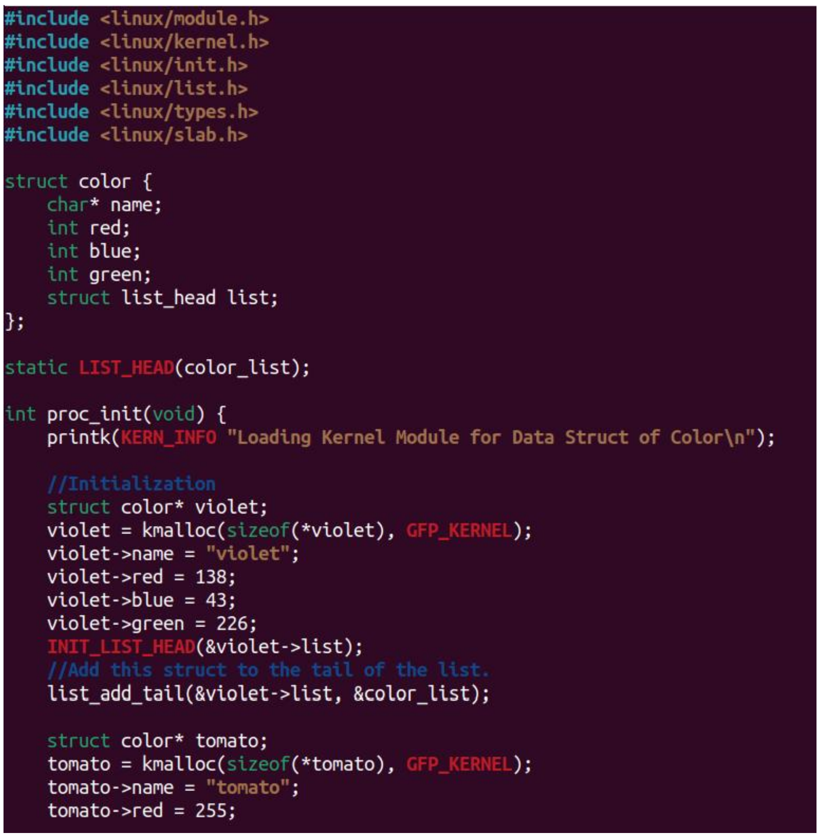
Report of Projects

Description:

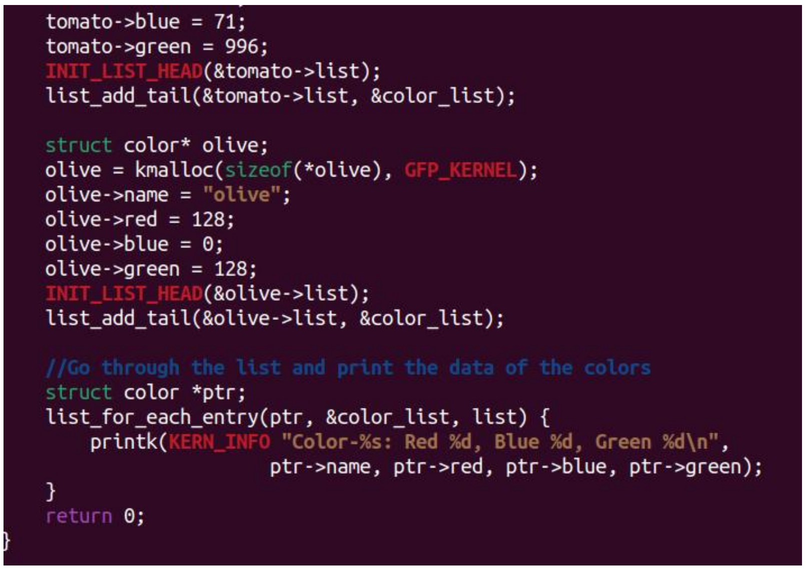
There are 2 projects: Kernel Data Structure and Multithreaded Sorting
And I will show you the codes, results, and explanation of each project.

Project 1: Kernel Data Structure (colors).

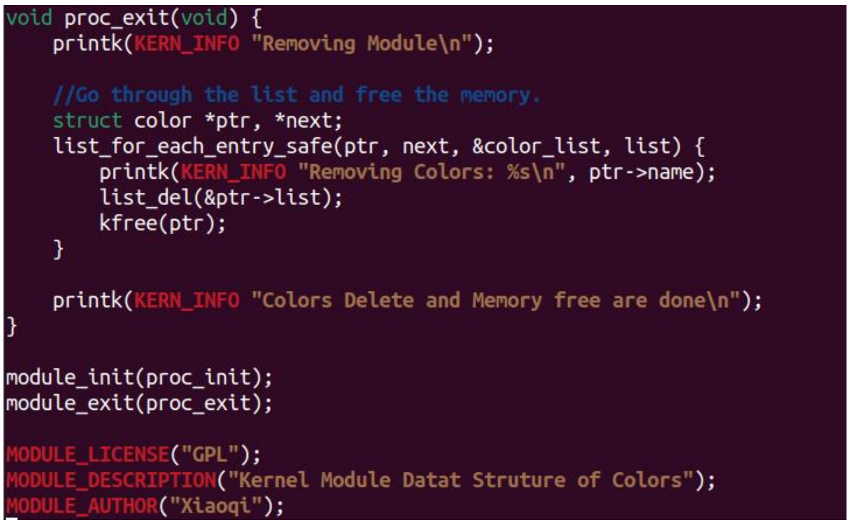
Codes:



```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/list.h>
#include <linux/types.h>
#include <linux/slab.h>

struct color {
    char* name;
    int red;
    int blue;
    int green;
    struct list_head list;
};

static LIST_HEAD(color_list);

int proc_init(void) {
    printk(KERN_INFO "Loading Kernel Module for Data Struct of Color\n");

    //Initialization
    struct color* violet;
    violet = kmalloc(sizeof(*violet), GFP_KERNEL);
    violet->name = "violet";
    violet->red = 138;
    violet->blue = 43;
    violet->green = 226;
    INIT_LIST_HEAD(&violet->list);
    //Add this struct to the tail of the list.
    list_add_tail(&violet->list, &color_list);

    struct color* tomato;
    tomato = kmalloc(sizeof(*tomato), GFP_KERNEL);
    tomato->name = "tomato";
    tomato->red = 255;
```



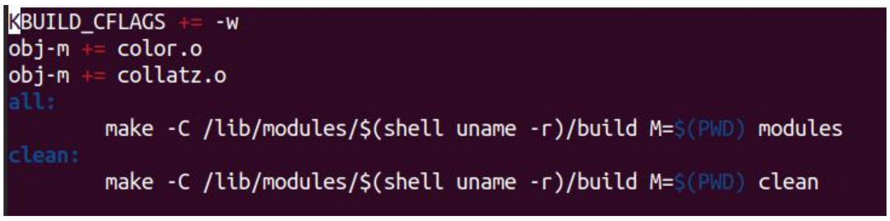
```c
    tomato->blue = 71;
    tomato->green = 996;
    INIT_LIST_HEAD(&tomato->list);
    list_add_tail(&tomato->list, &color_list);

    struct color* olive;
    olive = kmalloc(sizeof(*olive), GFP_KERNEL);
    olive->name = "olive";
    olive->red = 128;
    olive->blue = 0;
    olive->green = 128;
    INIT_LIST_HEAD(&olive->list);
    list_add_tail(&olive->list, &color_list);

    //Go through the list and print the data of the colors
    struct color *ptr;
    list_for_each_entry(ptr, &color_list, list) {
        printk(KERN_INFO "Color-%s: Red %d, Blue %d, Green %d\n",
                        ptr->name, ptr->red, ptr->blue, ptr->green);
    }
    return 0;
}
```



```
void proc_exit(void) {
    printk(KERN_INFO "Removing Module\n");

    //Go through the list and free the memory.
    struct color *ptr, *next;
    list_for_each_entry_safe(ptr, next, &color_list, list) {
        printk(KERN_INFO "Removing Colors: %s\n", ptr->name);
        list_del(&ptr->list);
        kfree(ptr);
    }

    printk(KERN_INFO "Colors Delete and Memory free are done\n");
}

module_init(proc_init);
module_exit(proc_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Kernel Module Datat Struture of Colors");
MODULE_AUTHOR("Xiaoqi");
```

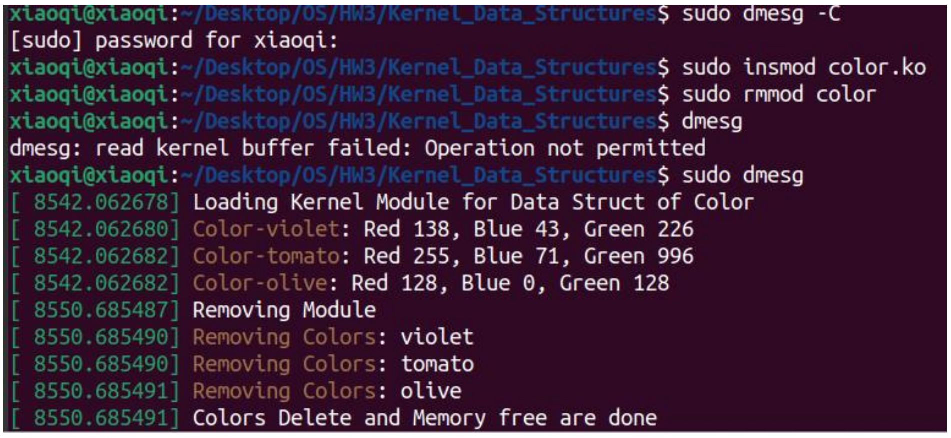
1) I stored 3 colors: violet, tomato, and olive in the proc_init() function, and used "list_add_tail()" to add it into the kernel linked list. The data of each color, name, red, blue, green values, will be showed after the proc exited.
2) I used "list_for_each_safe()" to travel through every nodes, deleted and freed memory at the same time.

Compile (Makefile):



```
KBUILD_CFLAGS += -w
obj-m += color.o
obj-m += collatz.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
```

Results:

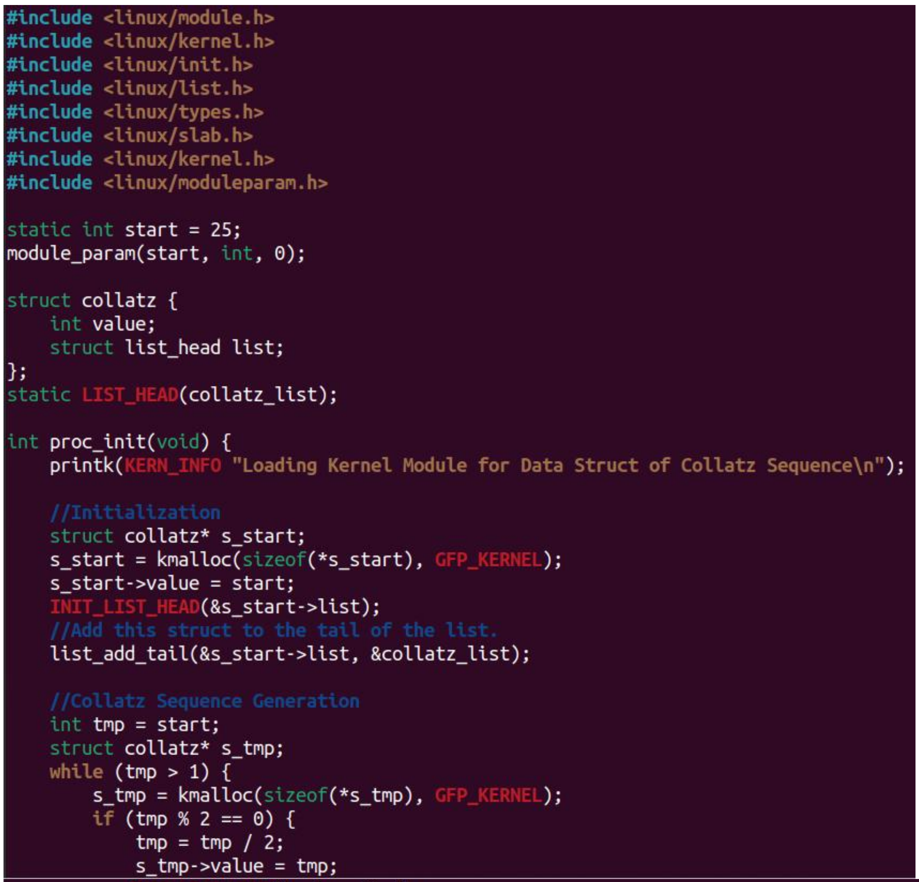


```
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo dmesg -C
[sudo] password for xiaoqi:
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo insmod color.ko
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo rmmod color
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ dmesg
dmesg: read kernel buffer failed: Operation not permitted
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo dmesg
[ 8542.062678] Loading Kernel Module for Data Struct of Color
[ 8542.062680] Color-violet: Red 138, Blue 43, Green 226
[ 8542.062682] Color-tomato: Red 255, Blue 71, Green 996
[ 8542.062682] Color-olive: Red 128, Blue 0, Green 128
[ 8550.685487] Removing Module
[ 8550.685490] Removing Colors: violet
[ 8550.685490] Removing Colors: tomato
[ 8550.685491] Removing Colors: olive
[ 8550.685491] Colors Delete and Memory free are done
```

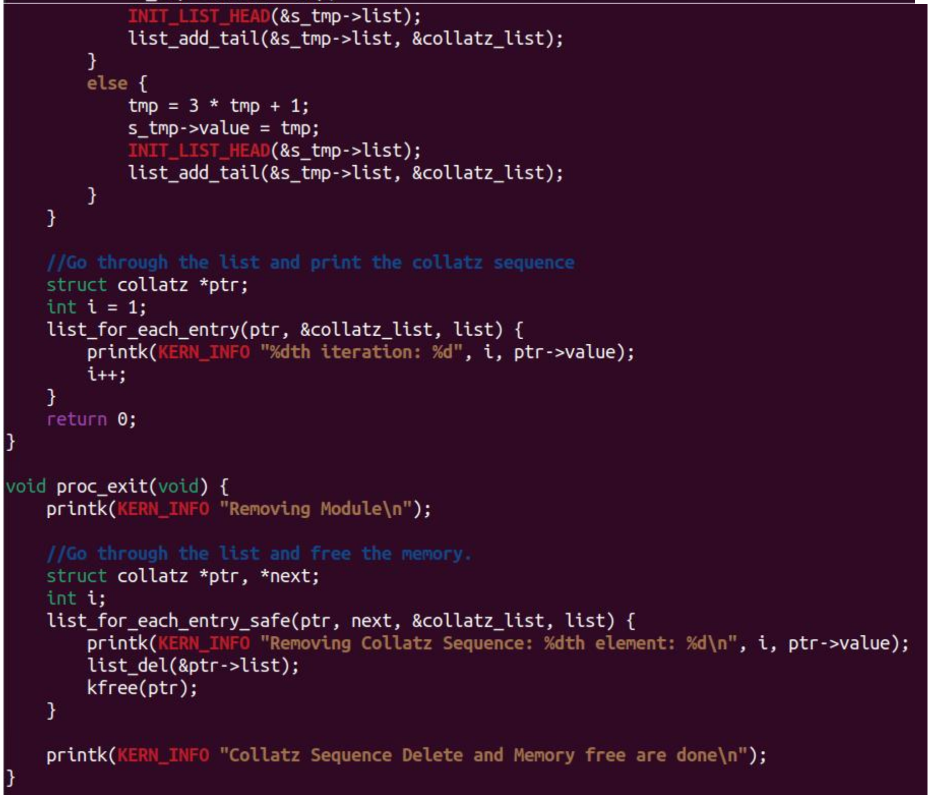
The results shown by dmesg.

Project 1: Kernel Data Structure (collatz sequences).

Codes:



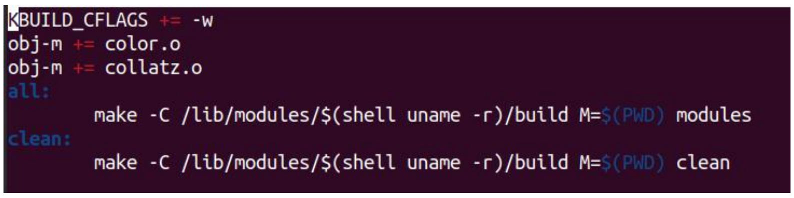
```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/list.h>
#include <linux/types.h>
#include <linux/slab.h>
#include <linux/kernel.h>
#include <linux/moduleparam.h>

static int start = 25;
module_param(start, int, 0);

struct collatz {
    int value;
    struct list_head list;
};
static LIST_HEAD(collatz_list);

int proc_init(void) {
    printk(KERN_INFO "Loading Kernel Module for Data Struct of Collatz Sequence\n");

    //Initialization
    struct collatz* s_start;
    s_start = kmalloc(sizeof(*s_start), GFP_KERNEL);
    s_start->value = start;
    INIT_LIST_HEAD(&s_start->list);
    //Add this struct to the tail of the list.
    list_add_tail(&s_start->list, &collatz_list);

    //Collatz Sequence Generation
    int tmp = start;
    struct collatz* s_tmp;
    while (tmp > 1) {
        s_tmp = kmalloc(sizeof(*s_tmp), GFP_KERNEL);
        if (tmp % 2 == 0) {
            tmp = tmp / 2;
            s_tmp->value = tmp;
```



```c
            INIT_LIST_HEAD(&s_tmp->list);
            list_add_tail(&s_tmp->list, &collatz_list);
        }
        else {
            tmp = 3 * tmp + 1;
            s_tmp->value = tmp;
            INIT_LIST_HEAD(&s_tmp->list);
            list_add_tail(&s_tmp->list, &collatz_list);
        }
    }

    //Go through the list and print the collatz sequence
    struct collatz *ptr;
    int i = 1;
    list_for_each_entry(ptr, &collatz_list, list) {
        printk(KERN_INFO "%dth iteration: %d", i, ptr->value);
        i++;
    }
    return 0;
}

void proc_exit(void) {
    printk(KERN_INFO "Removing Module\n");

    //Go through the list and free the memory.
    struct collatz *ptr, *next;
    int i;
    list_for_each_entry_safe(ptr, next, &collatz_list, list) {
        printk(KERN_INFO "Removing Collatz Sequence: %dth element: %d\n", i, ptr->value);
        list_del(&ptr->list);
        kfree(ptr);
    }

    printk(KERN_INFO "Collatz Sequence Delete and Memory free are done\n");
}
```

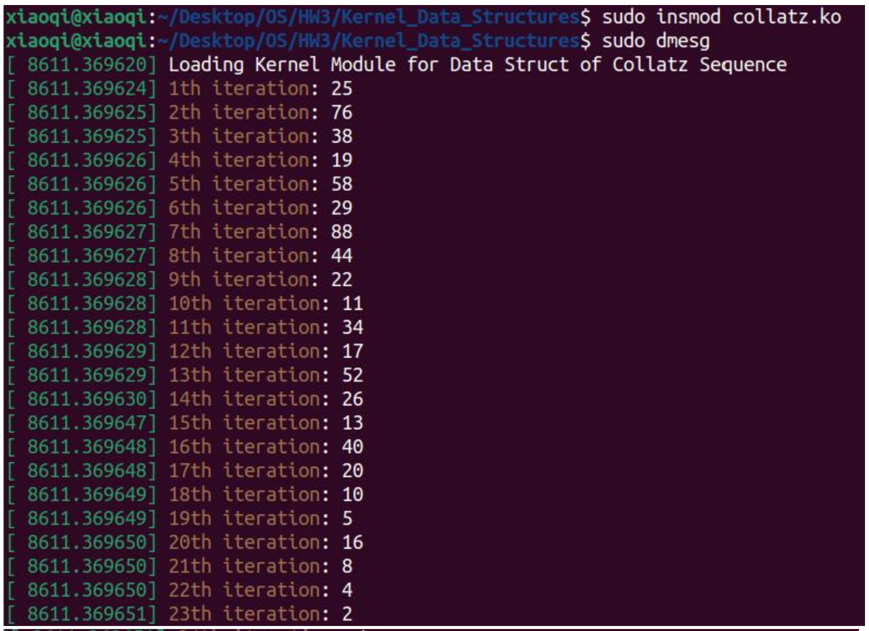
1) These codes are edited based on the above project (Data Structure of Colors).
2) The main part I made changes is the while loop in the proc_init()" fucntion. While the current value is greater than 1, I created a new node of collatz, and add it into the Linked List using "list_add_tail().

Compile (Makefile):

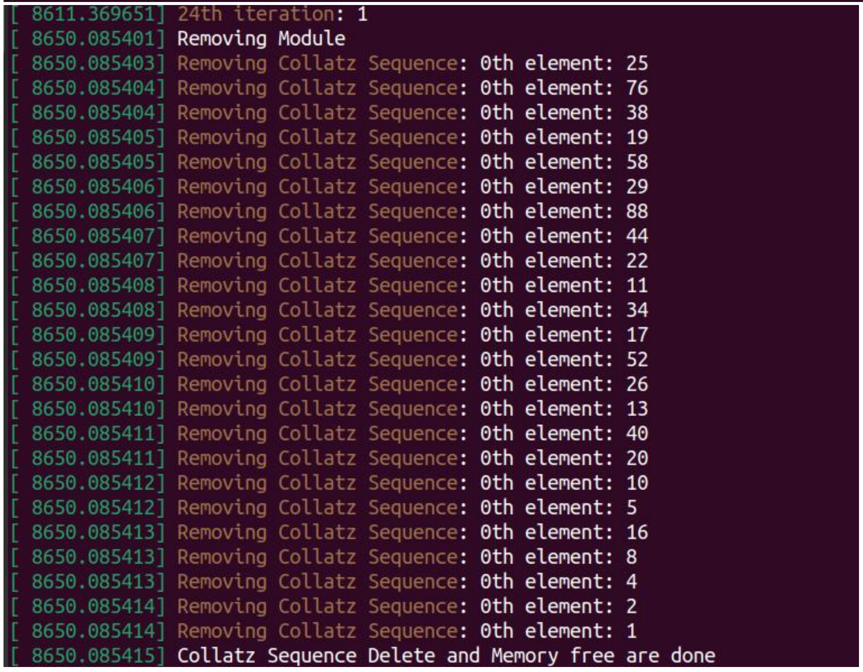


```
KBUILD_CFLAGS += -w
obj-m += color.o
obj-m += collatz.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Results (without kernel parameter):



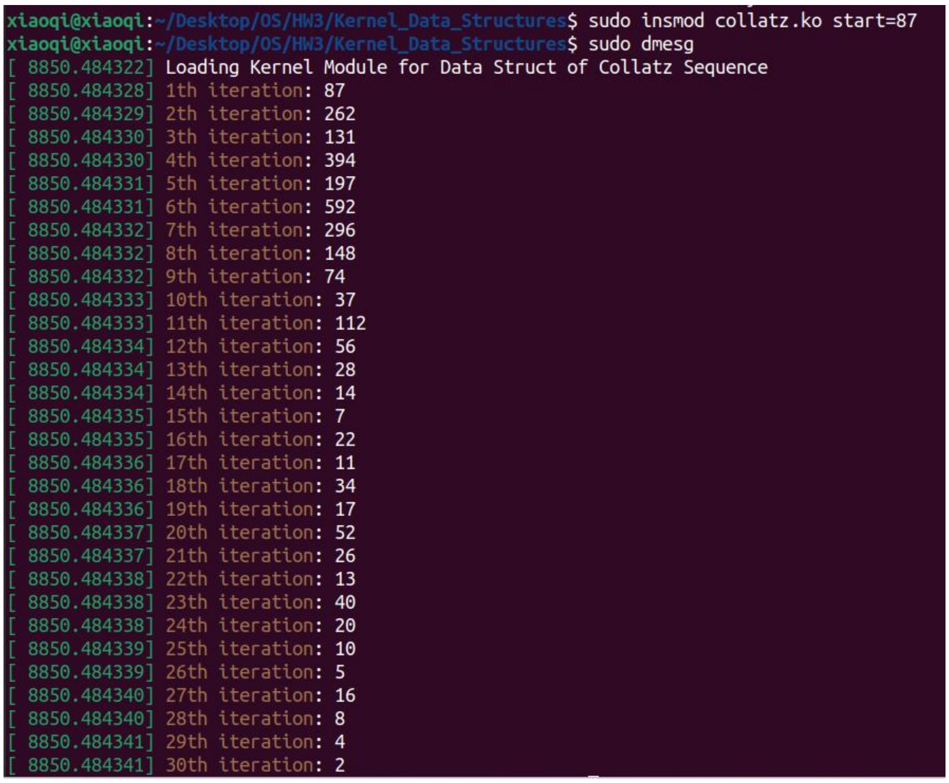
```
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo insmod collatz.ko
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo dmesg
[ 8611.369620] Loading Kernel Module for Data Struct of Collatz Sequence
[ 8611.369624] 1th iteration: 25
[ 8611.369625] 2th iteration: 76
[ 8611.369625] 3th iteration: 38
[ 8611.369626] 4th iteration: 19
[ 8611.369626] 5th iteration: 58
[ 8611.369626] 6th iteration: 29
[ 8611.369627] 7th iteration: 88
[ 8611.369627] 8th iteration: 44
[ 8611.369628] 9th iteration: 22
[ 8611.369628] 10th iteration: 11
[ 8611.369628] 11th iteration: 34
[ 8611.369629] 12th iteration: 17
[ 8611.369629] 13th iteration: 52
[ 8611.369630] 14th iteration: 26
[ 8611.369647] 15th iteration: 13
[ 8611.369648] 16th iteration: 40
[ 8611.369648] 17th iteration: 20
[ 8611.369649] 18th iteration: 10
[ 8611.369649] 19th iteration: 5
[ 8611.369650] 20th iteration: 16
[ 8611.369650] 21th iteration: 8
[ 8611.369650] 22th iteration: 4
[ 8611.369651] 23th iteration: 2
```



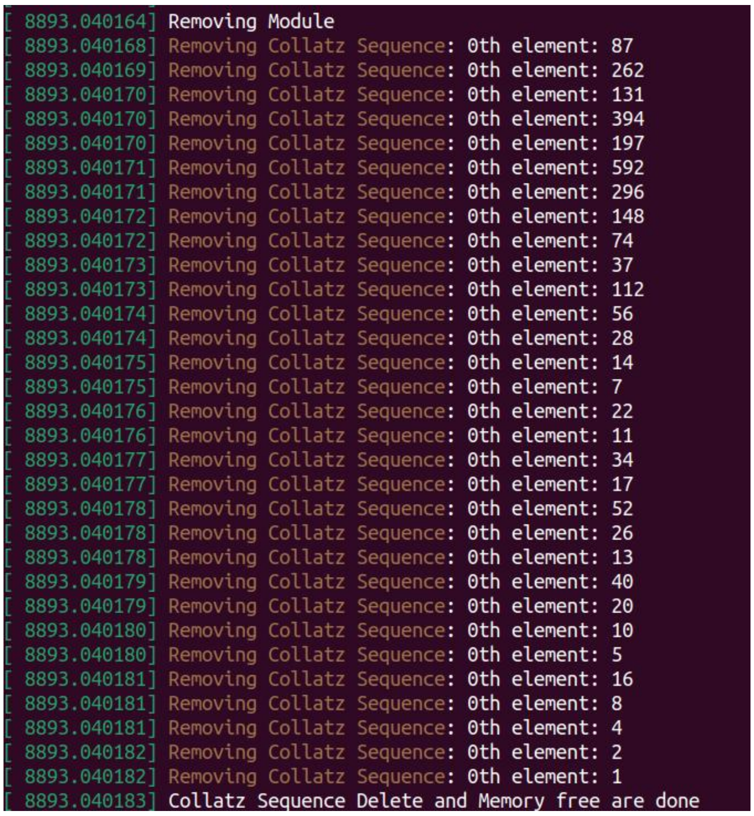
```
[ 8611.369651] 24th iteration: 1
[ 8650.085401] Removing Module
[ 8650.085403] Removing Collatz Sequence: 0th element: 25
[ 8650.085404] Removing Collatz Sequence: 0th element: 76
[ 8650.085404] Removing Collatz Sequence: 0th element: 38
[ 8650.085405] Removing Collatz Sequence: 0th element: 19
[ 8650.085405] Removing Collatz Sequence: 0th element: 58
[ 8650.085406] Removing Collatz Sequence: 0th element: 29
[ 8650.085406] Removing Collatz Sequence: 0th element: 88
[ 8650.085407] Removing Collatz Sequence: 0th element: 44
[ 8650.085407] Removing Collatz Sequence: 0th element: 22
[ 8650.085408] Removing Collatz Sequence: 0th element: 11
[ 8650.085408] Removing Collatz Sequence: 0th element: 34
[ 8650.085409] Removing Collatz Sequence: 0th element: 17
[ 8650.085409] Removing Collatz Sequence: 0th element: 52
[ 8650.085410] Removing Collatz Sequence: 0th element: 26
[ 8650.085410] Removing Collatz Sequence: 0th element: 13
[ 8650.085411] Removing Collatz Sequence: 0th element: 40
[ 8650.085411] Removing Collatz Sequence: 0th element: 20
[ 8650.085412] Removing Collatz Sequence: 0th element: 10
[ 8650.085412] Removing Collatz Sequence: 0th element: 5
[ 8650.085413] Removing Collatz Sequence: 0th element: 16
[ 8650.085413] Removing Collatz Sequence: 0th element: 8
[ 8650.085413] Removing Collatz Sequence: 0th element: 4
[ 8650.085414] Removing Collatz Sequence: 0th element: 2
[ 8650.085414] Removing Collatz Sequence: 0th element: 1
[ 8650.085415] Collatz Sequence Delete and Memory free are done
```

1) The results here is for the command "sudo insmod collatz.ko", note that here doesn't have a kernel parameter, and the default value of start is 25.
2) The insmod, and rmmod results of collatz sequences of value 25 with 24 iterations are listed here.

Results (with kernel parameter):



```
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo insmod collatz.ko start=87
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Kernel_Data_Structures$ sudo dmesg
[ 8850.484322] Loading Kernel Module for Data Struct of Collatz Sequence
[ 8850.484328] 1th iteration: 87
[ 8850.484329] 2th iteration: 262
[ 8850.484330] 3th iteration: 131
[ 8850.484330] 4th iteration: 394
[ 8850.484331] 5th iteration: 197
[ 8850.484331] 6th iteration: 592
[ 8850.484332] 7th iteration: 296
[ 8850.484332] 8th iteration: 148
[ 8850.484332] 9th iteration: 74
[ 8850.484333] 10th iteration: 37
[ 8850.484333] 11th iteration: 112
[ 8850.484334] 12th iteration: 56
[ 8850.484334] 13th iteration: 28
[ 8850.484334] 14th iteration: 14
[ 8850.484335] 15th iteration: 7
[ 8850.484335] 16th iteration: 22
[ 8850.484336] 17th iteration: 11
[ 8850.484336] 18th iteration: 34
[ 8850.484336] 19th iteration: 17
[ 8850.484337] 20th iteration: 52
[ 8850.484337] 21th iteration: 26
[ 8850.484338] 22th iteration: 13
[ 8850.484338] 23th iteration: 40
[ 8850.484338] 24th iteration: 20
[ 8850.484339] 25th iteration: 10
[ 8850.484339] 26th iteration: 5
[ 8850.484340] 27th iteration: 16
[ 8850.484340] 28th iteration: 8
[ 8850.484341] 29th iteration: 4
[ 8850.484341] 30th iteration: 2
```

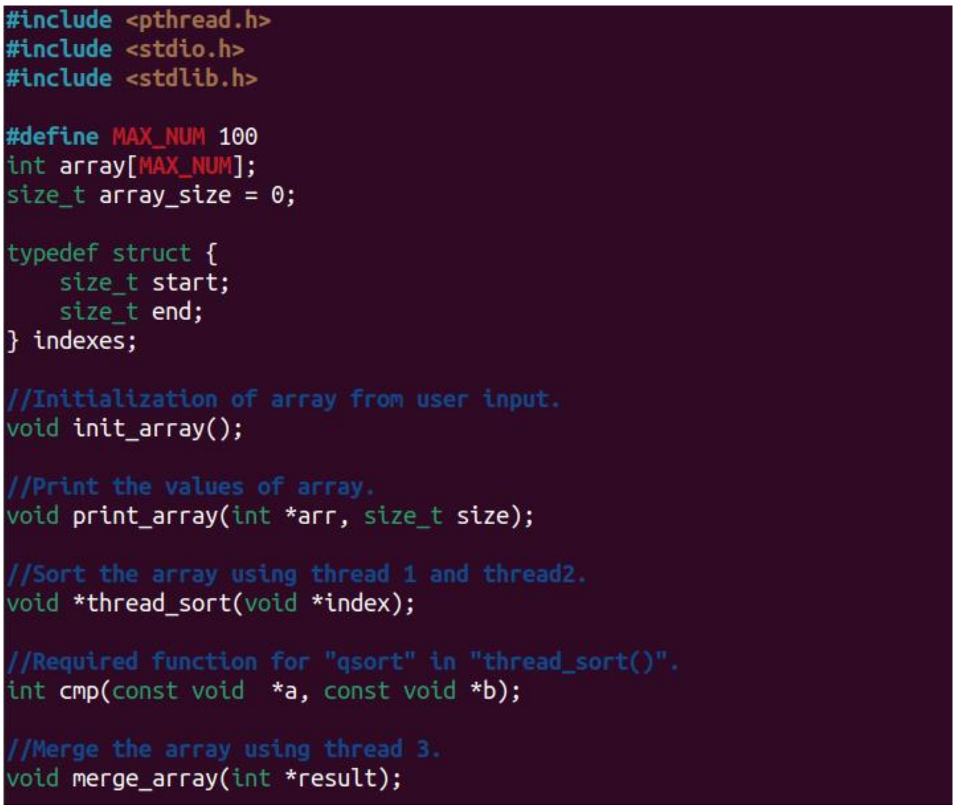


```
[ 8893.040164] Removing Module
[ 8893.040168] Removing Collatz Sequence: 0th element: 87
[ 8893.040169] Removing Collatz Sequence: 0th element: 262
[ 8893.040170] Removing Collatz Sequence: 0th element: 131
[ 8893.040170] Removing Collatz Sequence: 0th element: 394
[ 8893.040170] Removing Collatz Sequence: 0th element: 197
[ 8893.040171] Removing Collatz Sequence: 0th element: 592
[ 8893.040171] Removing Collatz Sequence: 0th element: 296
[ 8893.040172] Removing Collatz Sequence: 0th element: 148
[ 8893.040172] Removing Collatz Sequence: 0th element: 74
[ 8893.040173] Removing Collatz Sequence: 0th element: 37
[ 8893.040173] Removing Collatz Sequence: 0th element: 112
[ 8893.040174] Removing Collatz Sequence: 0th element: 56
[ 8893.040174] Removing Collatz Sequence: 0th element: 28
[ 8893.040175] Removing Collatz Sequence: 0th element: 14
[ 8893.040175] Removing Collatz Sequence: 0th element: 7
[ 8893.040176] Removing Collatz Sequence: 0th element: 22
[ 8893.040176] Removing Collatz Sequence: 0th element: 11
[ 8893.040177] Removing Collatz Sequence: 0th element: 34
[ 8893.040177] Removing Collatz Sequence: 0th element: 17
[ 8893.040178] Removing Collatz Sequence: 0th element: 52
[ 8893.040178] Removing Collatz Sequence: 0th element: 26
[ 8893.040178] Removing Collatz Sequence: 0th element: 13
[ 8893.040179] Removing Collatz Sequence: 0th element: 40
[ 8893.040179] Removing Collatz Sequence: 0th element: 20
[ 8893.040180] Removing Collatz Sequence: 0th element: 10
[ 8893.040180] Removing Collatz Sequence: 0th element: 5
[ 8893.040181] Removing Collatz Sequence: 0th element: 16
[ 8893.040181] Removing Collatz Sequence: 0th element: 8
[ 8893.040181] Removing Collatz Sequence: 0th element: 4
[ 8893.040182] Removing Collatz Sequence: 0th element: 2
[ 8893.040182] Removing Collatz Sequence: 0th element: 1
[ 8893.040183] Collatz Sequence Delete and Memory free are done
```

1) The results here is for the command “sudo insmod collatz.ko start=87”, note that here have a kernel parameter “start=87”, so the value of start will be changed..
2) The insmod, and rmmod results of collatz sequences of value 87 with 31 iterations are listed here.

Project 2: Multithreaded Sorting.

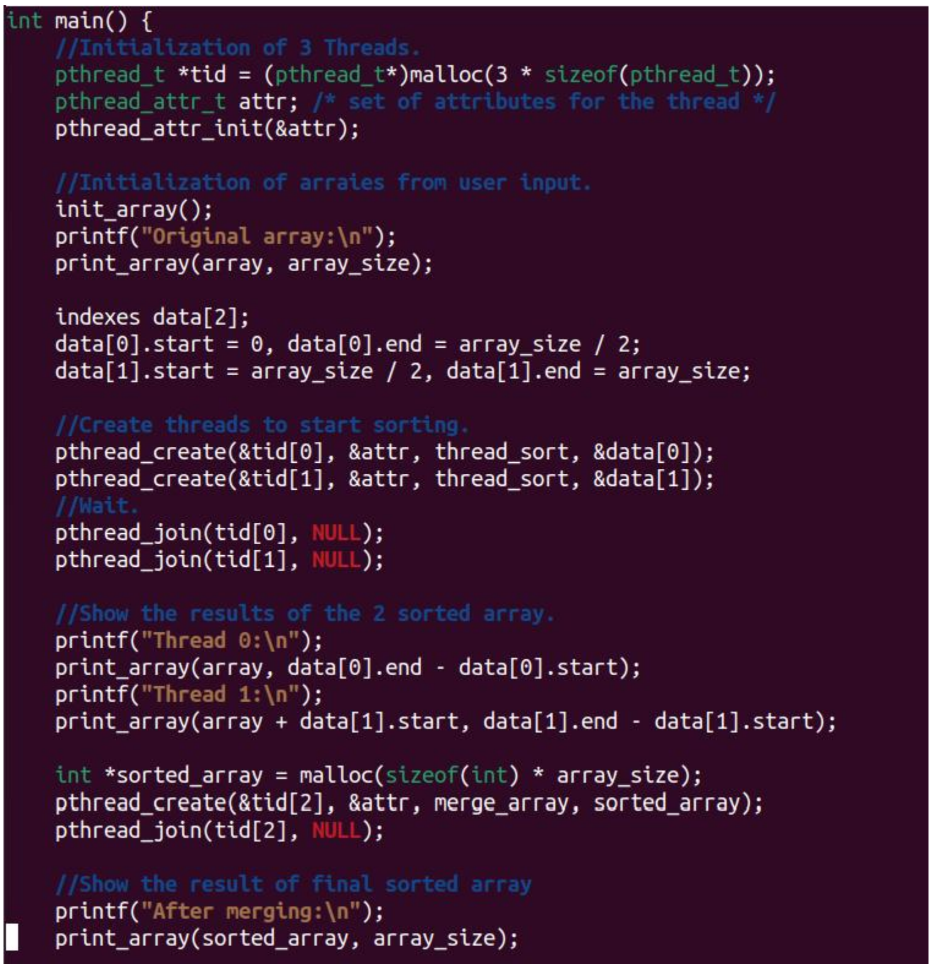
Functions:



```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define MAX_NUM 100
int array[MAX_NUM];
size_t array_size = 0;

typedef struct {
    size_t start;
    size_t end;
} indexes;

//Initialization of array from user input.
void init_array();

//Print the values of array.
void print_array(int *arr, size_t size);

//Sort the array using thread 1 and thread2.
void *thread_sort(void *index);

//Required function for "qsort" in "thread_sort()".
int cmp(const void  *a, const void *b);

//Merge the array using thread 3.
void merge_array(int *result);
```

As you can see, I put the “blue” explanation on each of the functions.

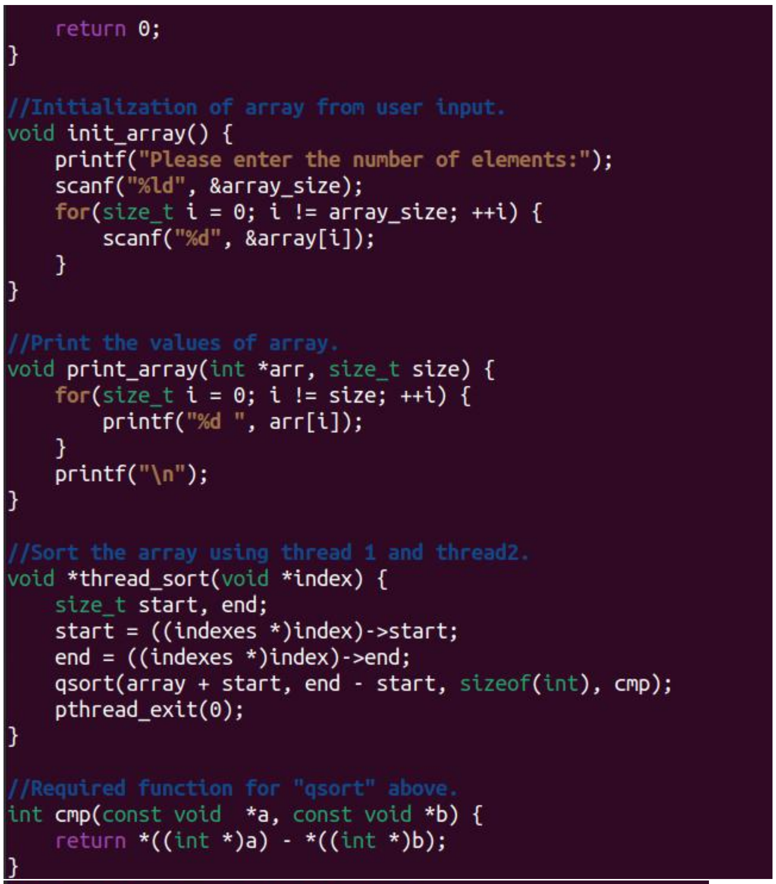
Main:



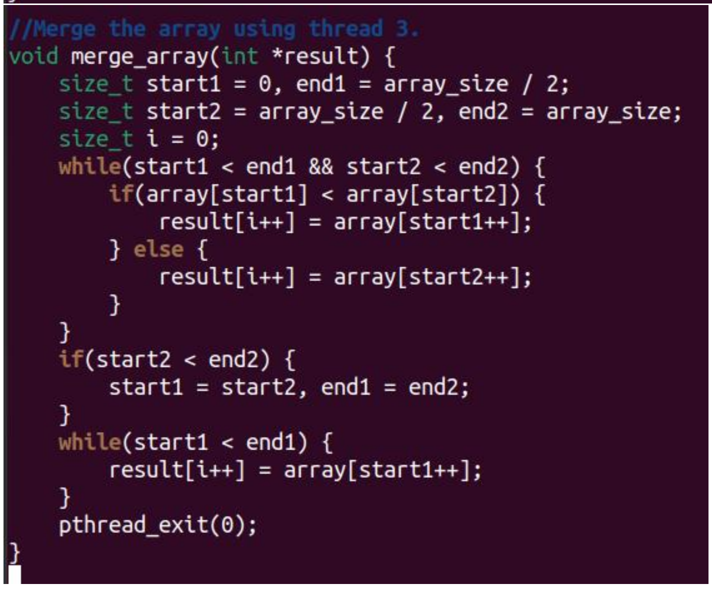
```
int main() {
    //Initialization of 3 Threads.
    pthread_t *tid = (pthread_t*)malloc(3 * sizeof(pthread_t));
    pthread_attr_t attr; /* set of attributes for the thread */
    pthread_attr_init(&attr);

    //Initialization of arraies from user input.
    init_array();
    printf("Original array:\n");
    print_array(array, array_size);

    indexes data[2];
    data[0].start = 0, data[0].end = array_size / 2;
    data[1].start = array_size / 2, data[1].end = array_size;

    //Create threads to start sorting.
    pthread_create(&tid[0], &attr, thread_sort, &data[0]);
    pthread_create(&tid[1], &attr, thread_sort, &data[1]);
    //Wait.
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);

    //Show the results of the 2 sorted array.
    printf("Thread 0:\n");
    print_array(array, data[0].end - data[0].start);
    printf("Thread 1:\n");
    print_array(array + data[1].start, data[1].end - data[1].start);

    int *sorted_array = malloc(sizeof(int) * array_size);
    pthread_create(&tid[2], &attr, merge_array, sorted_array);
    pthread_join(tid[2], NULL);

    //Show the result of final sorted array
    printf("After merging:\n");
    print_array(sorted_array, array_size);
```

1) Firstly, I created 3 threads using malloc, the first 2 of threads are used for sorting the array, and the third one is used to merging the 2 sorted array.
2) Then I initialized the array from the user input, and created a structure to store the indexes of the 2 array to be sorted.
3) I used “pthread_creat()” to start sorting using the first 2 threads by the function “thread_sort()” and “pthread_join()” to wait for them to be finished.
4) I used the third thread to execute the function “merge_array()” to merge the 2 sorted array together and get the final sorted array.

Other Codes:



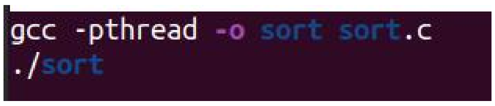
```c
    return 0;
}

//Initialization of array from user input.
void init_array() {
    printf("Please enter the number of elements:");
    scanf("%ld", &array_size);
    for(size_t i = 0; i != array_size; ++i) {
        scanf("%d", &array[i]);
    }
}

//Print the values of array.
void print_array(int *arr, size_t size) {
    for(size_t i = 0; i != size; ++i) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

//Sort the array using thread 1 and thread2.
void *thread_sort(void *index) {
    size_t start, end;
    start = ((indexes *)index)->start;
    end = ((indexes *)index)->end;
    qsort(array + start, end - start, sizeof(int), cmp);
    pthread_exit(0);
}

//Required function for "qsort" above.
int cmp(const void  *a, const void *b) {
    return *((int *)a) - *((int *)b);
}
```



```c
//Merge the array using thread 3.
void merge_array(int *result) {
    size_t start1 = 0, end1 = array_size / 2;
    size_t start2 = array_size / 2, end2 = array_size;
    size_t i = 0;
    while(start1 < end1 && start2 < end2) {
        if(array[start1] < array[start2]) {
            result[i++] = array[start1++];
        } else {
            result[i++] = array[start2++];
        }
    }
    if(start2 < end2) {
        start1 = start2, end1 = end2;
    }
    while(start1 < end1) {
        result[i++] = array[start1++];
    }
    pthread_exit(0);
}
```

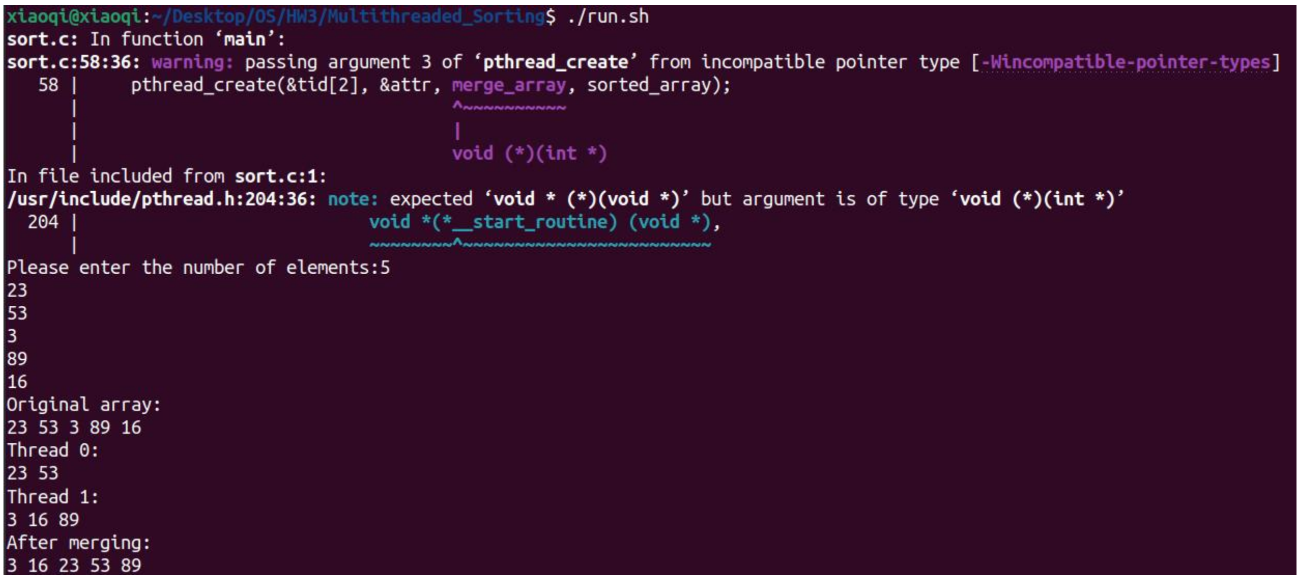
1) Implementation of the rest 4 function: init_array(), print_array(), thread_sort(), and “merge_array()”
2) For the algorithm to sort array in the function “thread_sort()”, I used a default defined function in C language, which is “qsort()”, and the function “cmp()” is an needed parameter function for “qsort()”.

Compile (run.sh):



```
gcc -pthread -o sort sort.c
./sort
```

Results:



```
xiaoqi@xiaoqi:~/Desktop/OS/HW3/Multithreaded_Sorting$ ./run.sh
sort.c: In function ‘main’:
sort.c:58:36: warning: passing argument 3 of ‘pthread_create’ from incompatible pointer type [-Wincompatible-pointer-types]
   58 |     pthread_create(&tid[2], &attr, merge_array, sorted_array);
      |                                    ^~~~~~~~~~~
      |                                    |
      |                                    void (*)(int *)
In file included from sort.c:1:
/usr/include/pthread.h:204:36: note: expected ‘void * (*)(void *)’ but argument is of type ‘void (*)(int *)’
  204 |                            void *(*__start_routine) (void *),
      |                            ~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~
Please enter the number of elements:5
23
53
3
89
16
Original array:
23 53 3 89 16
Thread 0:
23 53
Thread 1:
3 16 89
After merging:
3 16 23 53 89
```

1) This is the compilation and result of this program.
2) Firstly, it requires the number of elements that you want. I set 5 here, and gave a list of 5 numbers: 23, 53, 3 89, 16.
3) The original array, sorted array by thread 0, sorted array by thread 1, and final merged sorted array by thread 2 are all listed here. In this way, it gives you a obvious sense of what my program worked.

That's the end of this report, thank you very much for your attention!
Xiaoqi LIU 999009335