Exercise #2
Due Friday, 01/15/2016, at 11:59pm

Decisions

In class this week you’ll be introduced to if statements, the simplest structure used to allow your programs to make decisions. Since you’ll be using them in the rest of your time in CS16*, it’s important to be familiar with how they work in different situations.

The three questions below cover some usual and unusual uses of if statements. If you’re not sure what the answer to a question is, try creating a simple C++ program that uses the code referenced in the question to try it out!

1. Consider a single execution of the following if/else if/else block:

   ```cpp
   bool condition1, condition2;
   // Some code that sets condition1 and condition2
   if (condition1) {
       // Code block A
   } else if (condition2) {
       // Code block B
   } else {
       // Code block C
   }
   ```

   Is it possible for code blocks A and B to both run? If so, what boolean values should the variables condition1 and condition2 have to make that happen? If not, explain why not.

   What about code blocks A, B, and C? Explain your answer as you did for the previous question.

2. The code below will print out “3 is true, -10 is true.”. Even though if statements usually take boolean parameters, this shows that they can use integer values to make decisions as well!

   Based on this result, which integer values correspond to boolean ‘true’ (Hint: the answer isn’t just ‘3 and -10’)? Which integer values correspond to boolean ‘false’?

   ```cpp
   if (3) {
       cout << "3 is true, ";
   }
   if (0) {
       cout << "0 is true, ";
   }
   if (-10) {
       cout << "-10 is true, ";
   }
   ```

3. There’s a mistake in the code below– it will print out “x is equal to one!”, even though we assign the value 3 to the variable x right before the if block (Try it out yourself!). What is the mistake, and what should the author of this code do to fix it?

   ```cpp
   int x = 3;
   if (x = 1) {
       cout << "x is equal to one!" << endl;
   } else {
       cout << "x is something other than one." << endl;
   }
   ```
Assignment 2 Design

A major part of each future assignment is developing a design and testing plan for your programs before you start programming them. Thinking beforehand about how your program will be structured and how it should behave for different inputs is crucial to make writing the program as pain-free as possible.

As the last part of this exercise, develop the testing table for your assignment 2 program. (Check out the ‘Testing’ section of the template design document if you need a reference!) Remember that your table should cover both valid and invalid inputs— and it’s perfectly acceptable (for now) if invalid inputs would cause your program to crash, malfunction, or just behave in a way you can’t predict!

For take-home exercises completed in peer-led groups, each student must participate in the class discussion and write answers to each of the questions on his/her own paper to show for credit.

For take-home exercises completed on your own, turn in your work electronically using the TEACH website.