This is CS50x

CS50's Introduction to Computer Science

OpenCourseWare

Donate ↗ (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
✋ (https://www.clubhouse.com/@davidjmalan) f (https://www.facebook.com/dmalan)
 (https://github.com/dmalan) ⃝ (https://www.instagram.com/davidjmalan/)
(https://www.linkedin.com/in/malan/) (https://orcid.org/0000-0001-5338-2522) Q
(https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com
/user/davidjmalan) (https://www.tiktok.com/@davidjmalan)
(https://davidjmalan.t.me/) (https://twitter.com/davidjmalan)

Hello

Getting Started

Recall that Visual Studio Code (aka VS Code) is a popular "integrated development environment" (IDE) via which you can write code. So that you don't have to download, install, and configure your own copy of VS Code, we'll use a cloud-based version instead that has everything you'll need pre-installed.

Log into code.cs50.io (https://code.cs50.io/) using your GitHub account. Once your "codespace" loads, you should see that, by default, VS Code is divided into three regions. Toward the top of VS Code is your "text editor", where you'll write all of your programs. Toward the bottom of is a "terminal window", a command-line interface (CLI) that allows you to explore your codespace's files and directories (aka folders), compile code, and run programs. And on the left is your file "explorer," a graphical user interface (GUI) via which you can also explore your codespace's files and directories.

Start by clicking inside your terminal window, then execute `cd` by itself. You should find that its "prompt" resembles the below.

```
$
```

Click inside of that terminal window and then type

```
mkdir hello
```

followed by Enter in order to make a directory called `hello` in your codespace. Take care not to overlook the space between `mkdir` and `hello` or any other character for that matter!

Here on out, to execute (i.e., run) a command means to type it into a terminal window and then hit Enter. Commands are "case-sensitive," so be sure not to type in uppercase when you mean lowercase or vice versa.

Now execute

```
cd hello
```

to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
hello/ $
```

If not, retrace your steps and see if you can determine where you went wrong!

Shall we have you write your first program? Execute

```
code hello.c
```

to create a new file called `hello.c`, which should open automatically in your codespace's text editor. As soon as you save the file with command-S (on macOS) or control-S (on Windows), it should also appear in your codespace's explorer.

Proceed to write your first program by typing precisely these lines into `hello.c`:

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

Notice how VS Code adds "syntax highlighting" (i.e., color) as you type, though VS Code's choice of colors might differ from this problem set's. Those colors aren't actually saved inside of the file itself; they're just added by VS Code to make certain syntax stand out. Had you not saved the file as `hello.c` from the start, VS Code wouldn't know (per the filename's extension) that you're writing C code, in which case those colors would be absent.

Listing Files

Next, in your terminal window, immediately to the right of the prompt (`hello/ $`), execute

```
ls
```

You should see just `hello.c` ? That's because you've just listed the files in your `hello` folder. In particular, you executed a command called `ls`, which is shorthand for "list." (It's such a frequently used command that its authors called it just `ls` to save keystrokes.) Make sense?

Compiling Programs

Now, before we can execute the `hello.c` program, recall that we must *compile* it with a *compiler*, translating it from *source code* into *machine code* (i.e., zeroes and ones). Execute the command below to do just that:

```
make hello
```

And then execute this one again:

```
ls
```

This time, you should see not only `hello.c` but `hello` listed as well? You've now translated the source code in `hello.c` into machine code in `hello`.

Now execute the program itself by executing the below.

```
./hello
```

Hello, world, indeed!

Getting User Input

Suffice it to say, no matter how you compile or execute this program, it only ever prints `hello, world`. Let's personalize it a bit, just as we did in class.

Modify this program in such a way that it first prompts the user for their name and then prints `hello, so-and-so`, where `so-and-so` is their actual name.

As before, be sure to compile your program with:

```
make hello
```

And be sure to execute your program, testing it a few times with different inputs, with:

```
./hello
```

Walkthrough

Here's a "walkthrough" (i.e., tour) of this problem, if you'd like a verbal overview of what to do too!

Hints

Don't recall how to prompt the user for their name?

Recall that you can use `get_string` as follows, storing its *return value* in a variable called `name` of type `string`.

```
string name = get_string("What's your name? ");
```

Don't recall how to format a string?

Don't recall how to join (i.e., concatenate) the user's name with a greeting? Recall that you can use `printf` not only to print but to format a string (hence, the `f` in `printf`), a la the below, wherein `name` is a `string`.

```
printf("hello, %s\n", name);
```

Use of undeclared identifier?

Seeing the below, perhaps atop other errors?

```
error: use of undeclared identifier 'string'; did you mean 'stdin'?
```

Recall that, to use `get_string`, you need to include `cs50.h` (in which `get_string` is *declared*) atop a file, as with:

```
#include <cs50.h>
```

How to Test Your Code

Execute the below to evaluate the correctness of your code using `check50`, a command-line program that will output happy faces whenever your code passes CS50's automated tests and sad faces whenever it doesn't! But be sure to compile and test it yourself as well!

```
check50 cs50/problems/2023/x/hello
```

Execute the below to evaluate the style of your code using `style50`, a command-line program that will output additions (in green) and deletions (in red) that you should make to your program in order to improve its style. If you have trouble seeing those colors, `style50` supports other modes (https://cs50.readthedocs.io/style50/) too!

```
style50 hello.c
```

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2023/x/hello
```