Nicholas LaJoie, ECE 331, HW 13

```
// Author: Nicholas LaJoie
// ECE 331 - HW 13
// Date: May 8, 2017
```

1. PHP Table

```
<?php
function table($w, $h)
{
    print "<TABLE BORDER=1\n>";
    for ($i = 0; $i < ($w * $h); $i++) {
        if ($i % $w == 0) {
            print "<TR>";
        } else if ($i % $w == ($w - 1)) {
            print "</TR>";
        } else {
            print "<TD>$i";
        }
    }
    print "</TABLE>\n";
}
?>
```

2. Addresses & Netmasks

The following addresses were calculated using the following techniques:
--> NA = IP & NM
--> BA = IP | (~NM)
--> The NM was determined by lining up the binary representations of the NA & BA and determining the point when they are no longer identical (for the second one, that point was the last 11 binary digits). 32 - 11 = 21, which is the NM, /21.
--> Using this NM, an assignable IP was selected based on the addresses available.

Network (DDN)	Assignable IP (DDN)	Netmask (CIDR)	Broadcast (DDN)
40.240.0.0	40.241.231.123	/13	40.247.255.255
2.3.88.0	2.3.89.1	/21	2.3.95.255

3. Python Morse Tester

```
#!/usr/bin/python

m = open("/dev/morse")              # Morse Device
f = open("/usr/share/dict/words")   # Dictionary file

for line in f:
    print line          # Print to stdout
    m.write(line)       # Write to morse device

# Clean up
m.close()
f.close()
```

4. Run two instances of above script: ./pmorse.py & ./pmorse.py

5. Q & A

a) The ’addrlen’ specifies the length of a socket address in order to match the format of the address being passed. It is possible many protocols may exist, which means the length may vary. (Source: socket(2))

b) The processor of a single core system can often suspend a process to run another proces s at any given point (known as a ’context switch’). This is a key feature of a preemptive kern el (which the Linux kernel is). If multiple processes require a piece of data, context switche

s can cause unexpected behavior, which implies that a lock is required. A lock will ensure the re are no concurrency issues due to context switching on a processor.

c) printf() is a userspace program that is part of the C library. There is no stdout at th e kernel level (printk() prints to logs), so there is no use for printf() in the kernel! The k ernel and user spaces use different libraries (i.e. kernel does NOT use the C library), so tha t is one reason for the separation of the functions.

d) If proper error checking is not done in the kernel space, it is very easy to accidental ly have a buffer overflow that could cause serious security issues, and data may be overwritte n unintentionally. This could corrupt the kernel/system. Proper error checking can prevent a " kernel panic", and writing code like this is just a good practice in general. Additionally, it will prevent security flaws such as an unintentional "backdoor" on the machine or network.

6. Kernel Questions

a) No

b) It won’t work because the lock is initialized every time the write function is called. It creates a lock, then immediately checks if it is locked. For every call made to this functi on, there is a race condition to modify (initialize) the ’lock’ variable. This will cause conc urrency issues, as it will be difficult to tell what state the lock is in if it’s constantly b eing initialized.

Additionally, there is a memory leak! ’dat_buf’ is being allocated but never freed during any of the return code when a failure occurs.

c) The initialize code for the lock (definition & initialization) should be moved outside of the function and declared as a global variable. A single global mutex struct can then be ch ecked by the code following the ’init_mutex’ above, and will behave appropriately. Also, I wou ld add a line ’kfree(dat_buf);’ to prevent the memory leak.