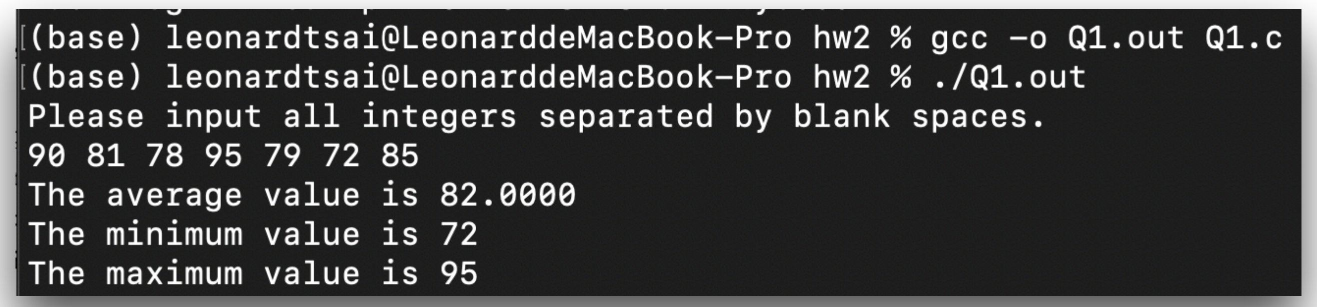
2023 Spring OS-HW2 Report

資管二 蔡可亮 B10705010

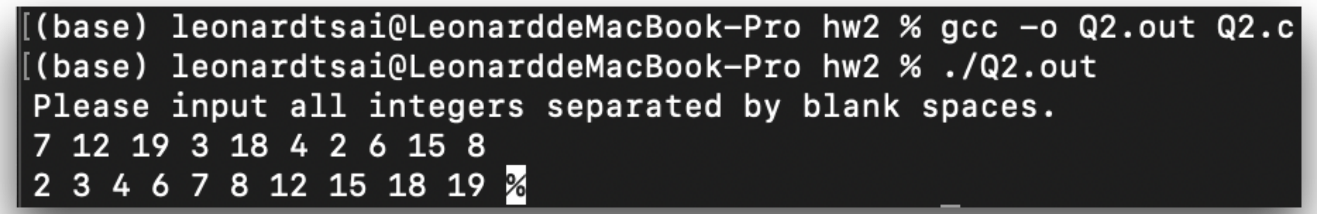
Q1.



```
[(base) leonardtsai@LeonarddeMacBook-Pro hw2 % gcc -o Q1.out Q1.c
[(base) leonardtsai@LeonarddeMacBook-Pro hw2 % ./Q1.out
Please input all integers separated by blank spaces.
90 81 78 95 79 72 85
The average value is 82.0000
The minimum value is 72
The maximum value is 95
```

1. First, input the starting number in terminal.
2. Use `strtok` to extract all integers in the input string. And store them in `parameters` which would be passed to threads.
3. Initialize `pthread_t`, `pthread_attr_t`. Use `pthread_create` to create three threads including calculating the average value, minimum value, maximum value with three thread functions (thread_average, thread_min, thread_max).
4. `average`, `min`, `max` are stored as global variables. Thus, each thread can easily access the variable and change its value.
5. After three threads complete and exit, use `pthread_join` to release the resources used in terminated threads.
6. Finally, output all values required.

Q2.



```
[(base) leonardtsai@LeonarddeMacBook-Pro hw2 % gcc -o Q2.out Q2.c
[(base) leonardtsai@LeonarddeMacBook-Pro hw2 % ./Q2.out
Please input all integers separated by blank spaces.
7 12 19 3 18 4 2 6 15 8
2 3 4 6 7 8 12 15 18 19 %
```

1. First, input the starting number in terminal.
2. Use `strtok` to extract all integers in the input string. And store them in `parameters` which would be passed to threads.
3. Initialize `pthread_t`, `pthread_attr_t`. Use `pthread_create` to create three threads including sorting first half array, sorting second half array, merging the array with three thread functions (thread_left_sort, thread_right_sort, thread_merge).

A. thread_left_sort: using qsort() to sort first half part of the array, i.e., `int_arr[:data->right - data->left - 1]`.

B. thread_left_sort: using qsort() to sort second half part of the array, i.e., ` int_arr[data->right:]`.

C. thread_merge:

   Here, `sorted_arr` is the sorted list that we are constructing, `int_arr[:data->right - data->left - 1]` and ` int_arr[data->right:]` are the two sorted sublists that we are merging, and `left_size` and `right_size` are the sizes of the left and right sublists, respectively. The **while** loop in the beginning of the function compares the first element of each sublist, and adds the smaller one to the `sorted_arr`. This continues until we reach the end of one of the sublists. Then we simply add the remaining elements of the other sublist to the end of `sorted_arr`.

4. `int_arr`, `sorted_arr` are stored as global variables. Thus, each thread can easily access the variable and change its value.
5. After three threads complete and exit, use `pthread_join` to release the resources used in terminated threads.
6. Finally, output `sorted_arr`