The `if` Statement and Practice Problems

The Simple `if` Statement

Use

To specify the conditions under which a statement or group of statements should be executed.

Form

```
        if (boolean-expression)
            statement;
```

where `if` is a reserved word, `boolean-expression` is an expression that evaluates to `true` or `false`, and `statement` is a C++ statement or a group of statements enclosed in curly braces (a compound statement).

Action

If the boolean expression is `true`, the specified statement is executed; otherwise it is not. In either case, execution continues with the next statement in the program.

Examples

```
   // grade is a char
   if (grade == 'A')
      cout << "EXCELLENT WORK!" << endl;

---------------------------------------------------------------------------

   // yards is an int
   if (yards != 0)
      cout << "There were " << yards << " yards." << endl;

---------------------------------------------------------------------------

   // temperature, normalTemp and degreesOfFever are doubles;
   // hasFever is a bool
   if (temperature > normalTemp) {
      degreesOfFever = temperature - normalTemp;
      hasFever = true;
   }
```

The `if-else` Statement

Use

To choose exactly one out of two statements (possibly compound statements) to be executed; specifies the conditions under which the first statement is to be executed and provides an alternative statement to execute if these conditions are not met.

Form

```
        if (boolean-expression)
            statement-1;
        else
            statement-2;
```

where `if` and `else` are reserved words, `boolean-expression` is an expression that evaluates to `true` or `false`, and `statement-1` and `statement-2` are C++ statements (possibly compound statements, i.e. a group of statements enclosed by curly braces).

Action

If the boolean expression is `true`, `statement-1` is executed and `statement-2` is skipped; otherwise `statement-1` is skipped and `statement-2` is executed. In either case, execution continues with the next statement in the program.

Examples

```
  // numItems is an int; averageCost and totalCost are doubles
  if (numItems >= 0)
     averageCost = totalCost / numItems;
  else
     cout << "No items were purchased." << endl;
```

```
----------------------------------------------------------------------
```

```
  const double POLLUTION_CUTOFF = 3.5;
  // pollutionIndexis a double
  if (pollutionIndex < POLLUTION_CUTOFF)
     cout << "Safe Condition" << endl;
  else
     cout << "Hazardous Condition" << endl;
```

The Extended-`if` Statement

Use

To choose one statement (possibly compound) to be executed from among a group of statements (possibly compound); specifies the conditions under which each statement may be executed and may contain a default statement (in an `else` clause at the end) to be executed if *none* of these conditions are met. Note that in the absence of a final `else` clause, it may be the case that *none* of the statements are executed.

Form

```
if (boolean-expression-1)
     statement-1;
else if (boolean-expression-2)
     statement-2;
          .
          .
          .
else if (boolean-expression-n)
     statement-n;
else
     statement-default;
```

where `if` and `else` are reserved words, `boolean-expression-1`, `boolean-expression-2`, . . ., `boolean-expression-n` are expressions that evaluate to `true` or `false`, and `statement-1`, `statement-2`, . . ., `statement-n`, and `statement-default` are C++ statements, possibly compound statements. (A compound statement is a group of statements enclosed by curly braces.)

Action

The boolean expressions are evaluated in the order of their appearance to determine the first expression that is `true`. The associated statement is executed, and execution continues with the first statement following the entire `if-else-if` construct. If none of the boolean expressions is `true`, the statement associated with the `else` clause is executed, and execution then continues with the statement following the construct. If none of the boolean expressions is `true` and the `else` clause is omitted, execution "falls through" to (continues with) the next statement in the program after the construct.

Examples

(next page)

Examples

```
// score is a double; grade is a char

if (score == 100) {
   grade = 'A';
   cout << "Superb" << endl;
}
else if (score >= 90) {
   grade = 'A';
   cout << "Excellent" << endl;
}
else if (score >= 80) {
   grade = 'B';
   cout << "Very Good" << endl;
}
else if (score >= 70) {
   grade = 'C';
   cout << "Good" << endl;
}
else if (score >= 60)
   grade = 'D';
else
   grade = 'F';
```

```
// Ch is a char

if ((Ch >= 'a') && (Ch <= 'z')) {
   // code to process lower-case leter
}
else if ((Ch >= 'A') && (Ch <= 'Z')) {
   // code to process upper-case leter
}
else if ((Ch >= '0') && (Ch <= '9')) {
   // code to process digit character
}
else {
   // code to process non-letter/non-digit characters
}
```

Practice Problems

- What is wrong with the following `if` statement (there are at least 3 errors). The indentation indicates the desired behavior.

```
if numNeighbors >= 3 || numNeighbors = 4
   ++numNeighbors;
   cout << "You are dead!" << endl;
else
   --numNeighbors;
```

- Describe the output produced by this poorly indented program segment:

```
int number = 4;
double alpha = -1.0;
if (number > 0)
   if (alpha > 0)
      cout << "Here I am!" << endl;
else
   cout << "No, I’m here!" << endl;
cout << "No, actually, I’m here!" << endl;
```

- Consider the following `if` statement, where `doesSignificantWork`, `makesBreakthrough`, and `nobelPrizeCandidate` are all boolean variables:

```
if (doesSignificantWork) {
   if (makesBreakthrough)
      nobelPrizeCandidate = true;
   else
      nobelPrizeCandidate = false;
}
else if (!doesSignificantWork)
   nobelPrizeCandidate = false;
```

  First, write a simpler `if` statement that is equivalent to this one. Then write a single assignment statement that does the same thing.

- Write `if` statements to do the following:
  - If character variable `taxCode` is ’T’, increase price by adding the `taxRate` percentage of `price` to it.
  - If integer variable `opCode` has the value 1, read in double values for `X` and `Y` and calculate and print their sum.

- If integer variable `currentNumber` is odd, change its value so that it is now 3 times `currentNumber` plus 1, otherwise change its value so that it is now half of `currentNumber` (rounded down when `currentNumber` is odd).
- Assign `true` to the boolean variable `leapYear` if the integer variable `year` is a leap year. (A leap year is a multiple of 4, and if it is a multiple of 100, it must also be a multiple of 400.)
- Assign a value to double variable `cost` depending on the value of integer variable `distance` as follows:

Distance	Cost
0 through 100	5.00
More than 100 but not more than 500	8.00
More than 500 but less than 1,000	10.00
1,000 or more	12.00