1. [1 pt] What is the main reason we moved from single core to multicore processors?

2. Suppose we have two MPI processes.
   a. [1 pt] Suggest one reason why they would execute slower on a system with two processors than on a system with one processor.
   b. [1 pt] Suggest another reason why the reverse could be true.

3. Suppose we have the following piece of code:
   ```c
   for (i = 0; i < 100; i++)
       do_work(i);
   ```
   a. [2 pts] What are the characteristics of do_work() that makes the above code suitable for MPI?
   b. [2 pts] What are the characteristics of do_work() that makes the above code suitable for OpenMP with dynamic scheduling?
4. [1 pt] Provide a scenario where you need to split the communicator in MPI (no need to write code)

5. [2 pts] We have seen loop unrolling in CUDA. But obviously, it can also be used in OpenMP. When do you think it will be beneficial in OpenMP?

6. [4 pts] For the following code:
   
   ```c
   for (i = 0; i < 100000; i++)
       a[i + 1000] = a[i] + 1;
   ```
   
   Can we, somehow, parallelize the above code using OpenMP? If so, please rewrite the parallel code. If no, explain why.

7. [2 pts] How could the following code sequence be changed to expose more parallelism but still achieve the same final result (i.e. at the end: x, a, b, and c have the same value as the sequential code)?
   
   ```c
   x++; 
   a = x + 2;
   b = a + 3;
   c++;
   ```
8. a. [1 pt] What is thread divergence?

b. [1 pt] Why is it bad for performance?

c. [2 pts] Based on your answers in a and b above, does the following kernel suffer from thread divergence? Justify.

```c
__global__ void do_work(int i){
    int result = 0;

    if( i < 5)
        for(j = 0; j < blockIdx.x; j++)
            result += j;

    a[threadIdx.x] = result;
}
```