55             cin >> months;
56             charges = months * SENIOR;
57             cout << "The total charges are $" << charges << endl;
58             break;
59
60          case QUIT_CHOICE:
61             cout << "Program ending.\n";
62             break;
63
64          default:
65             cout << "The valid choices are 1 through 4. Run the\n"
66                  << "program again and select one of those.\n";
67       }
68
69       return 0;
70    }
```

**Program Output with Example Input Shown in Bold**

```
          Health Club Membership Menu

1. Standard Adult Membership
2. Child Membership
3. Senior Citizen Membership
4. Quit the Program

Enter your choice: 2 [Enter]
For how many months? 6 [Enter]
The total charges are $120.00
```

## Checkpoint

myprogramminglab *www.myprogramminglab.com*

4.27 Explain why you cannot convert the following `if/else if` statement into a `switch` statement.

```
if (temp == 100)
   x = 0;
else if (population > 1000)
   x = 1;
else if  (rate < .1)
   x = -1;
```

4.28 What is wrong with the following `switch` statement?

```
switch (temp)
{
   case temp < 0 : cout << "Temp is negative.\n";
                   break;
   case temp == 0: cout << "Temp is zero.\n";
                   break;
   case temp > 0 : cout << "Temp is positive.\n";
                   break;
}
```