# Concurrency & Parallelism
# Sample Test

José Duarte

June 22, 2020

1. **C** -
2. **C** - Partitioning is part of the process of parallelization of an algorithm. In this case, task partitioning.
3. **B** -
4. **C** - MapReduce operates on arbitrary kinds of elements, it is up to the programmer.
5. **D** - Also called the master/slave pattern, farm works over streams since the tasks are distributed by the master.
6. **C** - Map works over collections, the only statement which does the same is C.
7. **B** - In line 4 the statement combines two elements using `f`, thus we have a reduce pattern.
8. **B** - From the IBM documentation[1] we have: *The omp single directive identifies a section of code that must be run by a single available thread.*
9. **B** - From the moment the stack is popped local variables (not allocated on the heap) become invalid.
10. **A** - Monte Carlo methods, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.[2]
11. **D** - RAW, WAR and WAW affect the correctness of the program given the program is only correct if the dependency relationship is uphold.
12. **D** - If we run some iterations of the loop we see `a[1][0] = a[0][1]`, `a[1][2] = a[0][3]`, `a[2][0] = a[1][1]`, `a[2][2] = a[1][3]`. Thus there are no dependencies between loop iterations.
13. **D** - If we assume the whole program takes $T$ time to run we have:

$$0.5T + 0.5\frac{T}{100} = 0.505T$$
$$0.1T + 0.9\frac{T}{3} = 0.4T$$
$$0.4T + 0.6\frac{T}{50} = 0.412T$$
$$0.3T + 0.7\frac{T}{30} = 0.323T$$

And so we can conclude that having 70% of the code run 30 times faster is the better choice.

14. **A** -
15. **D** - The span is defined as the critical-path length, that is, the minimum of steps the algorithm must execute.
16. **D** - See Question 15.
17. **B** - A thread cannot acquire a lock if it is not free, thus the holder thread must first release it, synchronizing both events.
18. **D** - When the queue is empty $n = 0$ and thus the implication does not apply.
19. **D** - We cannot make guarantees about $T(op)$ based on $T_e(op)$.
20. **D** - The implementation does not ensure progress since the processes can be synchronized and do the following:
    (a) Put their flag up.
    (b) See the other flag as up.
    (c) Put their flag down.
    (d) Since their flag is not up this process repeats *ad eternum*.

    However the implementation provides mutual exclusion since both processes are unable to access the critical region at the same time.
21. **C** - The lock-freedom condition states that when the program threads are run sufficiently long, at least one makes progress.
22. **A** - Iterate the list until we arrive at the possible candidate.
23. **B** - We validate the previous and current nodes to check for deletions and "chain" correctness, that is `pred.next == curr`.
24. **A** - We see if the key exists, if it does we check if it is not marked for deletion.
25. **B** - The *LockSet* is initialized to the universal set.

[1] `https://tinyurl.com/y7qszwxh`
[2] `https://en.wikipedia.org/wiki/Monte_Carlo_method`

26. **D** - We first compute the maximal views of the threads:

$$M(T_1) = V_1$$
$$M(T_2) = V_2$$
$$M(T_3) = V_4$$

Comparing $M(T_1)$ with the other thread's views:

$$V_2 \subseteq M(T_1)$$
$$V_3 \subseteq M(T_1)$$

Comparing $M(T_2)$ with the other thread's views:

$$M(T_2) \subseteq V_1$$
$$V_3 \nsubseteq M(T_2) \wedge M(T_2) \nsubseteq V_3$$

We have an high-level data race.

27. **A** - When a new process enters a system, it must declare the maximum number of instances of each resource type that it may ever claim; clearly, that number may not exceed the total number of resources in the systems.[3]

28. **C** - See the labs.

[3] `https://en.wikipedia.org/wiki/Banker's_algorithm`