Parallel and Distributed Computing (6E / 6F) Quiz 03 (Spring 2022). Instructor: Dr. Syed M. Irteza		Name:
Date: 2022-04-25		Roll Number:
Total Marks: 15 (5*2m + 5m)	Time Allowed: 10 mins	

1. If we use the `schedule(static, 2)` clause within the `#pragma omp parallel for`, we are enabling:
    a. ***Each thread is assigned 2 contiguous iterations of the for loop in round-robin manner***
    b. Each idle thread is dynamically assigned the 2 leftmost contiguous remaining iterations of the *for* loop
    c. Each thread is assigned half of the total iterations of the *for* loop in round-robin manner
    d. Each idle thread is dynamically assigned half of the remaining iterations of the *for* loop

2. If we use the `schedule(dynamic, 3)` clause within the `#pragma omp parallel for`, we are enabling:
    a. Each thread is assigned 3 contiguous iterations of the *for* loop in round-robin manner
    b. ***Each idle thread is dynamically assigned the 3 leftmost contiguous remaining iterations of the for loop***
    c. Each thread is assigned 1/3rd of the total iterations of the *for* loop in round-robin manner
    d. Each idle thread is dynamically assigned 1/3rd of the remaining iterations of the *for* loop

3. With the `schedule(guided, 4)` clause within the `#pragma omp parallel for`, we are enabling:
    a. A scheduling mechanism where the fixed chunk size is 4 iterations
    b. A scheduling mechanism where the maximum chunk size is 4 iterations
    c. A scheduling mechanism where chunk sizes vary, but will not exceed 4 iterations
    d. ***A scheduling mechanism where chunk sizes decrease, but don't go below 4 iterations***

4. The `OMP_DYNAMIC` environment variable when set to FALSE, indicates that OpenMP will:
    a. Create threads according to its dynamic adjustment algorithm
    b. Always use static scheduling
    c. ***Generate the same number of threads as requested by the `num_threads()` clause***
    d. Always use guided scheduling

5. When the `OMP_NESTED` environment variable is set to `TRUE`, this indicates that OpenMP will:
    a. ***Create a new team of threads with each layer of nested parallel pragma code***
    b. Not create a new team of threads with each layer of nested parallel pragma code,
    c. Consider each new nested *for* loop to be an OpenMP *for* loop construct

d. Not parallelize any for loops mentioned in the code

6. When parallelizing linked list traversal, why was the `#pragma omp single` clause used in one of the solutions? Can you explain the purpose of this clause in that context?[5m]

When written within a parallel region, the `simple` clause makes sure that the subsequent block or region will only be executed by a single thread (this may any one of the already created threads). This was necessary when we wanted to do linked list traversal using the original `while` loop along with the OpenMP `task` construct. Traversal of the list must be done by only one thread, but with the task construct, we are creating a unique `task` for applying the `complex_func()` on each node.