



Ideas, Mechanics, Now Coding

In lab1, we talked ideas -- algorithms, instructions, programs, logic errors, syntax errors, flow of execution. In lab2, we talked how-to -- login, using Linux commands, manage directories and files, edit and compile.

Now we are ready for the real work of the course: devising and analyzing procedures machines can perform. Each week we shall look at some specific vocabulary and syntax in C++ and then put those capabilities to work devising and analyzing procedures. All set?

Conditional Pricing

At some movie theaters, tickets are cheaper in the afternoon. How much cheaper? Why do they do that? Aren't they losing money?

☞ What are other examples of prices that depend on various conditions?

In C++ we can express conditional pricing with:

```
if ( hour < 17 )
{
    ticketPrice = 6.00;
}
else
{
    ticketPrice = 9.00;
}
```

Movie prices can also depend on other conditions. Some have discounts for children under 12 or for adults 65 or older. Keeping track of all these combinations of conditions can get so complicated that you need a computer to figure out the price. One such example is the world-famous Minnie's Mini Golf.

Minnie's Mini Golf

Minnie owns and operates a miniature golf course on Cape Cod. Needless to say, there's a lot of competition. On a beautiful Sunday afternoon, there are enough customers for everyone, but at other times, there are fewer people out looking for fun. Minnie has

decided she needs some way to differentiate her business and to attract customers.

Minnie has asked you to apply your skill with conditionals and boolean expressions to help her write a program to do conditional pricing. This program will determine the price based on certain desirability factors. The goal would be to lower the price when the market is down and raise the price when the market is booming. Your job is to write one function:

```
double price(int degreesF, int dayOfWeek,
             int hourOfDay, bool isRaining,
             int windSpeedMPH)
```

This function will determine the price depending on the input according to rules. A main routine that reads in data is already written. You will add the code to compute the price.

The Rules

The Rules for pricing are:

The base price is \$9.
 If the temperature is over 80 degrees, subtract 10 cents for every degree above 80.
 If the temperature is under 65 degrees, subtract 10 cents for every degree below 65.
 Subtract \$1 if it is a weekday and the hour is before 5pm.
 Subtract \$2 if it is raining.
 Subtract \$1.50 for each 15mph of wind.

The Rules for programming are:

Minnie does not like numbers in her code. Your function must contain **ZERO** numbers.
 Example: Rather than use 0.1 to represent 10 cents, use PER_DEGREE_PENALTY.
 Example: Rather than use 15.0 as wind speed granularity, use WIND_GRANULARITY.

Designing a Program: Step 1

This is our first lab in which you design and write C++ code pretty much from scratch. We discussed in

our first lab that designing code is like writing an essay or research paper -- you need a method to understand the topic and organize your thoughts.

We also said that once a program is written, it might not work correctly (logic errors), so you need ways to determine if your code is correct.

One very important and valuable technique that addresses both these needs (designing *and* analyzing) is to write some test cases. Do so now.

☞ Make up two test cases:

- Pick some values for all the variables,
- follow the pricing rules to figure the price
- Put the values and the answer on the worksheet
- Volunteer a few to put on the board.

Why Making Test Cases is Great

Starting a programming project by making test cases is one of the most useful strategies for designing successful programs. Creating the test data requires thinking through the problem actively. Furthermore, having the test data in hand means you are ready to test your code once you get past the syntax errors.

Starting Small: Step 2

Now that you have written your test data you are ready to start writing code. For this project, we provide the framework for the program, you just have to fill in the function that takes in the various settings and returns a price. Get started by:

0. Make a folder called lab3 in your comp11 folder. If you have a labs folder, make the lab3 folder inside the labs folder. Now, change into that folder.

1. Copy the file by typing:

```
cp /comp/11/files/lab3/golf.cpp .
```

Note the dot at the end of the command is NOT punctuation; the dot represents the current directory.

2. Open the file in any editor, text or graphic:

```
kate golf.cpp &
```

3. Edit the function to handle the temperature rule *only*. Use a conditional statement with a boolean expression. Start small, keep it simple at first.

4. Save, Compile, Edit, ...

```
g++ -Wall -Wextra golf.cpp
```

If you have syntax errors, note the line numbers, return to the editor, fix the errors, save, compile, ...

5. Run the program, Test some values

```
./a.out
```

Enter some data. Since your program only cares about the temperature, testing it should be pretty easy. Try a temperature below 80, above 80, and equal to 80. Does the price vary from \$9 the way you expect? If not, you have a logic error. If the program does work as expected, your code might be correct. But then again, it might not work for others.

Adding the Rest: Step 3

At this point you could add new conditions one by one, compiling and testing as you go. Or you could go for broke and write the rest of the conditions before compiling and testing. Pick a method and try it. If you don't like it, try the other. Feel free to discuss your ideas with your neighbors.

Testing the Whole Thing

Once the whole thing is written and compiles without warnings or syntax errors, you are ready to test it.

- ☞ Test it with the data from the person to your left
- ☞ Test it with the data from the person to your right
- ☞ Test it with your data

Were there any logic errors?

Finding Logic Errors

If your program does not produce the expected price based on the combination of settings, your program might have a bug. Here are two techniques for finding the cause of the error:

Trace the Code by Hand

Tracing is an important technique. Take the values that break your program and step through the code line by line performing the tests, operations, and assignments. See where things go wrong.

Explain the Code to Someone Else

Peer review is also a very important technique. As you explain the ideas and steps to another person, you are forced to think through the logic, and the other person is likely to ask you to explain things you may have skimmed over.

Submitting Your Work

You will use a program called `provide` to submit your work electronically. To submit your work for this lab, type:

```
provide comp11 lab3 golf.cpp
```