Part 1. Open MP "Hello World" with output ordered by rank

Objective:

Start with Program 5.1 from the textbook as a starting point, and modify it to achieve the output ordered by rank.

Number of threads = 4
Hello from thread 0 of 4
Hello from thread 1 of 4
Hello from thread 2 of 4
Hello from thread 3 of 4

Explanation:

My strategy for achieving communication between the threads and ordering the outputs was with #pragma omp for. I first wanted to use nested loops of #pragma omp parallel, #pragma omp single and #pragma omp for. However, I kept getting the "work-sharing region may not be closely nested inside of work-sharing, critical, ordered, master or explicit task region" error. This error was caused because a different omp was nested in the single omp. The few times I did figure out a way to make the program compile with omp single the output was limited to the first thread.

```
ubuntu@codespaces_861a96:/workspaces/65145297$ gcc -g -Wall -fopenmp -o omp_hello hello.c
ubuntu@codespaces_861a96:/workspaces/65145297$ ./omp_hello 4
Number of threads = 4
Hello from thread 0 of 1
ubuntu@codespaces_861a96:/workspaces/65145297$ gcc -g -Wall -fopenmp -o omp_hello hello.c
```

So, after researching more ways to get the threads in sequential order, I found #pragma omp for ordered, which identifies a structured block of code that must be executed in sequential order. With the use of this and a for loop, I was able to print out the Hello() function in the order desired by the objective. As I was about to submit without #pragma omp single, I thought of using it at the end of the parallel zone to see if it would work. Surprisingly, it worked, and my output was:

```
ubuntu@codespaces_861a96:/workspaces/65145297$ ./omp_hello 4
Hello from thread 0 of 4
Hello from thread 1 of 4
Hello from thread 2 of 4
Hello from thread 3 of 4
Number of threads = 4
```

My format of the parallel zone was:

```
1.    #  pragma omp parallel num_threads(thread_count)
2.    #  pragma omp for ordered schedule(static, 1)
3.    for(int i = 0; i < omp_get_num_threads(); i++)
4.       #pragma omp ordered
5.          Hello();
6.       #pragma omp single
7.  printf("Number of threads = %d\n", thread_count);
```