

## Conditionals and Constants

In lab you used conditional statements to write the function to compute the conditional price for Minnie's golf course. Conditional execution is the essential feature that makes computers more than simply high speed calculators. This assignment will give you practice using boolean expressions in conditional statements.

There are three problems. As we did in lab, you will start by writing some test data. That gives you a goal. Then you will write the code to solve the problem. After compiling and testing and correcting your program, you will submit the programs and the test data. As we did in lab, your test data will be shared with other students. Therefore, the test data you submit may be used, anonymously, to test submissions of other students. And vice versa.

## Computerized Instant Testing

For this assignment, and for most of the rest of the term, your programs will be compiled and tested automatically by the *provide* program. That means you do not have to wait to know if your program is likely to be correct. Even using many test cases, you can never be sure you have handled all possible situations. But, by pooling test data from many people, we can check for a larger number of values than one person alone might devise.

**IMPORTANT: OUTPUT FORMAT MATTERS** Automated testing means you get feedback right away, and you also get testing from many more people, but there is one restriction. The way your program displays the results has to fit the requirements exactly.

The testing program will feed input into your program and compare the output to the expected result. If you add extra messages or put in extra blank lines, your output, even if numerically correct, will not match the expected output and will be considered incorrect. Therefore, please do not print helpful prompts or cheery output messages. Data values go in, a result comes out, and that is it.

## Start Soon: This Homework Takes Time

Make a directory on the Linux server for this assignment. Change into that directory. Then copy some starter files from our directory into the directory you make:

```
cp /comp/11/files/hw3/* .
```

There is a dot at the end of that command. The dot is the Linux term for "the current directory".

## Exercise 3-1: Largest of Three Numbers

The first problem is to write a program that reads in three numbers and prints the largest of those three. Here are the steps to follow to write your program:

(a) **Make Test Data:** Use an editor to edit the file called `max3.data` that contains test data. The file already contains two lines of test data. Each line is one test. The input is on the left, then there is a colon, then the correct output is on the right. Edit this file. Add two or more lines of test data. You can use these values to test your program, and the automated tester will also use this file.

(b) **Write the Code:** Edit the file called `max3.cpp` by filling in the details of the function called `max3` which is given three numbers as input and returns the largest value of those three. You will need to use some boolean expressions and one or more conditional statements.

(c) **Compile, Fix, ..., Test...** As you did in lab, you compile, fix syntax errors, compile again, and finally run the program and test it:

```
g++ -Wall -Wextra max3.cpp
./a.out
```

If the program does not work, trace the code by hand, show it to someone else, explain it to a friend or roommate. Come to office hours and ask.

## Exercise 3-2: Alphabetizing Three Words

The lab sign-in sheet lists names in alphabetical order; a program was used to do that. How does a computer put words in alphabetical order? You are about to figure that out. For this exercise, write a function that takes as arguments three strings and prints out the three words in alphabetical order on one line separated by spaces and ended with `endl`.

In C++ the comparison operators (`<`, `>`, `<=`, `>=`) can be used with strings to see if one word is before or after another word.

(a) **Make Test Data:** Use an editor to edit the file called `3wordsort.data` by adding two or more lines of test data. Again, each data set has two parts: input and output. The input goes before a colon, the output goes after it.

(b) **Write the Code:** Write the function that prints out the words in alphabetical order. Note that this function is declared to be of type `void` which means it does not return anything. This function *does* something, but does not return a value. Functions that print out data in a particular format are a typical example of a void function.

(c) **Compile, Fix, ..., Test...**

## Exercise 3-3: Checking for Valid Dates

In lab2 you used the Linux `cal` program to find calendars for various months and years. We saw that the program rejects years that are before 1 or after 9999. The program also rejects months that are before 1 or after 12. Those tests are easy to do. But what if you wanted a program to check

if a given specific date was valid. For example, 2 30 2012 is not valid, but 3 30 2012 is valid.

For this exercise, write a program called `chkdate.cpp` that reads in three integers from `cin` representing month, day, and year and checks if the entire date is a valid date between Jan 1, 1 and Dec 31, 9999.

This program simply prints out **Y** or **N** to report if the date is valid or not. A sample run might be:

```
./a.out          ← type this to run the program
9 29 2012       ← this is input to the program
Y               ← this is output from the program

./a.out
9 31 2012
N
```

example if the input is 2 8 2012 2 15 2012 the output will be 7. As another example, if the input is 1 1 2012 2 2 2012 the output will be 32.

#### REQUIREMENTS

- Do not worry about September 1752
- **DO** worry about leap years
- **DO** use constants in your code instead of month numbers

As usual:

**(a) Make Test Data:** Put some more test data in a file called `chkdate.data`. The data consists of three integers separated by spaces, then a colon, then a Y or N.

**(b) Write the Program:** As we did in lab, start small. Add new parts piece by piece.

**(c) Compile Insanely Often:** Compile, correct, compile, ..., then run and test.

#### Submitting Your Work

You are required to write three programs *and* to add two or more pieces of samples data to each of the three sample data files. Once you have done that, submit your work with:

```
provide comp11 hw3 max3.cpp max3.data 3wordsort.cpp 3wordsort.data chkdate.cpp chkdate.data
```

The provide program will compile and test your programs with the data you submit and with data from other students. You will receive instant feedback. You will have three chances to submit before the system will not accept any more submissions. Therefore, use your own test data before you use up a submission on compiling and testing you could do yourself.

#### Not Required, Extra Practice

The purpose of these weekly assignments is to give you practice with skills we learn and discuss in lab. These exercises build on the basic skills and reinforce understanding. Of course, more practice at increasing levels of difficulty is even better. If you want extra practice, not to be turned in, try this problem. It requires conditionals and loops.

#### The Problem

Write a program called `datediff.cpp` that takes as input two dates and prints out the number of days between them. The dates are entered as three integers: month day year. For