Project 1 Part II Reference Solution

Note that this is a nonworking code. You need to complete code before compiling it. However, the essential implementation is included. This will be helpful for practicing your skills by actually typing your own code rather than copying the solution.

```
struct birthday
{
        char* name;
    int month;
    int day;
    int year;

        struct list_head list;
};

/**
 * The following defines and initializes a list_head object named birthday_list
 */
static LIST_HEAD(birthday_list);

int simple_init(void)
{
    /* the pointer for memory allocation */
    struct birthday *person_one;

    /* the pointer for list traversal */
    struct birthday *ptr, *next;

        printk(KERN_INFO "Loading Module\n");

        person_one = kmalloc(sizeof(*person_one), GFP_KERNEL);
        person_one->name = "Alex";
        person_one->month = 8;
        person_one->day = 13;
        person_one->year = 1995;
    INIT_LIST_HEAD(&person_one->list);

        /* add the new node */
    list_add_tail(&person_one->list, &birthday_list);
```

Repeat this for 5 students

```
    /* now traverse the list */
    struct birthday *tmp;
    int tmp_year = -9999;
    int tmp_month = -9999;
    int tmp_day = -9999;


    list_for_each_entry(ptr, &birthday_list, list) {
        if (tmp_year < ptr->year)
        {
            tmp = ptr;
            tmp_year = ptr->year;
            tmp_month = ptr->month;
            tmp_day = ptr->day;
        }
        else if (tmp_year == ptr->year)
        {
            if (tmp_month < ptr->month){
                tmp = ptr;
                tmp_month = ptr->month;
                tmp_day = ptr->day;
            }
            else if (tmp_month == ptr->month)
                if (tmp_day < ptr->day){
                    tmp = ptr;
                    tmp_day = ptr->day;
                    //printk(KERN_INFO "%d-%d vs %d-%d", tmp_month, tmp_day, ptr->month, ptr->day);
                }

        }
        //else {printk (KERN_INFO "%d vs %d", tmp_year, ptr->year);}
        printk(KERN_INFO "Name: %s, Birthday: Month %d Day %d Year %d\n",ptr->name, ptr->month,ptr->day,ptr->year);
    }
    printk(KERN_INFO "Youngest person is: %s, Birthday: Month %d Day %d Year %d\n",tmp->name, tmp->month,tmp->day,tmp->year);

   //Delete the youngest student from the list
   list_for_each_entry_safe(ptr, next, &birthday_list, list){
      if(ptr == tmp){
         printk(KERN_INFO "Removed student: %s, Month %d, Day %d, Year %d",
                ptr->name,
                ptr->month,
                ptr->day,
                ptr->year);
         list_del(&ptr->list);
         kfree(ptr);
      }
   }
   //Print the updated list
   printk(KERN_INFO "Updated list");

   list_for_each_entry(ptr, &birthday_list, list){

      printk(KERN_INFO "%s, Birthday: Month %d Day %d, Year %d\n",
          ptr->name,
          ptr->month,
          ptr->day,
          ptr->year);
   }

      return 0;
}
```

Now you can implement the exit point simple_exit(void)