```
/* ANRC RHKI */
/* Lab7: Memory Management Lab */
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

#include <linux/delay.h>
#include <linux/kthread.h>

#include <asm/string.h>
#define DRIVER_AUTHOR "ANRC"
#define DRIVER_DESC   "Lab7"

MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?

struct task_struct *ts;

/* Global kmem_cache struct */
static struct kmem_cache *my_cachep;

int init(void);
void cleanup(void);

static void init_my_cache(void)
{
        /* initialize my_cachep using kmem_cache_create */

        return;
}

void slab_ex(void)
{
        void *myobject;

        /* validate kmem_cache structure */
        printk("slabex: Cache name is %s\n", kmem_cache_name(my_cachep));
        printk("slabex: Cache object size is %d\n", kmem_cache_size(my_cachep));

        /* allocate a new object into the cache using kmem_cache_alloc */

        printk("slabex: Allocating object in my_cachep with GFP_KERNEL flag\n");

        if(myobject)
        {
                printk("slabex: Object created, freeing ...\n");

                /* Free myobject using kmem_cache_free */

                printk("slabex: Object freed\n");
        }
        else
        {
                printk("slabex: kmem_cache_alloc failed!\n");
        }
}

static void remove_my_cache(void)
{
        printk("slabex: destroying kmem cache\n");

        /* destroy kmem_cache using kmem_cache_destroy */
        /* check to see if the cache is valid first ... */
        return;
}

int init(void)
{
        printk(KERN_INFO "slabex: init_module() called\n");
        printk("slabex: Initializing kmem cache\n");
        init_my_cache();

        slab_ex(); /* create and free objects within the cache */

        return 0;
}

void cleanup(void)
{
        remove_my_cache();
        printk(KERN_ALERT "Unloading slabex ...\n");
}

module_init(init);
module_exit(cleanup);
```