# Problem 1-3: Fahrenheit

This is CS50. Harvard University. Fall 2014.

## Table of Contents



Questions? Feel free to head to CS50 on Reddit[1], CS50 on StackExchange[2], or the CS50 Facebook group[3].

## Objectives

- Use variables and perform mathematical manipulations.
- Continue experimenting with CS50 IDE.
- Build your first utility program in C.

## Recommended Reading

- Pages 1 – 7, 9, and 10 of http://www.howstuffworks.com/c.htm.
- Chapters 1 – 5, 9, and 11 – 17 of *Absolute Beginner's Guide to C*.
- Chapters 1 – 6 of *Programming in C*.

[1] https://www.reddit.com/r/cs50
[2] http://cs50.stackexchange.com
[3] https://www.facebook.com/groups/cs50

# Academic Honesty

This course's philosophy on academic honesty is best stated as "be reasonable." The course recognizes that interactions with classmates and others can facilitate mastery of the course's material. However, there remains a line between enlisting the help of another and submitting the work of another. This policy characterizes both sides of that line.

The essence of all work that you submit to this course must be your own. Collaboration on problems is not permitted (unless explicitly stated otherwise) except to the extent that you may ask classmates and others for help so long as that help does not reduce to another doing your work for you. Generally speaking, when asking for help, you may show your code or writing to others, but you may not view theirs, so long as you and they respect this policy's other constraints. Collaboration on quizzes and tests is not permitted at all. Collaboration on the final project is permitted to the extent prescribed by its specification.

Below are rules of thumb that (inexhaustively) characterize acts that the course considers reasonable and not reasonable. If in doubt as to whether some act is reasonable, do not commit it until you solicit and receive approval in writing from your instructor. If a violation of this policy is suspected and confirmed, your instructor reserves the right to impose local sanctions on top of any disciplinary outcome that may include an unsatisfactory or failing grade for work submitted or for the course itself.

## Reasonable

- Communicating with classmates about problems in English (or some other spoken language).
- Discussing the course's material with others in order to understand it better.
- Helping a classmate identify a bug in his or her code, such as by viewing, compiling, or running his or her code, even on your own computer.
- Incorporating snippets of code that you find online or elsewhere into your own code, provided that those snippets are not themselves solutions to assigned problems and that you cite the snippets' origins.
- Reviewing past years' quizzes, tests, and solutions thereto.
- Sending or showing code that you've written to someone, possibly a classmate, so that he or she might help you identify and fix a bug.

- Sharing snippets of your own solutions to problems online so that others might help you identify and fix a bug or other issue.
- Turning to the web or elsewhere for instruction beyond the course's own, for references, and for solutions to technical difficulties, but not for outright solutions to problems or your own final project.
- Whiteboarding solutions to problems with others using diagrams or pseudocode but not actual code.
- Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your work for you.

## Not Reasonable

- Accessing a solution to some problem prior to (re-)submitting your own.
- Asking a classmate to see his or her solution to a problem before (re-)submitting your own.
- Decompiling, deobfuscating, or disassembling the staff's solutions to problems.
- Failing to cite (as with comments) the origins of code, writing, or techniques that you discover outside of the course's own lessons and integrate into your own work, even while respecting this policy's other constraints.
- Giving or showing to a classmate a solution to a problem when it is he or she, and not you, who is struggling to solve it.
- Looking at another individual's work during a quiz or test.
- Paying or offering to pay an individual for work that you may submit as (part of) your own.
- Providing or making available solutions to problems to individuals who might take this course in the future.
- Searching for, soliciting, or viewing a quiz's questions or answers prior to taking the quiz.
- Searching for or soliciting outright solutions to problems online or elsewhere.
- Splitting a problem's workload with another individual and combining your work (unless explicitly authorized by the problem itself).

- Submitting (after possibly modifying) the work of another individual beyond allowed snippets.
- Submitting the same or similar work to this course that you have submitted or will submit to another.
- Using resources during a quiz beyond those explicitly allowed in the quiz's instructions.
- Viewing another's solution to a problem and basing your own solution on it.

## Assessment

Your work on this problem set will be evaluated along four axes primarily.

**Scope**
: To what extent does your code implement the features required by our specification?

**Correctness**
: To what extent is your code consistent with our specifications and free of bugs?

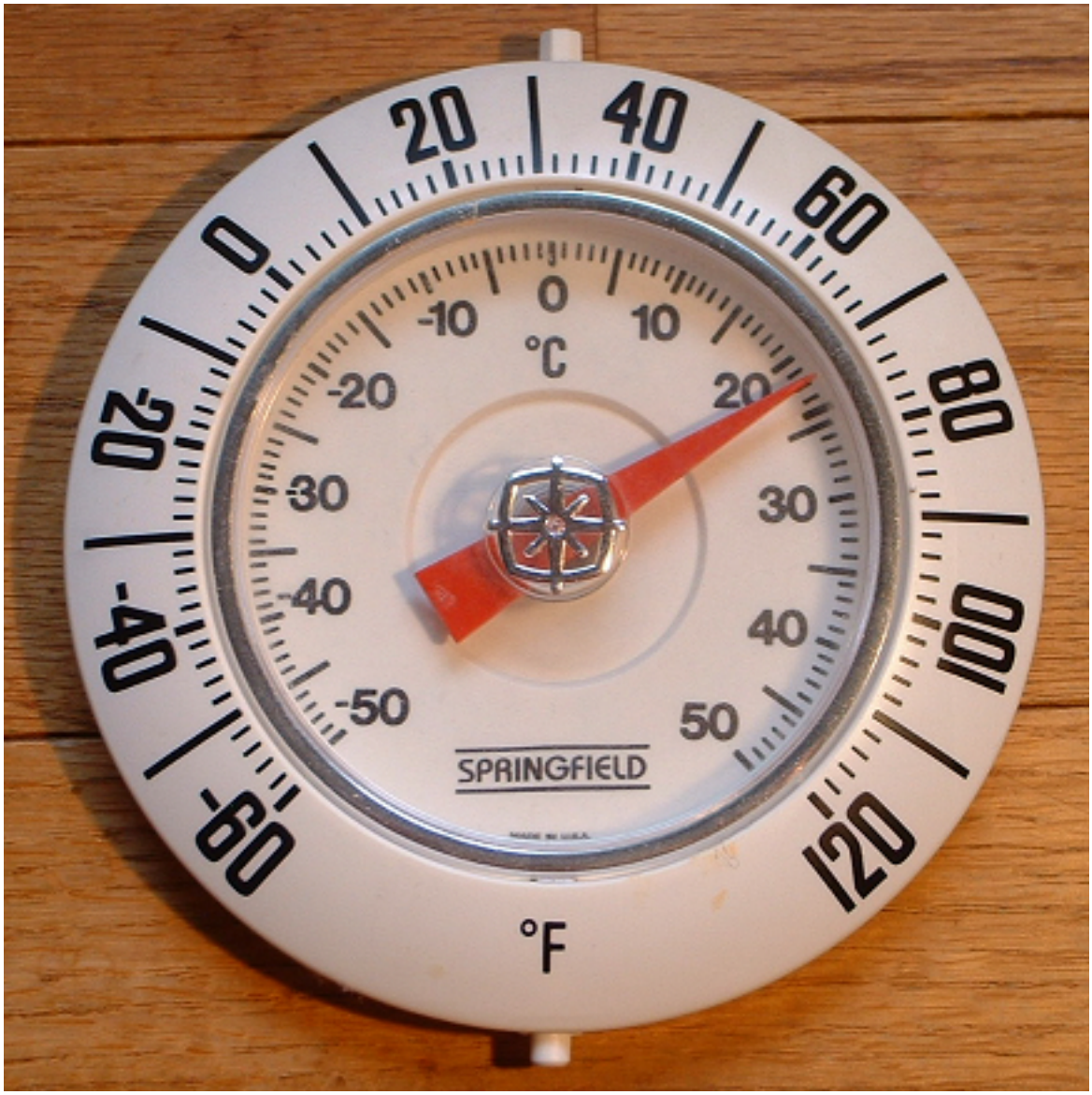
**Design**
: To what extent is your code written well (i.e., clearly, efficiently, elegantly, and/or logically)?

**Style**
: To what extent is your code readable (i.e., commented and indented with variables aptly named)?

To obtain a passing grade in this course, all students must ordinarily submit all assigned problems unless granted an exception in writing by the instructor.

## Turning Up the Heat

Time was if you wanted to know what the outdoor temperature was, you had to look around for one of these, which might have been mounted outside at your house.




Time really was that if you wanted to know what the temperature was outside, you just stood outside, but we digress. Nowadays, it's a fairly trivial exercise to pull open a weather app on your phone or visit a weather-reporting website to get the current temperature and the five-day forecast. No need to mess around with one of the above circular dials or its tube-of-mercury cousin[4].

Depending on where you live in the world, your country uses one of two major temperature scales. If you live in the United States (and don't spend most of your day working in a science lab), it's likely that you're familiar with the Fahrenheit scale, and so if you hear

[4] e.g., https://upload.wikimedia.org/wikipedia/commons/b/ba/Thermometer.jpg.

that it's 30° outside you're probably putting on a heavy coat and warm pants and bracing yourself for the possibility of snow, because that means it's quite cold, given that the freezing point of water is 32°. On the other hand, it's not until the temperature hits 212° that water boils.

In practically every other country of the world (or if you live in the United States and do in fact spend most of your day working in a science lab), you're probably most familiar with the Celsius or centigrade scale. In that case, if you hear that it's 30° outside you're probably going to rummage through your closet for a swimsuit and put on some sunblock, because it's a beautiful beach day. On the Celsius scale, 0° is the freezing point of water, and 100° is the boiling point.

We acknowledge that with most apps that tell you the weather, it's fairly simple to press that switch that switches the temperature display from Fahrenheit to Celsius (or vice versa), but before that process was automated for us, one either had to memorize some of the more common conversion points or had to use a mathematical formula to convert from one scale to another, which is important if you plan on traveling internationally.

For this problem, we're going to focus on converting in just one direction: *from* Celsius *to* Fahrenheit. As it happens, the formula for this conversion isn't terribly complex. (Phew!) One simply takes the current temperature in degrees Celsius (°C), multiplies it by 9, divides by 5, and then adds 32. The result is the equivalent temperature in degrees Fahrenheit (°F). Not bad, right? For the more visually inclined, this translates to this formula:

```
F = ((C * 9) / 5) + 32
```

Let's do a quick test to make sure things work as expected. Worldwide, the commonly accepted value for normal human body temperature is 37°C. If we plug "37" into that formula where °C goes and do the math (37 multiplied by 9 is 333, 333 divided by 5 is 66.6, 66.6 + 32 is 98.6) we get 98.6°F which is what folks in the United States know as normal human body temperature. So that checks out. Similarly if we plug in 0°C (the freezing point of water) into that formula does it convert to 32°F, and 100°C (the boiling point of water) is apparently equivalent to 212°F. Seems like things are going well.

# Getting Warmer

Log into your CS50 IDE account (remember how?) and be sure before doing anything to run `update50` in your terminal window, waiting until any updates finish processing before moving further. We'll remind you of this as much as possible, but do note that before each problem you should remember to run that command to be sure your IDE workspace is fully up-to-date.

Afterwards, create a file called `fahrenheit.c` (remember how?), ensuring that it is inside of your `unit1` directory, and then double-click on that file in the file tree to the left. You should now have an open window in Ace with the tab name `fahrenheit.c`, and you're ready to write your first utility program—a program that can be used by others to do something meaningful for them. (Not to knock the value of saying `hello, world`… but we can do so much more!)

Write a program that converts a temperature in Celsius to Fahrenheit, as per the sample output below, wherein underlined text represents some user's input[5].

```
username@ide50:~/workspace/unit1 $ ./fahrenheit
C: 100
F: 212.0
```

To solve this problem, you needn't do anything more complex than use your currently existing knowledge of C, and the information contained in this specification, including the temperature conversion formula. No matter how the user inputs the temperature in Celsius (that is, no matter to how many decimal places they choose), take care to display Fahrenheit to *exactly* one decimal place. No need to worry about floating-point imprecision or integer overflow, if you recall what those terms mean.[6]

Do recall that if you include `<cs50.h>` atop your `fahrenheit.c` file, you will have access to a function called `GetFloat`, which will allow the user to input a floating-point value (a number with a decimal point in it, also known as a *real number*).

[5] Incidentally, while spelling Fahrenheit correctly is a little tricky, but do be careful to do so lest you be told that your program doesn't exist!

[6] If you don't, you soon will!

Incidentally, know that `printf` can be used to specify how many places after the decimal point you wish to display to the user. For example, assuming you've written the following program in a file called `truncate.c`:

```
#include <stdio.h>

int main(void)
{
    float pi = 3.1415926535;
    printf("%.2f\n", pi);
}
```

When executed (by first compiling with `make truncate` and then executing with `./truncate`), this program will output the value of the variable `pi` to exactly 2 decimal places: `3.14`. Can you see why? Perhaps you can adapt that to display the converted temperature to just one decimal place?

Before turning in your solution, be sure to test the correctness of your program with `check50`, by executing the below.

```
check50 1516.unit1.fahrenheit fahrenheit.c
```

If you pass all the `check50` test cases, and get a green smiley face when you run things through `style50`, congratulations! If not, don't worry: it's just time for a little bit of debugging and/or cleaning up your code.

If you'd like to play with our own implementation of `fahrenheit` in the IDE, you may execute the below:

```
~cs50/unit1/fahrenheit
```

This was Problem 1-3.