Worksheet 4.1: Threads, Multiple-Choice Questions

In the questions below, circle the right answer. There is only one correct answer

1. Which of the following is (are) true about threads?
   a. Threads within the same process must be run on the same CPU.
   b. Threads within the same process share global variables.
   c. Threads within the same process share the same stack.
   d. Parallel programming using threads can utilize more cores than parallel prog. using processes.
   Both b and c are correct. f. Both b and d are correct. g. Both a and c are correct

Explanation: Threads within the same process share the same address space, including global variables, but they have separate stacks. Parallel programming with threads can indeed utilize more cores effectively because threads are lighter and have less overhead compared to processes.

2. Which of the following is true about threads and processes?
   a. Context switching between threads is faster than context switching between processes.
   b. Threads within the same process share global variables, but different processes can't share global variables.
   c. A thread generally uses more resources than a process does.
   d. A multi-core system can be utilized using multiple threads but can't be utilized using multiple processes.
   e. Both a and b are correct. f. Both a and d are correct g. a, b and d are correct.

Explanation: Context switching between threads is faster because they share the same address space. Threads within the same process share global variables, but processes do not share global variables.

3. What is (are) the advantage(s) of dividing an application into multiple threads relative to dividing it into multiple processes?
   a. Utilizing a multi-core system b. One slow task won't slow the whole application
   c. Using less resources d. Easier communication using global variables
   e. Both a and c are correct f. Both c and d are correct g. Both b and d are correct

Explanation: Threads can take advantage of multi-core systems more efficiently and generally use fewer resources compared to processes due to lower overhead.

4. What are the limitations of Amdahl's Law?
   a. It assumes that the parallelizable code can be divided *equally* among the CPUs.
   b. You cannot apply it to a system with more than 10 CPUs.
   c. It does not account for the communication or synchronization overhead.
   d. You can apply it only when all CPUs are on the same chip not on different chips.
   e. Both a and b are correct. f. Both a and c are correct. g. Both a and d are correct.

Explanation: Amdahl's Law assumes perfect parallelization and doesn't consider the overhead caused by communication and synchronization between parallel tasks.

5. In a given program, one fifth of the code is parallelizable. What's the maximum speedup factor that can be achieved on a quad-core processor under ideal conditions?

a. 4 b. 20/17 c. 20 d. 2.5 e. 5 f. 5/4 g. ¼

Explanation: Amdahl's Law: Speedup = 1 / [(1 - P) + P/N], where P is the parallelizable portion, and N is the number of processors.

Here, P = 1/5, N = 4.

Speedup = 1 / [(1 - 1/5) + (1/5)/4] = 1 / [0.8 + 0.05] = 1 / 0.85 = 20/17.

6. Which of the following is (are) true about concurrency and parallelism?

a. With concurrency, only one process may be *running* at a given point in time, while with parallelism multiple processes may be *running* simultaneously.
b. With concurrency, the total time needed to execute a given set of long CPU bursts from different processes is always less than the time needed to execute the processes sequentially.
c. Parallelism is achieved using multithreading, while concurrency is achieved using multiprocessing.
d. Concurrency requires multiple CPUs, while parallelism may be achieved on a single CPU.
e. Both a and d are correct. f. Both a and c are correct. g. a, c and d are correct.

Explanation: Concurrency involves managing multiple tasks at the same time, but not necessarily simultaneously, while parallelism involves executing multiple tasks simultaneously. Concurrency can be achieved with multiprocessing, where different processes run at different times, and parallelism is typically achieved using multithreading.