3.4 Equality and relational operators

Equality operators

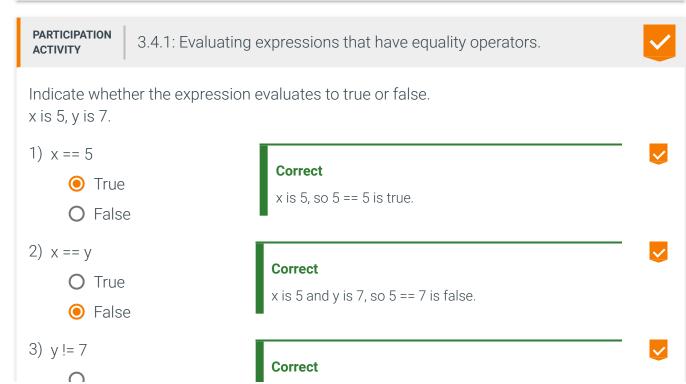
An **equality operator** checks whether two operands' values are the same (==) or different (!=). Note that equality is ==, not just =.

An expression involving an equality operator evaluates to a Boolean value. A **Boolean** is a type that has just two values: true or false.

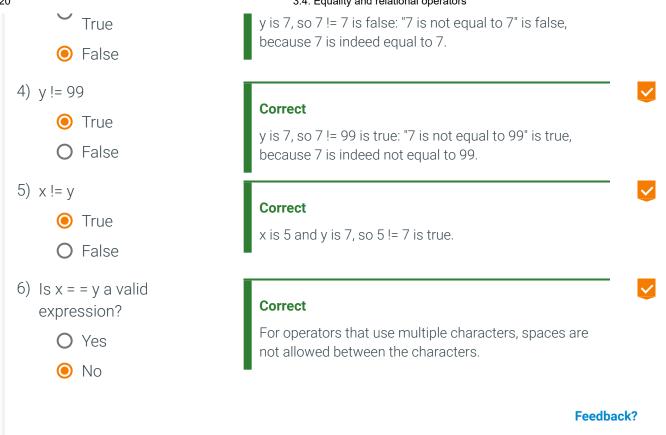
Table 3.4.1: Equality operators.

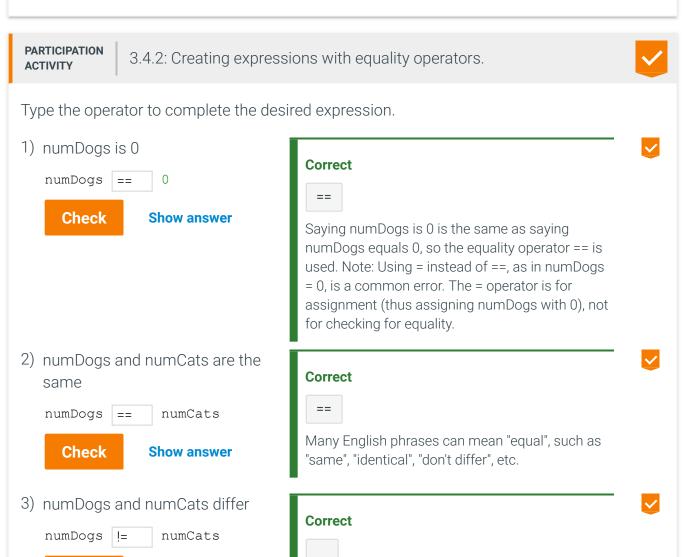
Equality operators	Description	Example (assume x is 3)
==	a == b means a is equal to b	x == 3 is true x == 4 is false
!=	a != b means a is not equal to b	x != 3 is false x != 4 is true

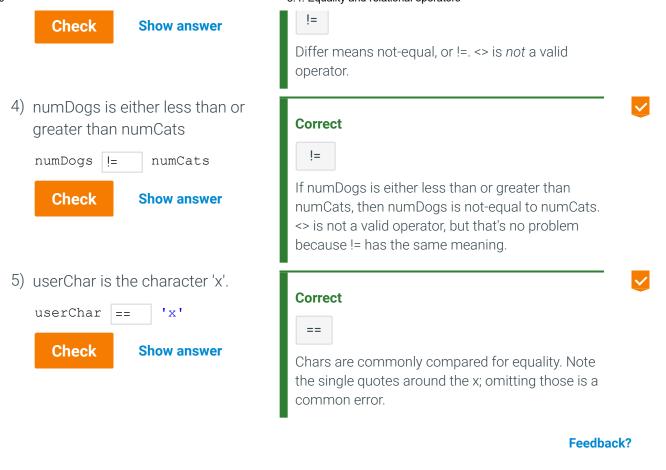
Feedback?



3.4. Equality and relational operators







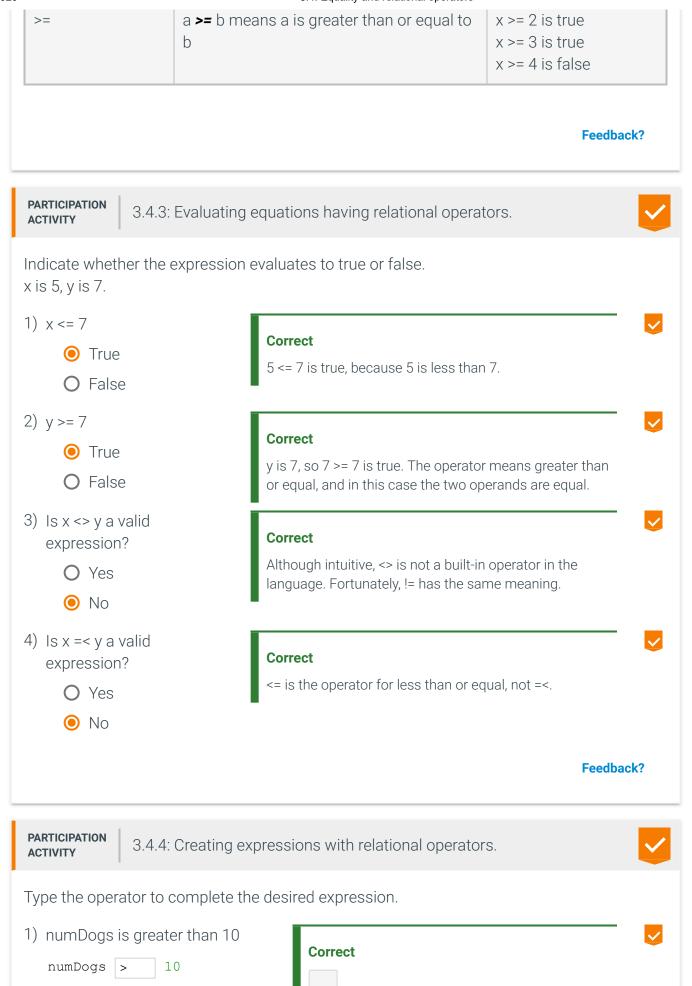
Relational operators

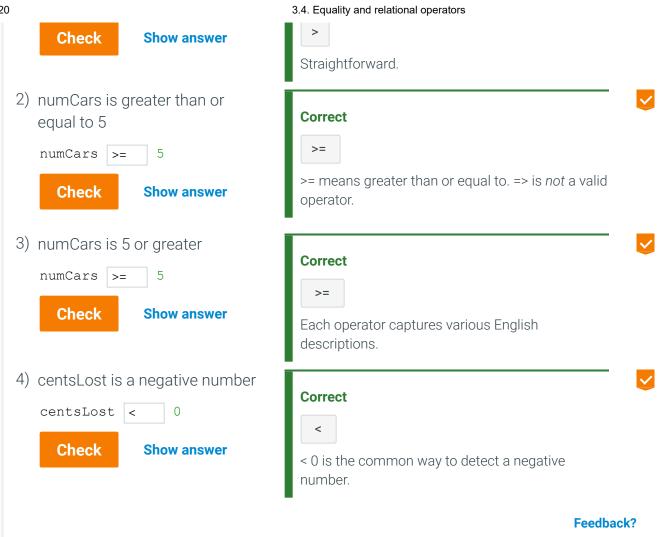
A **relational operator** checks how one operand's value relates to another, like being greater than.

Some operators like >= involve two characters. A programmer cannot arbitrarily combine the >, =, and < symbols; only the shown two-character sequences represent valid operators.

Table 3.4.2: Relational operators.

Relational operators	Description	Example (assume x is 3)
<	a < b means a is less than b	x < 4 is true x < 3 is false
>	a > b means a is greater than b	x > 2 is true x > 3 is false
<=	a <= b means a is less than or equal to b	x <= 4 is true x <= 3 is true x <= 2 is false





Example: Golf scores

In golf and miniature golf, each hole has a "par" indicating the normal number of strokes to get the ball in the hole. The following program outputs special names for numbers below par. The program uses one relational operator, and several equality operators, in a multi-branch if-else statement.

zyDE 3.4.1: Golf scores.

This program outputs special names for different numbers of strokes for a par 4 k Run to see the name for the input strokes of 2. Try changing the input strokes to c values, and press Run again to see the name.

If desired, try extending the program to print "Bogey" for 5, and "Double bogey" for Remember that equals is ==, not =.

Try typing "= 1" instead of "== 1" and see what happens. Try typing an unsupportelike =>, and see what happens.

```
2
7
       cin >> numStrokes;
8
9
       // Assumes "par 4"
10
       if (numStrokes <= 0) {</pre>
          cout << "Invalid entry." << endl;</pre>
11
12
                                                         Run
13
       else if (numStrokes == 1) {
          cout << "Hole in 1!!!" << endl;</pre>
14
                                                        Eagle!
15
       else if (numStrokes == 2) {
16
17
          cout << "Eagle!" << endl;</pre>
18
19
       else if (numStrokes == 3) {
20
          cout << "Birdie." << endl;</pre>
21
22
       else if (numStrokes == 4) {
23
          cout << "Par." << endl;</pre>
24
25
       else {
26
          cout << "Better luck next time." <<</pre>
27
```

Feedback?

CHALLENGE ACTIVITY

3.4.1: Enter the output for the branches with relational and equality operators.



Jump to level 1

Type the program's output.

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

int main() {
   int numApples;

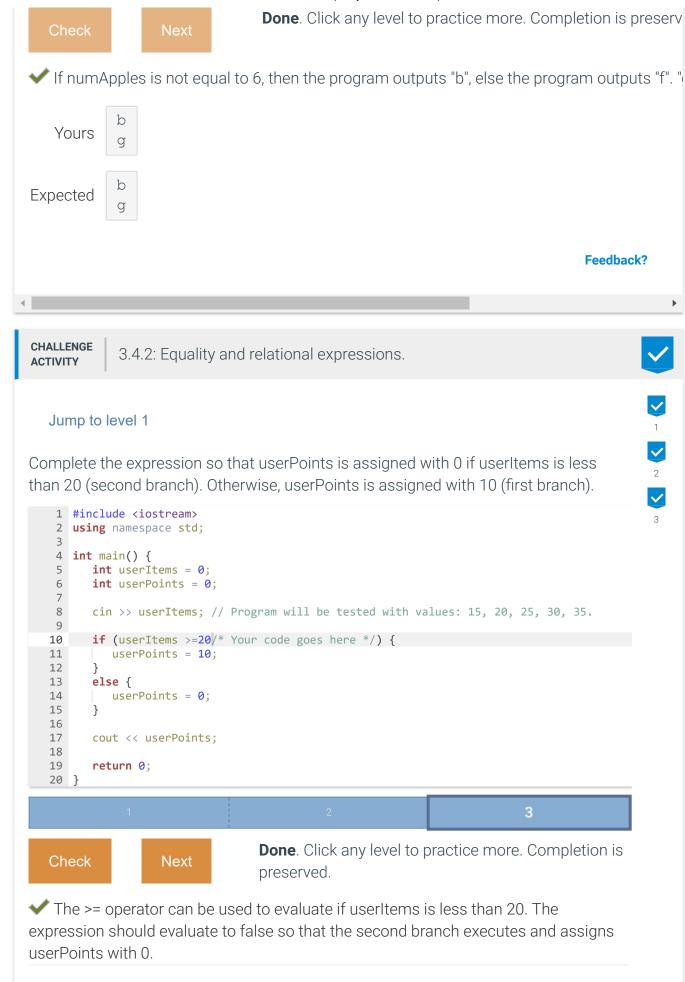
   numApples = 4;

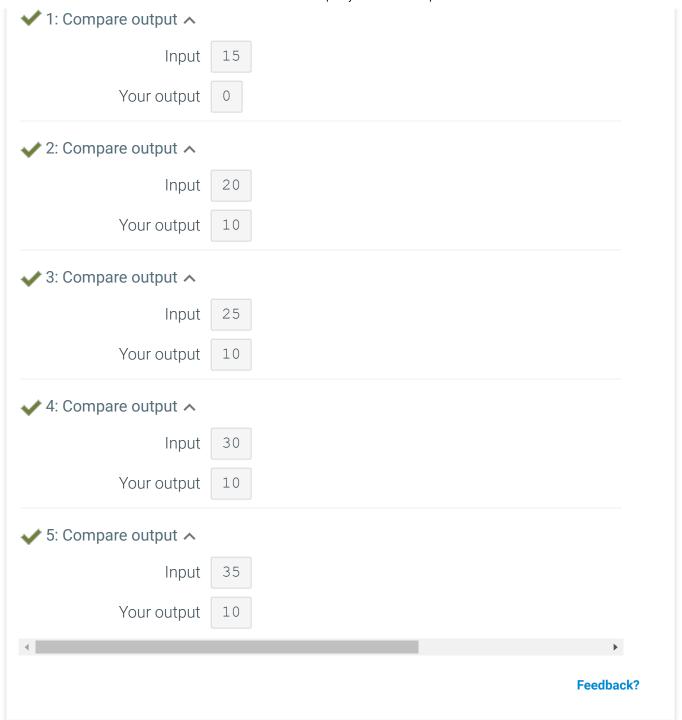
   if (numApples != 6) {
      cout << "b" << endl;
   }

   else {
      cout << "f" << endl;
   }

   cout << "g" << endl;
}</pre>
```

g





Comparing characters, strings, and floating-point types

The relational and equality operators work for integer, character, and floating-point built-in types.

Comparing characters compares their ASCII numerical encoding.

Floating-point types should not be compared using the equality operators, due to the imprecise representation of floating-point numbers, as discussed in a later section.

The operators can also be used for the string type. Strings are equal if they have the same number of characters and corresponding characters are identical. If string myStr = "Tuesday", then (myStr == "Tuesday") is true, while (myStr == "tuesday") is false because T differs from t.

20	5.4. Equality and relational operators	
PARTICIPATION ACTIVITY	3.4.5: Comparing various types.	
	rison will compile AND consistently yield expected results? Variables noted by their names.	
1) myInt == 4 O OK O Not 0		
2) myChar == O OK O Not 0		
3) myDouble O OK O Not (
4) myString = O OK O Not 0		
	Fee	edback?

Common errors

Perhaps the most <u>common error</u> in C and C++ is to use = rather than == in an if-else expression, as in: if (numDogs = 9) $\{ ... \}$. That code is not a syntax error. The statement assigns numDogs with 9, and then because that value is non-zero, the expression is considered true. C's designers allowed assignment in expressions to allow compact code, and use = for assignment rather than := or similar to save typing. Many people believe those language design decisions were mistakes, leading to many bugs. Some modern compilers provide a warning when = appears in an if-else expression.

Another <u>common error</u> is to use invalid character sequences like =>, !<, or <>, which are *not* valid operators.

PARTICIPATION 3.4.6: Watch out for assignment in an if-else expression.	
What is the final value of numltems?	
1)	

```
numItems = 3;
if (numItems == 3) {
    numItems = numItems + 1;
}

Check Show answer

2) numItems = 3;
if (numItems = 10) {
    numItems = numItems + 1;
}

Check Show answer

Feedback?
```

CHALLENGE ACTIVITY

3.4.3: If-else statement: Fix errors.



Jump to level 1

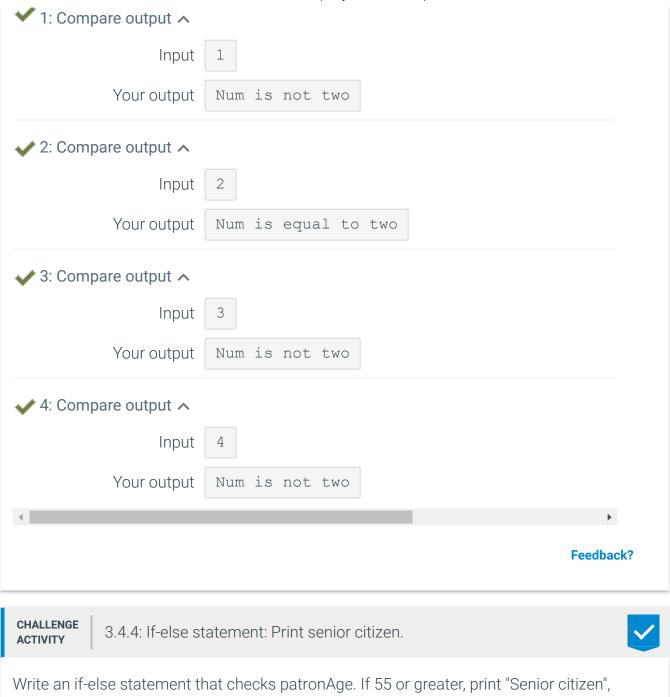
Find and fix the error in the if-else statement.

```
1 #include <iostream>
 2 using namespace std;
 3
4 int main() {
 5
      int userNum;
 6
7
      cin >> userNum; // Program will be tested with values: 1, 2, 3, 4.
 8
      if (userNum != 2) {
10
         cout << "Num is not two" << endl;</pre>
11
12
      else {
         cout << "Num is equal to two" << endl;</pre>
13
14
15
16
      return 0;
```

Check

Try again **Done**. Click any level to practice more. Completion is preserved.

✓ The if statement has to check if userNum is not 2.



```
otherwise print "Not senior citizen" (without quotes). End with newline.
    1 #include <iostream>
    2 using namespace std;
    3
    4 int main() {
    5
          int patronAge;
    6
    7
          cin >> patronAge;
    8
    9
          /* Your solution goes here */
   10
          if(patronAge>=55){
   11
             cout << "Senior citizen"<<endl;</pre>
   12
          }else{
             cout << "Not senior citizen"<< endl;</pre>
   13
   14
   15
```

