LAB 7: NESTED IF...ELSE STATEMENTS

Nested **if...else** statements test for multiple cases by placing if...else selection statements inside other if...else selection statements. For example, the following pseudocode if...else statement prints A for exam grades greater than or equal to 90, B for grades in the range 80 to 89, C for grades in the range 70 to 79, D for grades in the range 60 to 69 and F for all other grades:

```
If student's grade is greater than or equal to 90
  Print "A"
Else
  If student's grade is greater than or equal to 80
    Print "B"
  Else
    If student's grade is greater than or equal to 70
      Print "C"
   Else
      If student's grade is greater than or equal to 60
        Print "D"
      Else
        Print "F"
```

This pseudocode can be written in C++ as

```
     void main(){
       if ( studentGrade >= 90 ) // 90 and above gets "A"
         cout << "A";
       else
         if ( studentGrade >= 80 ) // 80-89 gets "B"
           cout << "B";
         else
           if ( studentGrade >= 70 ) // 70-79 gets "C"
             cout << "C";
           else
             if ( studentGrade >= 60 ) // 60-69 gets "D"
               cout << "D";
             else // less than 60 gets "F"
               cout << "F";
     }
```

Tip: A nested if...else statement can perform much faster than a series of single-selection if statements because of the possibility of early exit after one of the conditions is satisfied.

EXERCISE 1:

Write a program that read an integer and write "Positive" if the number is greater than zero, Print Negative if the number is smaller than zero and print "Zero" if the number equal to zero. (Use Nested If statement).

DANGLING-ELSE PROBLEM

The C++ compiler always associates an else with the immediately preceding if unless told to do otherwise by the placement of braces ({ and }). This behavior can lead to what is referred to as the **dangling-else problem**.

EXERCISE 2:

Write the flowing program and test it with values in the table, after each execution find the output and fill it in the table.

```
void main(){
  int x, y;
  cin << x << y;
  if ( x > 0 )
    if ( y > 0 )
      cout << "x and y are >5";
  else
  cout << "x is <=5";
}
```

x	**y**	**Output**
8	7	
3	8	
13	1	
3	4	

You will find if x is greater than 5, the nested if statement determines whether y is also greater than 5. If so, "x and y are > 5" is output. Otherwise, it appears that if x is not greater than 5, the else part of the if...else outputs "x is <= 5", so that the program will not print the message "x is <= 5" if both x and y is less than 5.

To force the nested if...else statement to execute as it was originally intended, we must write it as follows:

```
 void main(){
   int x, y;
   cin << x << y;
   if ( x > 0 )
   {
     if ( y > 0 ) cout << "x and y are >5";
   }
   else
   cout << "x is <=5";
 }
```

EXERCISE 3:

State the output for each of the following when x is 9 and y is 11 and when x is 11 and y is 9.

a.

```
if ( x < 10 )
  if ( y > 10 )
    cout << "*****" << endl;
  else
    cout << "#####" << endl;
cout << "$$$$$" << endl;
```

b.

```
if ( x < 10 )
{
  if ( y > 10 ) cout << "*****" << endl;
}
else
{
  cout << "#####" << endl;
  cout << "$$$$$" << endl;
}
```

ASSIGNMENT OPERATOR

C++ provides several assignment operator for abbreviating assignment expressions. For example, the statement

c = c + 3;

can be abbreviated with the **addition assignment operator +=** as c += 3;

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. Any statement of the form

variable = variable operator expression;

in which the same variable appears on both sides of the assignment operator and operator is one of the binary operators +, -, *, /, or % (or others we'll discuss later in the text), can be written in the form

variable operator= expression;

Thus the assignment c += 3 adds 3 to c. the following table shows the arithmetic assignment operator, sample expressions using these operators and explanations.

Assignment operator	**Sample expression**	**Explanation**	**Assigns**
`Assume: int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to c
`-=`	`d -= 4`	`d = d - 4`	1 to d
`*=`	`e *= 5`	`e = e * 5`	20 to e
`/=`	`f /= 3`	`f = f / 3`	2 to f
`%=`	`g %= 9`	`g = g % 9`	3 to g

EXERCISE 4:

Write the following program, trace it and find final results of each variable.

```
void main(){
int a=4, b=10, c=2, d=12;
a += b-2;
b -= b%a+2;
c *= a*b;
d %= b;
cout <<"a="<<a<<"\nb="<<b<<"\nc="<<c<<"\nd="<<d;
}
```