Let’s take a detailed look at some of the code files in our current working directory:

```
ls -l | head -n 3

## -rw-rw-r-- 1 sean sean 138 Jun 26 12:51 addseq.sh
## -rw-rw-r-- 1 sean sean 146 Jun 26 14:45 addseq2.sh
## -rw-rw-r-- 1 sean sean 140 Jan 29 10:06 bigmath.sh
```

The left column of this table contains a series of individual characters and dashes. The first hyphen (-) signifies that each of the entries in this list are files. If any of them were directories then instead of a hyphen there would be a d. Excluding the first hyphen we have the following string: rw-rw-r--. This string reflects the **permissions** that are set up for this file. There are three permissions that we can grant: the ability to **read** the file (r), **write** to or edit the file (w), or **execute** the file (x) as a program. These three permissions can be granted on three different levels of access which correspond to each of the three sets of rwx in the permissions string: the owner of the file, the group that the file belongs to, and everyone other than the owner and the members of a group. Since you created the file you are the owner of the file, and you can set the permissions for files that you own using the chmod command.

The chmod command takes two arguments. The first argument is a string which specifies how we’re going to change permissions for a file, and the second argument is the path to the file. The first argument has to be composed in a very specific way. First we can specify which set of users we’re going to change permissions for:

Character	Meaning
u	The owner of the file
g	The group that the file belongs to
o	Everyone else
a	Everyone above

We then need to specify whether we’re going to add, remove, or set the permission:

Character	Meaning
+	Add permission
-	Remove permission
=	Set permission

Finally we specify what permission we're changing:

Character	Meaning
r	Read a file
w	Write to or edit a file
x	Execute a file

Let's use echo to write a very short program which we'll call short.

```
echo 'echo "a small program"' > short
```

Normally if we wanted to run short we would enter bash short into the console. If we make this file executable we would only need to enter short into the command line to run the program, just like a command! Let's take a look at the permissions for short.

```
ls -l short

## -rw-r--r--  1 sean  staff  23 Jun 28 09:47 short

```

We want to make this file executable and we're the owner of this file since we created it. This means we can combine u, +, and x in order make short executable. Let's try it:

```
chmod u+x short
ls -l short

## -rwxr--r--  1 sean  staff  23 Jun 28 09:47 short

```

We successfully added the x! To run an executable file we need to specify the path to the file, even if the path is in the current directory, meaning we need to prepend ./ to short. Now let's try running the program.

```
./short

## a small program

```

Looks like it works! There is one small detail we should add to this program though. Even though we've made our file executable, if we give our program to somebody else they might be using a shell that doesn't know how to execute our program. We need to indicate how the program should be run by adding a special line of text to the beginning of our program called a shebang. The shebang always begins with #! followed by the path to the program which will execute the code in our file. The shebang for indicating that we want to use Bash is #!/usr/bin/env bash, which we've been adding to the start of our scripts for a while now! Let's rewrite this program to include the Bash shebang and then let's run the program.

```
echo '#!/usr/bin/env bash' > short
echo 'echo "a small program"' >> short
## a small program

```

Now our Bash script is ready to go!

Mark as completed