PRACTICE About classroom practice sessions

These and all of the following `PRACTCE` exercises are for you to complete in class. If you're not able, you should complete them at home using the available Zoom recordings.

If you have Emacs + Org-mode, you can do this effortlessly (but you have to know your way around Emacs + Org-mode). You can create new code blocks (like in Google Colab) and run them without having to enter `#include ... return 0` every time.

If you only have a browser, you have choices as explained in the orientation lecture. `onecompiler.com` is the easiest:

Open onecompiler.com/c in the browser if you haven't done it yet.

Create a new project with a `main.c` file

Give it a suitable title like `"Variables - Practice"`, write a short description, like:

> Project for practicing variables in C - CSC 100 class.

and tag it:

```
practice, variable, csc100
```

Make sure the visibility is `Public (visible to everyone)` so that you can post the link to Canvas if requested.

In the `editor settings`, check `Disable Code Autocomplete/Suggestions`

Delete the "hello world" printing line and off you go!

PRACTICE Everything is a number

- What do you get when you get the format wrong? The results will surprise you.
- Print these values using the requested format in a `printf` call:

	VALUE	FORMAT	SPECIFIER
1	`3.14`	integer	`%i`
2	`3.14`	character	`%c`
3	`3`	floating-point	`%f`
4	`3`	character	`%c`
5	`'3'`	integer	`%i`
6	`'3'`	floating-point	`%f`

- Solution & Explanation:

  1. **Printing a floating-point number as an integer:** The output is an integer representation of the bits read by `printf` - it changes because the mismatch between the format specifier and the value

argument causes **undefined behavior**.

```
printf("%i\n", 3.14);
```

```
-983310104
```

2. **Printing a floating-point number as character**: Prints a garbage character (not recognized as an ASCII character).

```
printf("%c\n", 3.14);
```

```
X
```

3. **Printing an integer as floating-point number**: Undefined behavior again. This result could also be another value.

```
printf("%f\n", 3);
```

```
0.0
```

4. **Printing an integer as character**: 3 is interpreted as the character with the ASCII code 3.

```
printf("%c\n", 3);
```

```
□
```

5. **Printing a character as an integer**: Results in printing the ASCII code of the character.

```
printf("%i\n", '3');
```

```
51
```

6. **Printing a character as a floating-point number**: Undefined behavior.

```
printf("%f\n", '3');
```

```
0.0
```

PRACTICE Print challenge

Print `3.14 - 3 = .14` using `printf` and the values `3.14`, `3`, and `.14`

```
printf("%f - %i = %f\n", 3.14, 3, .14);
printf("%.2f - %i = %.2f\n", 3.14, 3, .14);
```

```
3.140000 - 3 = 0.140000
3.14 - 3 = 0.14
```

Created: 2025-01-06 Mon 20:28