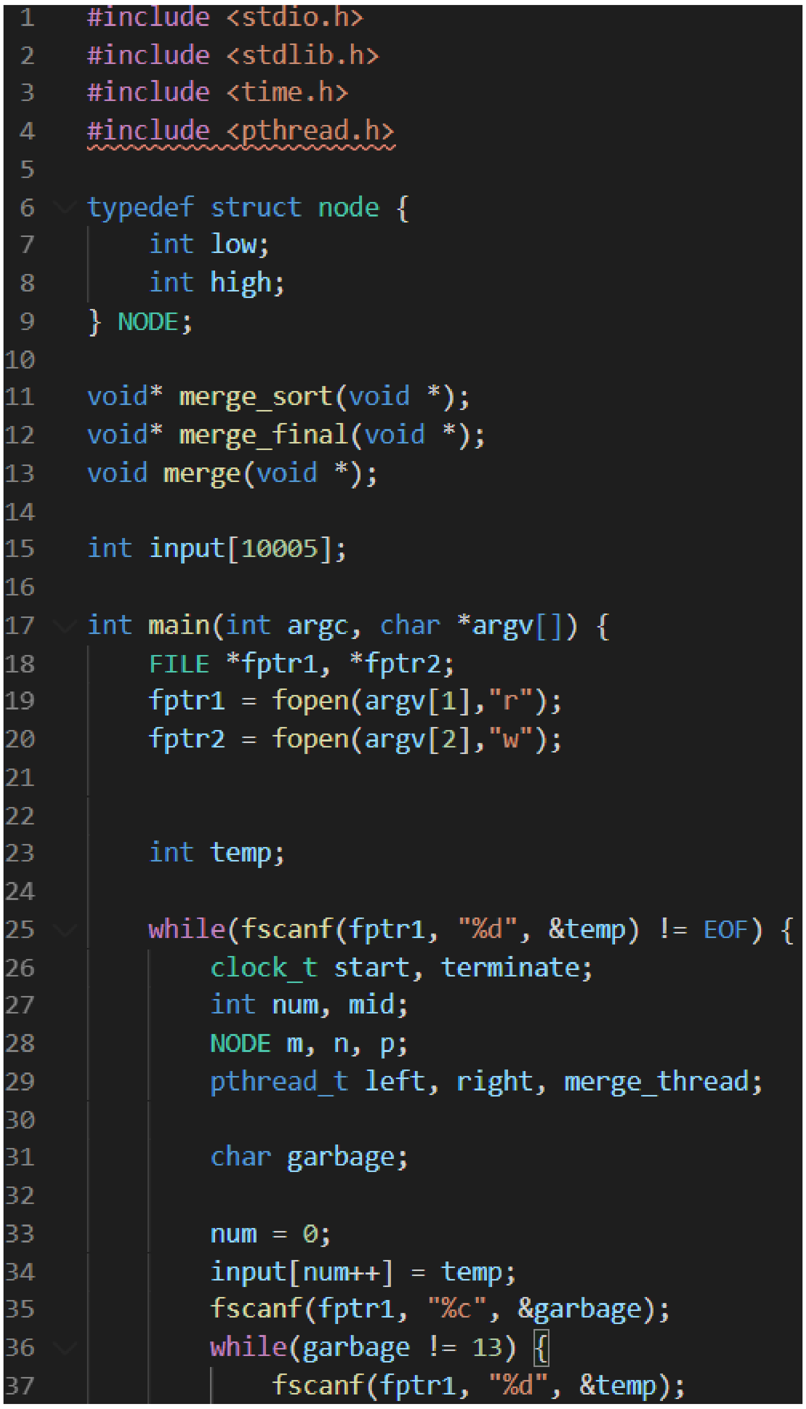
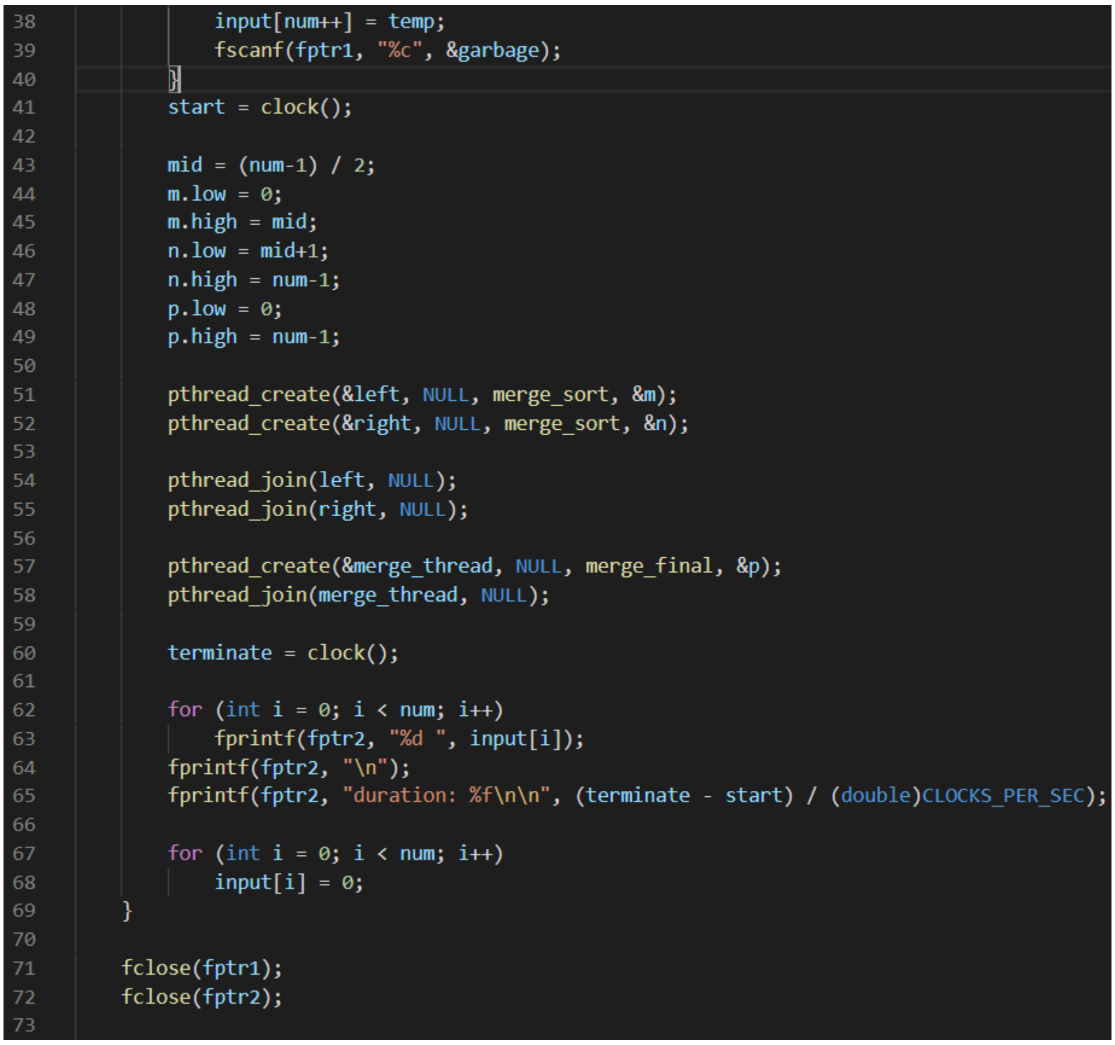
106030012 廖昱瑋 工院21 HW3

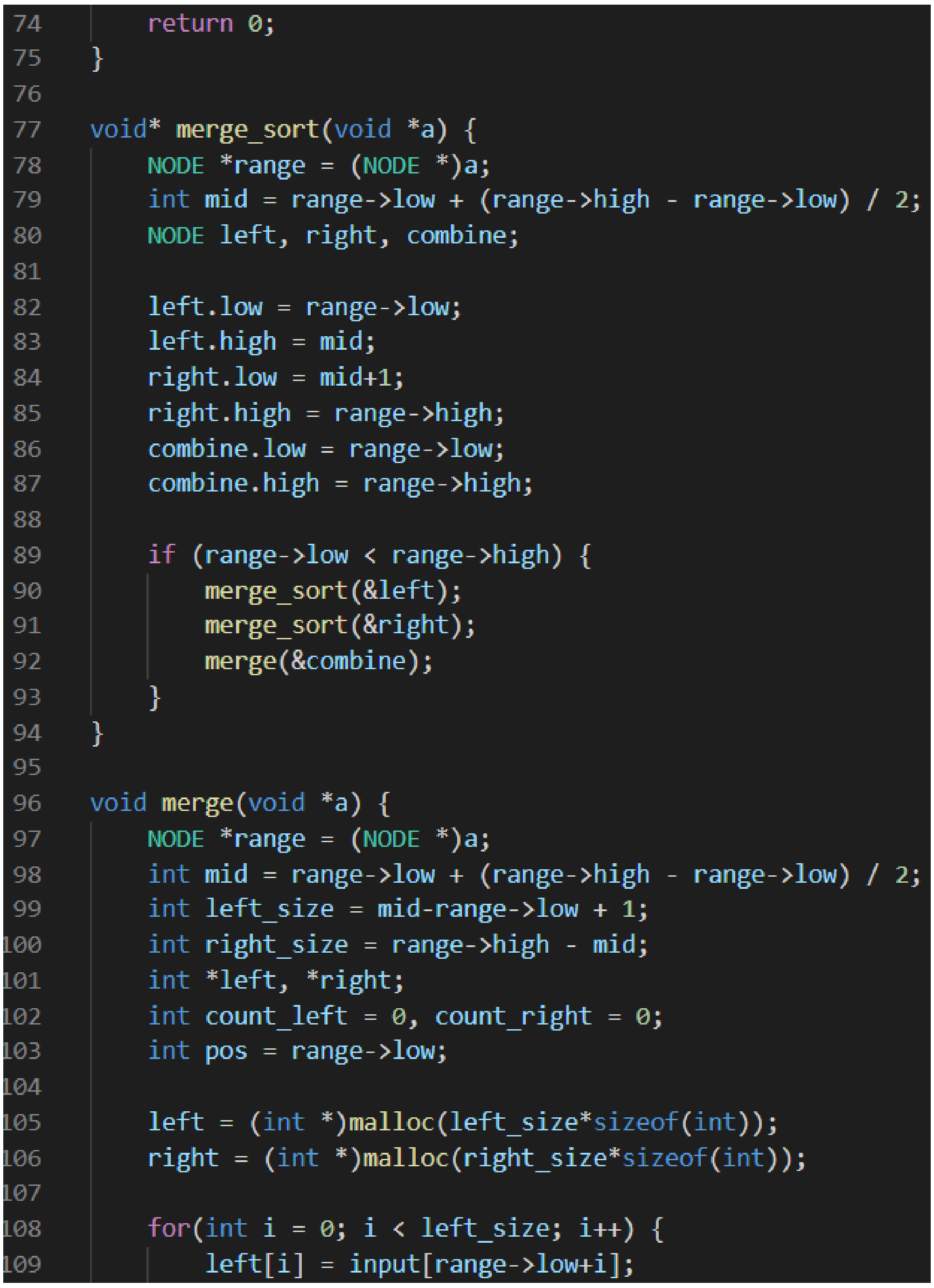
一、code 截圖



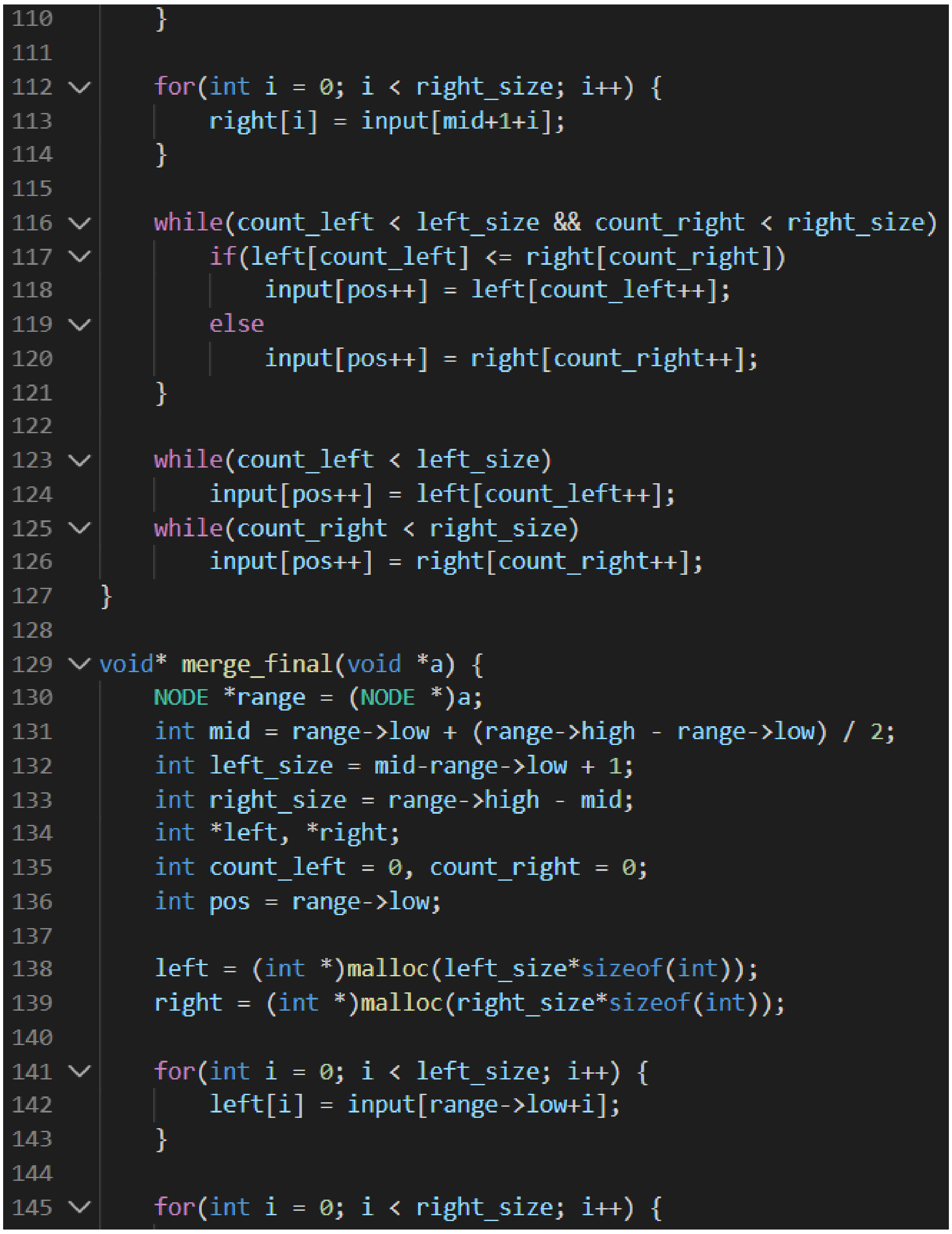
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>

typedef struct node {
    int low;
    int high;
} NODE;

void* merge_sort(void *);
void* merge_final(void *);
void merge(void *);

int input[10005];

int main(int argc, char *argv[]) {
    FILE *fptr1, *fptr2;
    fptr1 = fopen(argv[1],"r");
    fptr2 = fopen(argv[2],"w");


    int temp;

    while(fscanf(fptr1, "%d", &temp) != EOF) {
        clock_t start, terminate;
        int num, mid;
        NODE m, n, p;
        pthread_t left, right, merge_thread;

        char garbage;

        num = 0;
        input[num++] = temp;
        fscanf(fptr1, "%c", &garbage);
        while(garbage != 13) {
            fscanf(fptr1, "%d", &temp);
```



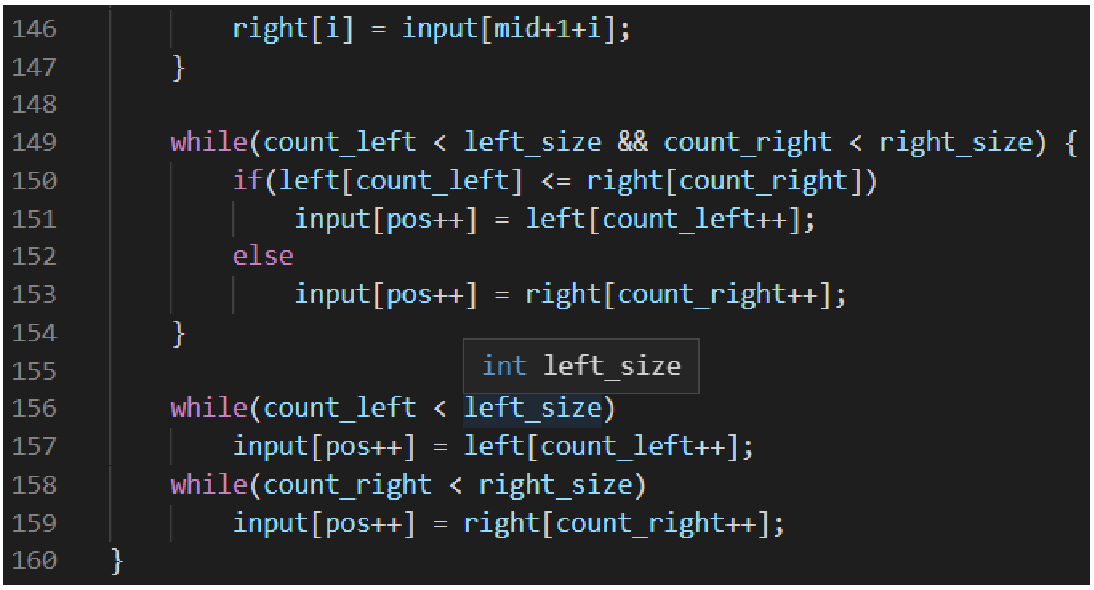
```c
            input[num++] = temp;
            fscanf(fptr1, "%c", &garbage);
        }
        start = clock();

        mid = (num-1) / 2;
        m.low = 0;
        m.high = mid;
        n.low = mid+1;
        n.high = num-1;
        p.low = 0;
        p.high = num-1;

        pthread_create(&left, NULL, merge_sort, &m);
        pthread_create(&right, NULL, merge_sort, &n);

        pthread_join(left, NULL);
        pthread_join(right, NULL);

        pthread_create(&merge_thread, NULL, merge_final, &p);
        pthread_join(merge_thread, NULL);

        terminate = clock();

        for (int i = 0; i < num; i++)
            fprintf(fptr2, "%d ", input[i]);
        fprintf(fptr2, "\n");
        fprintf(fptr2, "duration: %f\n\n", (terminate - start) / (double)CLOCKS_PER_SEC);

        for (int i = 0; i < num; i++)
            input[i] = 0;
    }

    fclose(fptr1);
    fclose(fptr2);

```



```c
    return 0;
}

void* merge_sort(void *a) {
    NODE *range = (NODE *)a;
    int mid = range->low + (range->high - range->low) / 2;
    NODE left, right, combine;

    left.low = range->low;
    left.high = mid;
    right.low = mid+1;
    right.high = range->high;
    combine.low = range->low;
    combine.high = range->high;

    if (range->low < range->high) {
        merge_sort(&left);
        merge_sort(&right);
        merge(&combine);
    }
}

void merge(void *a) {
    NODE *range = (NODE *)a;
    int mid = range->low + (range->high - range->low) / 2;
    int left_size = mid-range->low + 1;
    int right_size = range->high - mid;
    int *left, *right;
    int count_left = 0, count_right = 0;
    int pos = range->low;

    left = (int *)malloc(left_size*sizeof(int));
    right = (int *)malloc(right_size*sizeof(int));

    for(int i = 0; i < left_size; i++) {
        left[i] = input[range->low+i];
```



```c
    }

    for(int i = 0; i < right_size; i++) {
        right[i] = input[mid+1+i];
    }

    while(count_left < left_size && count_right < right_size)
        if(left[count_left] <= right[count_right])
            input[pos++] = left[count_left++];
        else
            input[pos++] = right[count_right++];
    }

    while(count_left < left_size)
        input[pos++] = left[count_left++];
    while(count_right < right_size)
        input[pos++] = right[count_right++];
}

void* merge_final(void *a) {
    NODE *range = (NODE *)a;
    int mid = range->low + (range->high - range->low) / 2;
    int left_size = mid-range->low + 1;
    int right_size = range->high - mid;
    int *left, *right;
    int count_left = 0, count_right = 0;
    int pos = range->low;

    left = (int *)malloc(left_size*sizeof(int));
    right = (int *)malloc(right_size*sizeof(int));

    for(int i = 0; i < left_size; i++) {
        left[i] = input[range->low+i];
    }

    for(int i = 0; i < right_size; i++) {
```

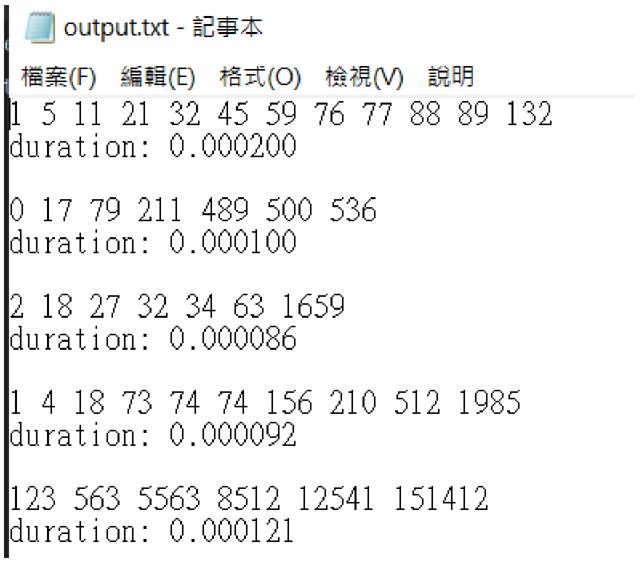

```
        right[i] = input[mid+1+i];
    }

    while(count_left < left_size && count_right < right_size) {
        if(left[count_left] <= right[count_right])
            input[pos++] = left[count_left++];
        else
            input[pos++] = right[count_right++];
    }
                            int left_size
    while(count_left < left_size)
        input[pos++] = left[count_left++];
    while(count_right < right_size)
        input[pos++] = right[count_right++];
}
```

二、說明

宣告一個 global variable “input”，其為 10005 個 entries 的 int 矩陣，用以存放每一行 testcase。宣告一個叫 NODE 的 structure，裡面有兩個 int，分別存 merge sort 時範圍上、下界的 index。宣告三個 function，void* merge_sort(void *) 、 void* merge_final(void *) 及 void merge(void *)，merge_sort 跟 merge_final 用來執行 thread，merge_sort 是給剛開始的兩個 threads 執行，那兩個 thread 結束後再開第三個 thread，用 merge_final 把前面的結果 merge 起來。而 merge 是在執行 merge_sort 時 recursion 會用到的 function，用來將左、右兩半邊的 testcase 分別排序好。

在 main 裡，先檢查是不是 EOF，不是的話，先用 clock_t 紀錄開始時間，並且讀一行 testcase，再來把 testcase 分為左右兩段的上、下界 index 紀錄好，創造兩個 threads 分別叫做 left、right，透過 merge_sort function 對 left、right 進行 merge sort，做完後結束這兩個 threads。再 create 第三個 thread 叫做 merge_thread，透過 merge_final function 將前面兩個 threads 運算結果 merge 起來，做完後結束第三個 thread。接著，紀錄運算好的時間並 print 結果。然俟將 input array 重新 initiate。重複以上動作直至遇到 EOF。

三、結果截圖



```
1 5 11 21 32 45 59 76 77 88 89 132
duration: 0.000200

0 17 79 211 489 500 536
duration: 0.000100

2 18 27 32 34 63 1659
duration: 0.000086

1 4 18 73 74 74 156 210 512 1985
duration: 0.000092

123 563 5563 8512 12541 151412
duration: 0.000121
```