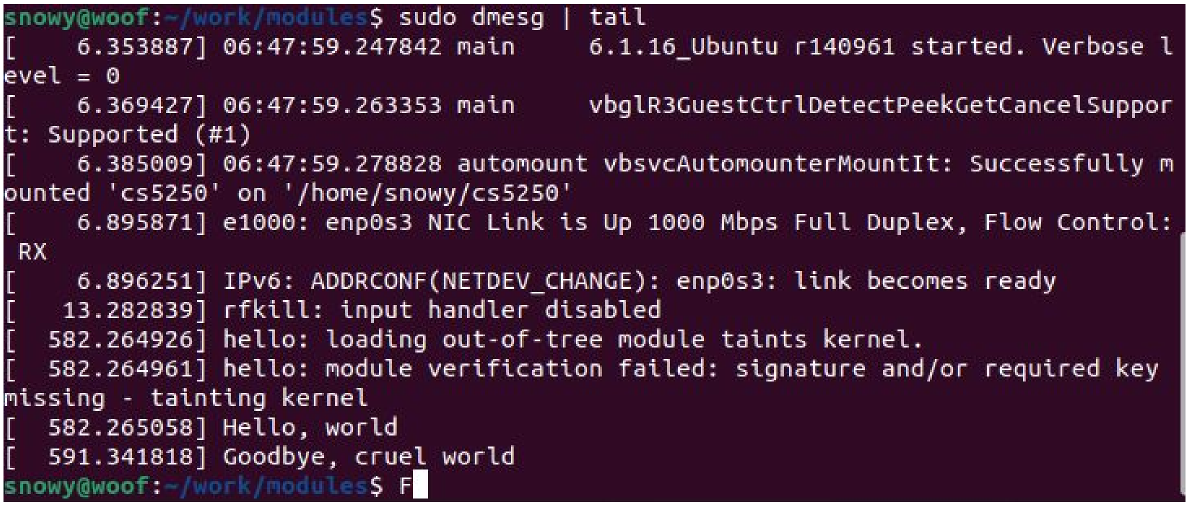
Assignment 3

Part A

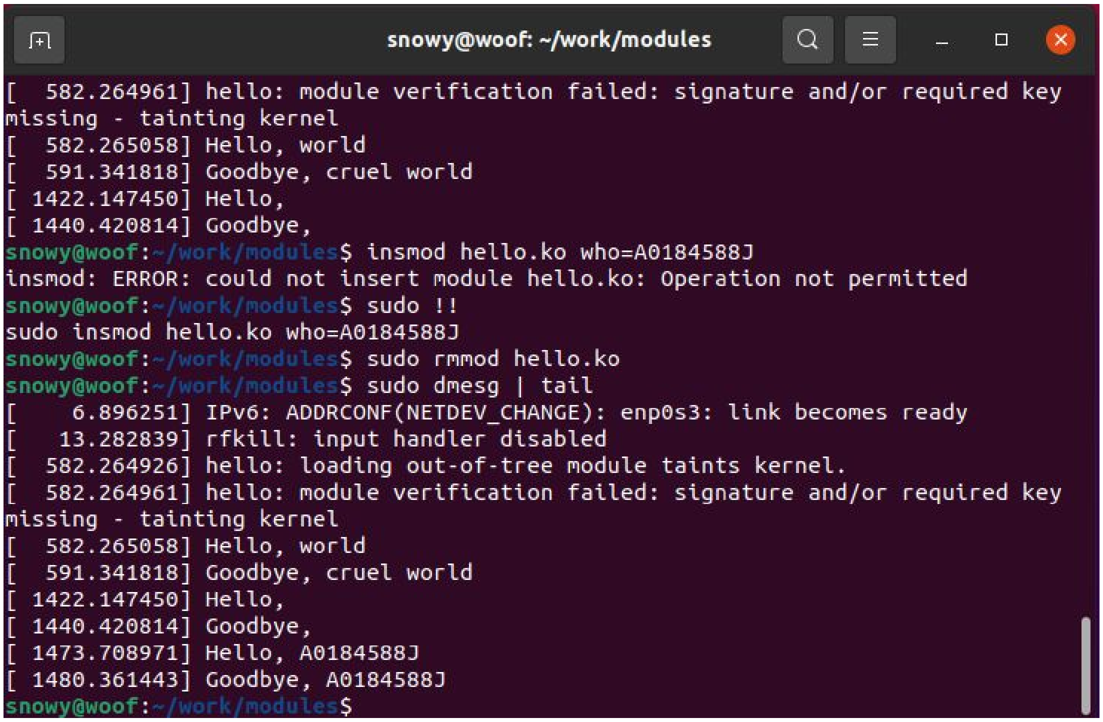
1. No 1.
    a. question a
        i. module_init is called when the module is installed
        ii. module_exit is called when the module is removed
    b. For:
        i. building the module using makefile

```
obj-m += hello.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

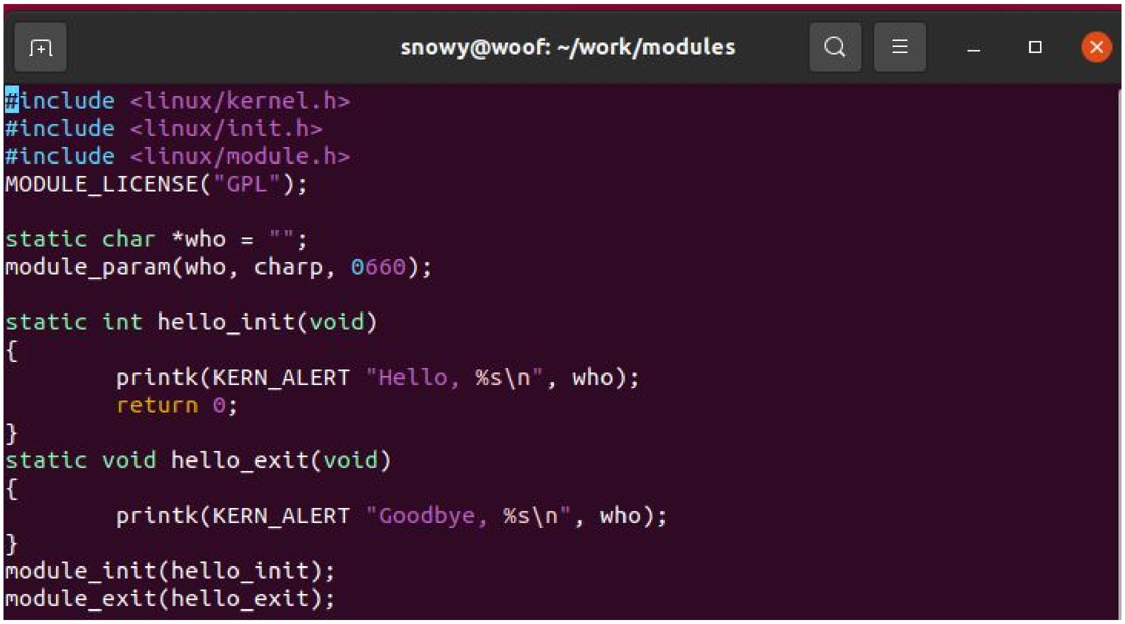
        ii. Installing the module using insmod hello.ko
        iii. Removing module using rmmod hello.ko
    c.



```
snowy@woof:~/work/modules$ sudo dmesg | tail
[    6.353887] 06:47:59.247842 main     6.1.16_Ubuntu r140961 started. Verbose l
evel = 0
[    6.369427] 06:47:59.263353 main     vbglR3GuestCtrlDetectPeekGetCancelSuppor
t: Supported (#1)
[    6.385009] 06:47:59.278828 automount vbsvcAutomounterMountIt: Successfully m
ounted 'cs5250' on '/home/snowy/cs5250'
[    6.895871] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    6.896251] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   13.282839] rfkill: input handler disabled
[  582.264926] hello: loading out-of-tree module taints kernel.
[  582.264961] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  582.265058] Hello, world
[  591.341818] Goodbye, cruel world
snowy@woof:~/work/modules$ F
```

d. 

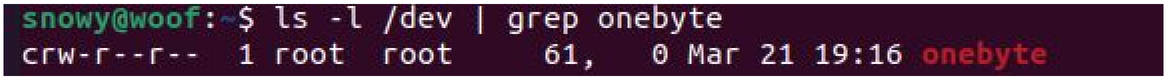
```
[  582.264961] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  582.265058] Hello, world
[  591.341818] Goodbye, cruel world
[ 1422.147450] Hello,
[ 1440.420814] Goodbye,
snowy@woof:~/work/modules$ insmod hello.ko who=A01845883
insmod: ERROR: could not insert module hello.ko: Operation not permitted
snowy@woof:~/work/modules$ sudo !!
sudo insmod hello.ko who=A01845883
snowy@woof:~/work/modules$ sudo rmmod hello.ko
snowy@woof:~/work/modules$ sudo dmesg | tail
[    6.896251] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   13.282839] rfkill: input handler disabled
[  582.264926] hello: loading out-of-tree module taints kernel.
[  582.264961] hello: module verification failed: signature and/or required key
missing - tainting kernel
[  582.265058] Hello, world
[  591.341818] Goodbye, cruel world
[ 1422.147450] Hello,
[ 1440.420814] Goodbye,
[ 1473.708971] Hello, A01845883
[ 1480.361443] Goodbye, A01845883
snowy@woof:~/work/modules$
```



```c
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("GPL");

static char *who = "";
module_param(who, charp, 0660);

static int hello_init(void)
{
        printk(KERN_ALERT "Hello, %s\n", who);
        return 0;
}
static void hello_exit(void)
{
        printk(KERN_ALERT "Goodbye, %s\n", who);
}
module_init(hello_init);
module_exit(hello_exit);
```

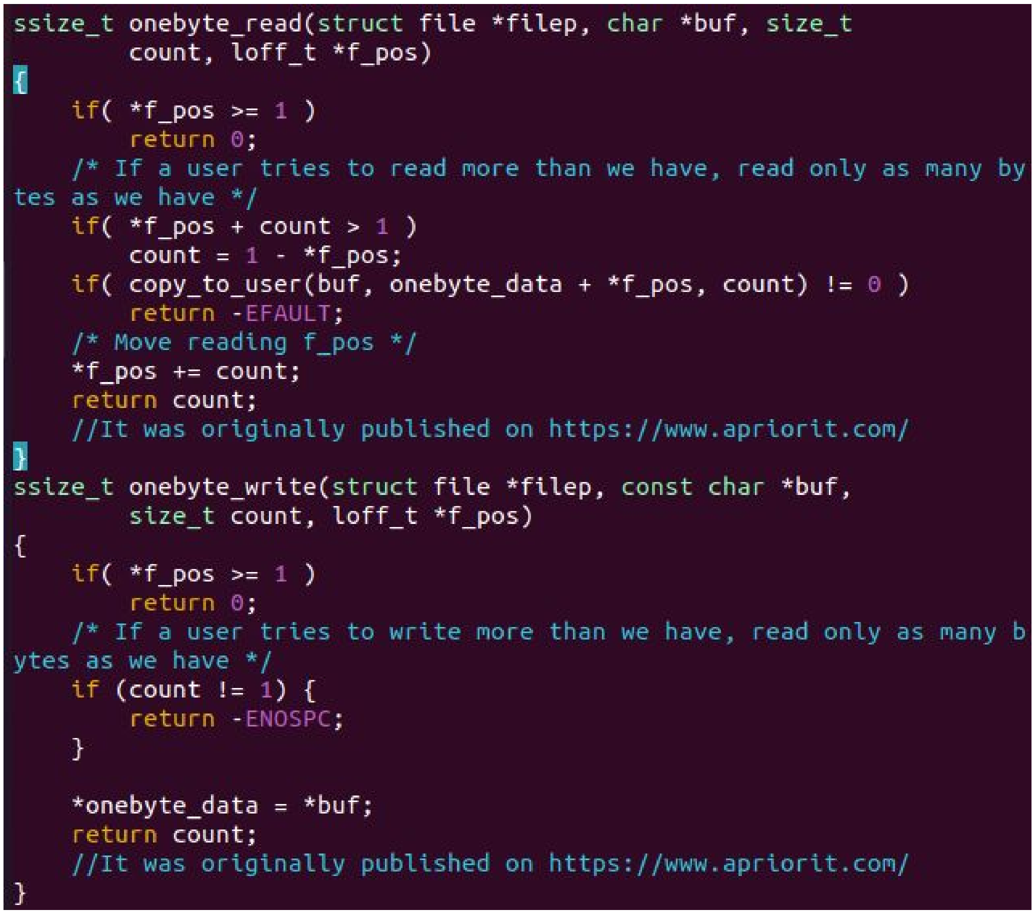
2.

a. Since my onebyte is in 61, my mknod command is
mknod /dev/onebyte c 61 0

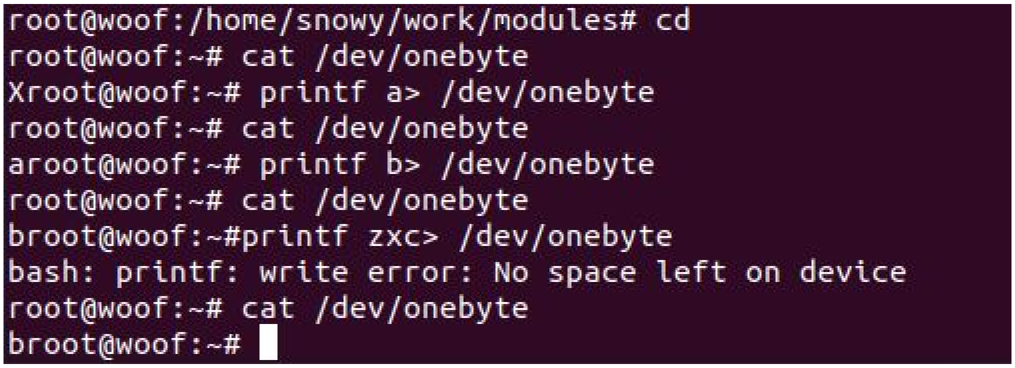
b. 

```
snowy@woof:~$ ls -l /dev | grep onebyte
crw-r--r--  1 root  root      61,   0 Mar 21 19:16 onebyte
```

c.



```
ssize_t onebyte_read(struct file *filep, char *buf, size_t
        count, loff_t *f_pos)
{
    if( *f_pos >= 1 )
        return 0;
    /* If a user tries to read more than we have, read only as many by
tes as we have */
    if( *f_pos + count > 1 )
        count = 1 - *f_pos;
    if( copy_to_user(buf, onebyte_data + *f_pos, count) != 0 )
        return -EFAULT;
    /* Move reading f_pos */
    *f_pos += count;
    return count;
    //It was originally published on https://www.apriorit.com/
}
ssize_t onebyte_write(struct file *filep, const char *buf,
        size_t count, loff_t *f_pos)
{
    if( *f_pos >= 1 )
        return 0;
    /* If a user tries to write more than we have, read only as many b
ytes as we have */
    if (count != 1) {
        return -ENOSPC;
    }

    *onebyte_data = *buf;
    return count;
    //It was originally published on https://www.apriorit.com/
}
```



```
root@woof:/home/snowy/work/modules# cd
root@woof:~# cat /dev/onebyte
Xroot@woof:~# printf a> /dev/onebyte
root@woof:~# cat /dev/onebyte
aroot@woof:~# printf b> /dev/onebyte
root@woof:~# cat /dev/onebyte
broot@woof:~#printf zxc> /dev/onebyte
bash: printf: write error: No space left on device
root@woof:~# cat /dev/onebyte
broot@woof:~#
```

Part B

1. Processes:
    a.
        i. P1 burst CPU time: 23, arrival time 0
        ii. P2 burst CPU time: 12, arrival time 5
        iii. P3 burst CPU time: 41, arrival time 10
        iv. P4 burst CPU time: 17, arrival time 15
        v. P5 burst CPU time: 29, arrival time 40

All the time here is before context switching

time	P1	P2	P3	P4	P5
0	23	-	-	-	-
5	18	12	-	-	-
10	18	8	41	-	-
15	18	3	41	17	-
18	18	0	41	17	-
36	18	0	41	0	-
40	15	0	41	0	29
55	0	0	41	0	29
85	0	0	41	0	0
127	0	0	0	0	0

P1 finished in 55 seconds
P2 finished in 18 seconds
P3 finished in 127 seconds
P4 finished in 36 seconds
P5 finished in 85 seconds

b.

time	P1	P2	P3	P4	P5	Queue
0	23	-	-	-	-	
5	18	12	-	-	-	1, 2
10	13	12	41	-	-	2, 3, 1
15	13	7	41	17	-	2, 3, 1, 4
20	13	2	41	17	-	3, 1, 4, 2
30	13	2	31	17	-	1, 4, 2, 3
40	3	2	31	17	29	4, 2, 3, 1, 5
50	3	2	31	7	29	2, 3, 1, 5, 4
52	3	0	31	7	29	3, 1, 5, 4
62	3	0	21	7	29	1, 5, 4, 3

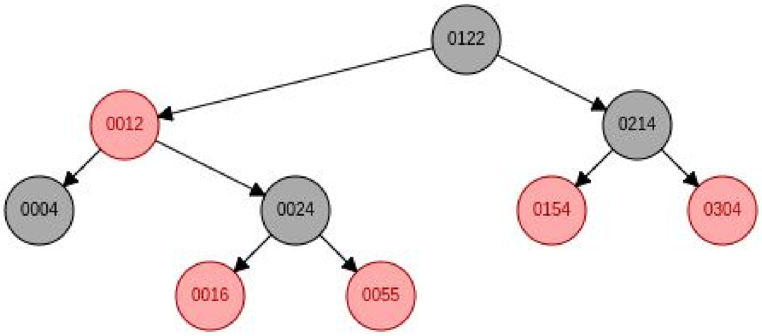
65	0	0	21	7	29	5, 4, 3
75	0	0	21	7	19	4, 3, 5
82	0	0	21	0	19	3, 5
92	0	0	11	0	19	5, 3
102	0	0	11	0	9	3, 5
112	0	0	1	0	9	5, 3
121	0	0	1	0	0	3
122	0	0	0	0	0	

P1 finished at 65 seconds.
P2 finished at 52 seconds.
P3 finished at 122 seconds.
P4 finished at 82 seconds.
P5 finished at 121 seconds.

2.

a. SJF will be the same as FCFS if all the jobs are the same.
So in a non-preemptive case and all the lengths are the same, as long as there is no job waiting, any scheduling will have the same minimizing the average response time.
No matter how you schedule the task, the number of tasks waiting for a particular time t[i] will always be the same in this case.
Since like the average response time is the sum of response time, moving around the order will only move the response time from process a to process b. But the sum will remain the same.

b. Since the SRTF will run the shortest jobs and the lengths are the same, the shortest job will be the one who is already running. If there is no program running, then all the jobs will have the same length.



3. https://www.cs.usfca.edu/~galles/visualization/RedBlack.html