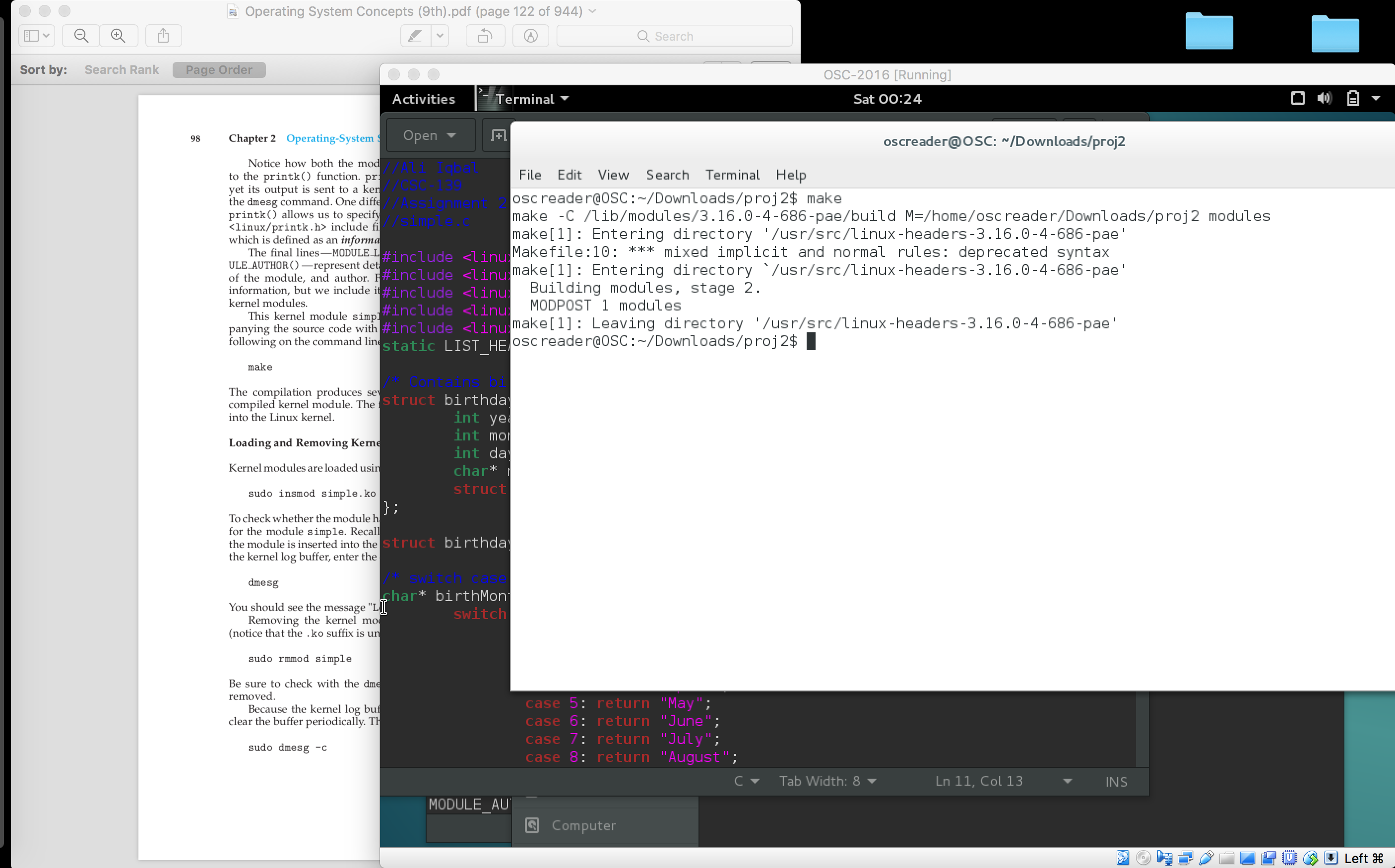
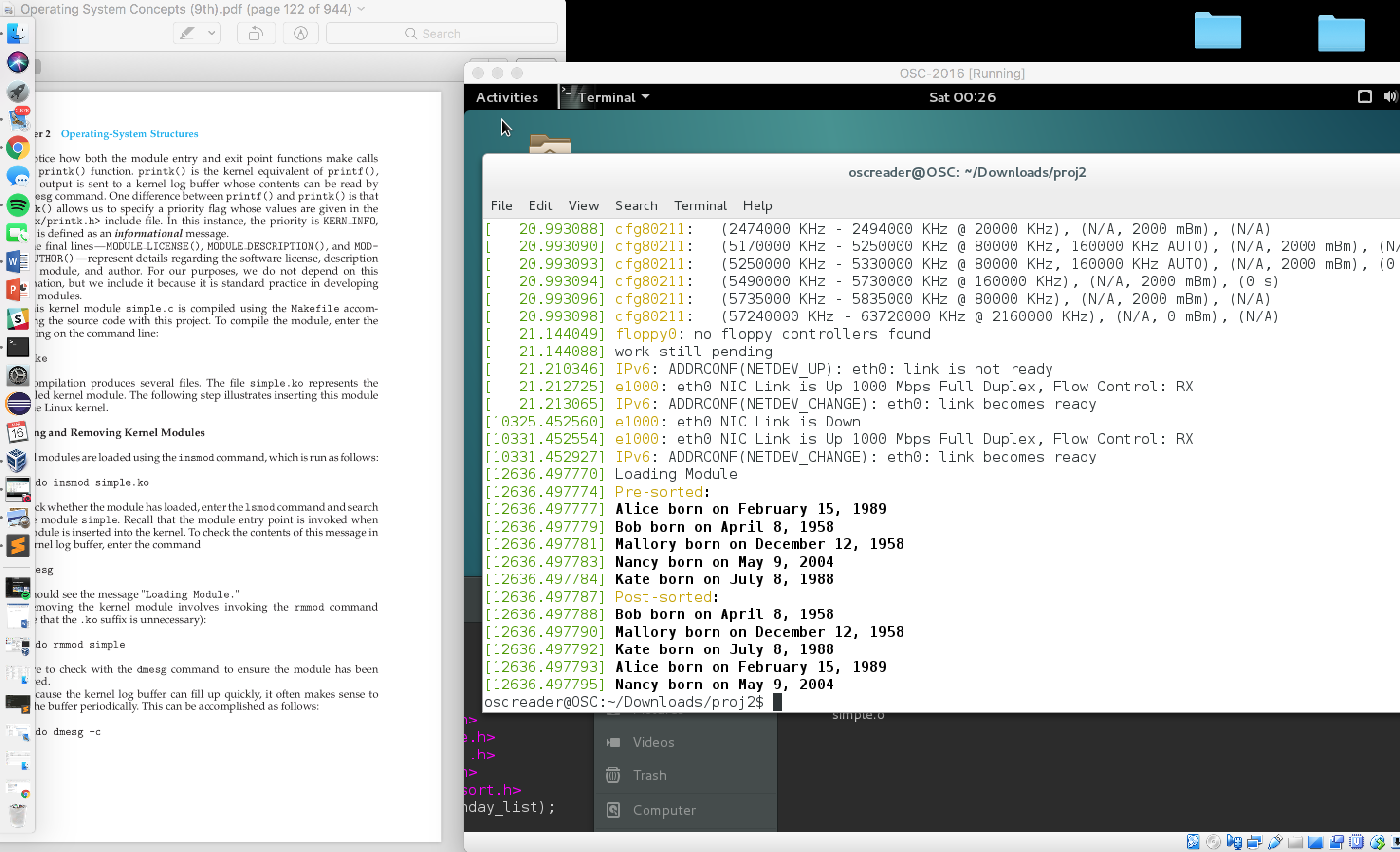
Assignment #2 – Linux Kernel Modules

(This report contains: source code, make-file code and necessary screenshots.)

**Screenshots:**





**source code:**

//Ali Iqbal

//CSC-139

//Assignment 2

//simple.c

#include <linux/init.h>

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/slab.h>

#include <linux/list\_sort.h>

static LIST\_HEAD(birthday\_list);

/\* Contains birthday attributes and linked list to hold data \*/

struct birthday{

int year;

int month;

int day;

char\* name;

struct list\_head list;

};

struct birthday \*ptr, \*nex;

/\* switch case to return name of month from actual month number\*/

char\* birthMonth(int month){

switch(month){

case 1: return "Januray";

case 2: return "February";

case 3: return "March";

case 4: return "April";

case 5: return "May";

case 6: return "June";

case 7: return "July";

case 8: return "August";

case 9: return "September";

case 10: return "October";

case 11: return "November";

case 12: return "December";

default: return "Not a Real Month";

}

}

/\* method compares birthday attributes in linkedList\*/

int compareBirthday(void \*ptr, struct list\_head \*one, struct list\_head \*two){

struct birthday \*brd1 = list\_entry(one, struct birthday, list);

struct birthday \*brd2 = list\_entry(two, struct birthday, list);

if(brd1 -> year < brd2 -> year) {

return -1;

} else if(brd1 -> year > brd2 -> year){

return 1;

} else {

if(brd1 -> month < brd2 -> month) {

return -1;

} else if(brd1 -> month > brd2 -> month){

return 1;

} else {

if(brd1 -> day < brd2 -> day) {

return -1;

} else if(brd1 -> day < brd2 -> day){

return 1;

} else {

return 0;

}

}

}

}

/\* This function is called when the module is loaded. \*/

int simple\_init(void)

{

struct birthday \*alice;

struct birthday \*bob;

struct birthday \*mallory;

struct birthday \*nancy;

struct birthday \*kate;

printk(KERN\_INFO "Loading Module\n");

alice = kmalloc(sizeof(\*alice), GFP\_KERNEL);

alice -> year = 1989;

alice -> month = 2;

alice -> day = 15;

alice -> name = "Alice";

INIT\_LIST\_HEAD(&alice -> list);

bob = kmalloc(sizeof(\*bob), GFP\_KERNEL);

bob -> year = 1958;

bob -> month = 4;

bob -> day = 8;

bob -> name = "Bob";

INIT\_LIST\_HEAD(&bob -> list);

mallory = kmalloc(sizeof(\*mallory), GFP\_KERNEL);

mallory -> year = 1958;

mallory -> month = 12;

mallory -> day = 12;

mallory -> name = "Mallory";

INIT\_LIST\_HEAD(&mallory -> list);

nancy = kmalloc(sizeof(\*nancy), GFP\_KERNEL);

nancy -> year = 2004;

nancy -> month = 5;

nancy -> day = 9;

nancy -> name = "Nancy";

INIT\_LIST\_HEAD(&nancy -> list);

kate = kmalloc(sizeof(\*kate), GFP\_KERNEL);

kate -> year = 1988;

kate -> month = 7;

kate -> day = 8;

kate -> name = "Kate";

INIT\_LIST\_HEAD(&kate -> list);

list\_add\_tail(&alice -> list, &birthday\_list);

list\_add\_tail(&bob -> list, &birthday\_list);

list\_add\_tail(&mallory -> list, &birthday\_list);

list\_add\_tail(&nancy -> list, &birthday\_list);

list\_add\_tail(&kate -> list, &birthday\_list);

printk("Pre-sorted: \n");

list\_for\_each\_entry(ptr, &birthday\_list,list) {

printk("%s born on %s %d, %d\n", ptr -> name,

birthMonth(ptr -> month), ptr -> day, ptr -> year);

}

list\_sort(NULL, &birthday\_list, compareBirthday);

printk("Post-sorted: \n");

list\_for\_each\_entry(ptr, &birthday\_list,list) {

printk("%s born on %s %d, %d\n", ptr -> name,

birthMonth(ptr -> month), ptr -> day, ptr -> year);

}

return 0;

}

/\* This function is called when the module is removed. \*/

void simple\_exit(void) {

printk(KERN\_INFO "Removing Module\n");

list\_for\_each\_entry\_safe(ptr, nex, &birthday\_list, list) {

printk("Removing birthday of: %s \n", ptr -> name);

/\*Remove the list\*/

list\_del(&ptr -> list);

/\*free the list memory\*/

kfree(ptr);

}

}

/\* Macros for registering module entry and exit points. \*/

module\_init( simple\_init );

module\_exit( simple\_exit );

MODULE\_LICENSE("GPL");

MODULE\_DESCRIPTION("Simple Module");

MODULE\_AUTHOR("SGG");

**make-file code:**

obj-m += simple.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