C++ Primer Plus, 5^{th} *Edition* by Stephen Prata
Chapter 6: Branching Statements and Logical Operators
Review Questions

1. Consider the following two code fragments for counting spaces and newlines:

```
// Version 1
while (cin.get(ch))      // quit on eof
{
      if(ch == ' ')
            spaces++;
      if (ch == '\n')
            newlines++;
}

// Version 2
while (cin.get(ch))      // quit on eof
{
      if(ch == ' ')
            spaces++;
      else if (ch == '\n')
            newlines++;
}
```

What advantages, if any, does the second form have over the first?

The second code fragment is more efficient. The logical expressions in the two if statements of both code fragments are mutually exclusive. In the first code fragment, the second logical expression in the body of the while loop is evaluated even if the first evaluates to true, which is unnecessary. In the second code fragment, the second logical expression in the while loop is evaluated only if the first one is false.

2. In Listing 6.2, what is the effect of replacing `++ch` with `ch+1`?

`cout` *will print the character code rather than the actual character. Since* `++ch` *is of type char,* `cout` *will print a character. However, the expression* `ch+1` *is of type* `int`*, so* `cout` *will print a number. In the expression* `ch+1`*, the* `ch` *variable is promoted to type* `int`*. The answer remains in* `int` *form.*

3. Carefully consider the following program:

```
#include <iostream>
using namespace std;
int main()
{
      char ch;
      int ct1, ct2;

      ct1 = ct2 = 0;
      while ((ch = cin.get()) != '$')
      {
            cout << ch;
            ct1++;
            if (ch = '$')
                  ct2++;
            cout << ch;
      }
      cout << "ct1 = " << ct1 << ", ct2 = " << ct2 << "\n";
      return 0;
}
```

Suppose you provide the following input, where `<Enter>` represents pressing Enter:

```
Hi!<Enter>
Send $10 or $20 now! <Enter>
```

What is the output? (Recall that input is buffered.)

The output is the following:

```
H$i$!$
$S$e$n$d$ $ct1 = 9, ct2 = 9
```

4. Construct logical expressions to represent the following conditions:

   a. `weight` is greater than or equal to 115 but less than 125

   See the following code:

   `(weight >= 115) && (weight < 125)`

   b. `ch` is `q` or `Q`

   See the following code:

   `(ch == 'q') || (ch == 'Q')`

   c. `x` is even but is not `26`

   See the following code:

   `(x % 2 == 0) && (x != 26)`

   d. `x` is even but is not a multiple of `26`

   See the following code:

   `(x % 2 == 0) && (x % 26 != 0)`

e. `donation` is in the range 1,000-2,000 or `guest` is `1`

See the following code:

```
(donation <= 2000 && donation >= 1000) || (guest == 1)
```

f. `ch` is a lowercase letter or an upper case letter. (Assume that lowercase letters are coded sequentially and that uppercase letters are coded sequentially but that there is a gap in the code between uppercase and lowercase.)

See the following code:

```
(ch <= 'z' && ch >= 'a') || (ch <= 'Z' && ch >= 'A')
```

5. In English, the statement "I will not not speak" means the same as "I will speak." In C++, is `!!x` the same as `x`?

   Yes. `!!x` *is equivalent to* `!(!x)` *which is logially equivalent to* `x`.

6. Construct a conditional expression that is equal to the absolute value of a variable. That is, if a variable `x` is positive, the value of the expression is just `x`, but if `x` is negative, the value of the expression is `-x`, which is positive.

   Consider the following code:

```
x = (x > 0) ? x : -x;
```

7. Rewrite the following fragment using `switch`:

```
if (ch == 'A')
      a_grade++;
else if (ch == 'B')
      b_grade++;
else if (ch == 'C')
      c_grade++;
else if (ch == 'D')
      d_grade++;
else
      f_grade++;
```

Consider the following code:

```
switch (ch)
{
    case 'A':
        a_grade++;
        break;
    case 'B':
        b_grade++;
        break;
    case 'C':
        c_grade++;
        break;
    case 'D':
        d_grade++;
        break;
    default:
        f_grade++;
}
```

8. In Listing 6.10, what advantage would there be in using character labels, such as `a` and `c`, instead of numbers for the menu choices and switch cases? (Hint: Think about what happens if the user types `q` in either case and what happens if the user types `5` in either case.)

   Well, by using character labels the exit condition could be `e` *or* `q`*, so typing either one would be a valid exit condition.*

9. Consider the following code fragment:

```
int line = 0;
char ch;
while (cin.get(ch))
{
      if (ch == 'Q')
            break;
      if (ch != '\n')
            continue;
      line++;
}
```

   Rewrite this without using `break` or `continue`.

   See the following code:

```
int line = 0;
char ch;
while (cin.get(ch) != 'Q' && cin.good())
{
    if (ch == '\n')
        line++;
}
```