I strongly suggest that you make sure all the code we are doing is in the course repositories. It will be a helpful reference when you are doing assignment 2!

1. Modify our hellomod.c file from last day to include the lines

```c
#include <linux/moduleparams.h> //Allows Command-Line Parameters

MODULE_LICENSE(“GPL”);
```

Note: The module license indicates that the LKM follows the GPL (GNU General Public License). Adding this line prevents the line in `/var/log/messages` about our unlicensed module tainting the kernel.

2. make our hellomod LKM again and insert it into the kernel. Check the log messages to ensure it still works and that the message about tainting is gone.

   $ make
   $ insmod ./hellomod.ko
   $ tail /var/log/messages

3. Remove our LKM before we change it again.

   $ rmmod hellomod

4. In order to use modprobe we need to first add some additional functionality to our Makefile. Modify your Makefile so it looks like this:

   ```make
   obj-m += hellomod.o
   all:
      make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
      make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules_install
   clean:
      make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
   
   Run
   $ make clean
   $ make
   ```
5. The following command will use \texttt{modprobe} to load our module.
\begin{verbatim}
$ modprobe hellomod
\end{verbatim}
Look at the log files again to verify our module is loaded and working. You can also look at the currently loaded modules to see \texttt{hellomod}
\begin{verbatim}
$ lsmod | head
\end{verbatim}

6. Lets unload our module using \texttt{modprobe}. This time lets use the additional \texttt{-v} option to get verbose output of what \texttt{modprobe} is doing.
\begin{verbatim}
$ modprobe -vr hellomod
\end{verbatim}
Note: The \texttt{-r} option is used to tell \texttt{modprobe} to remove the specified LKM. The \texttt{-v} option does not need to be used unless you want to see what \texttt{modprobe} is doing. The \texttt{-v} option can be used with any other \texttt{modprobe} options.

7. We can also specify a different name for our module than what \texttt{modprobe} uses by default.
\begin{verbatim}
$ modprobe hellomod -o hello
\end{verbatim}
This command tells \texttt{modprobe} to load the \texttt{hellomod.ko} module but to call it \texttt{hello} instead of \texttt{hellomod}. If you list out the modules now you will see our module is now called \texttt{hello}.

To remove this module we have to use the command
\begin{verbatim}
$ modprobe -r hellomod -o hello
\end{verbatim}

8. You can use \texttt{modprobe} to list out all modules that can be loaded.
\begin{verbatim}
$ modprobe -l | less
\end{verbatim}
This is a very long list so we can also search the list for a specific module.
\begin{verbatim}
$ modprobe -l hellomod
\end{verbatim}

9. Lets modify our LKM so that it takes command line parameters by adding the following lines. Add these lines right after the end of the \texttt{#include} statements and after our \texttt{MODULE_LICENSE}(“GPL”); line.
\begin{verbatim}
static char* message = " ";
static int num = 1;
module_param(message, charp, 0);
module_param(num, int, 0);
\end{verbatim}
Also add some \texttt{printk} statements to print out the values of these variables.
Note: The arguments to module_param are as follows: First argument is name of the variable, second argument is the type in this case a char pointer or an integer, third argument are flags/permissions, 0 should usually work as a third argument.

10. Run `make` again and then

   $$\text{modprobe hellomod message=\"Testing\" num=457}$$

   Check the log file again to verify the output of our module. It should now print out the line “Testing” and the number 457 in addition to its “Hello World” lines.

11. Remove our LKM once more and you are done! You should now understand how to load an LKM that takes command line parameters with `modprobe`. You will need this for part of assignment 2. Don’t forget to commit your code to the repository!