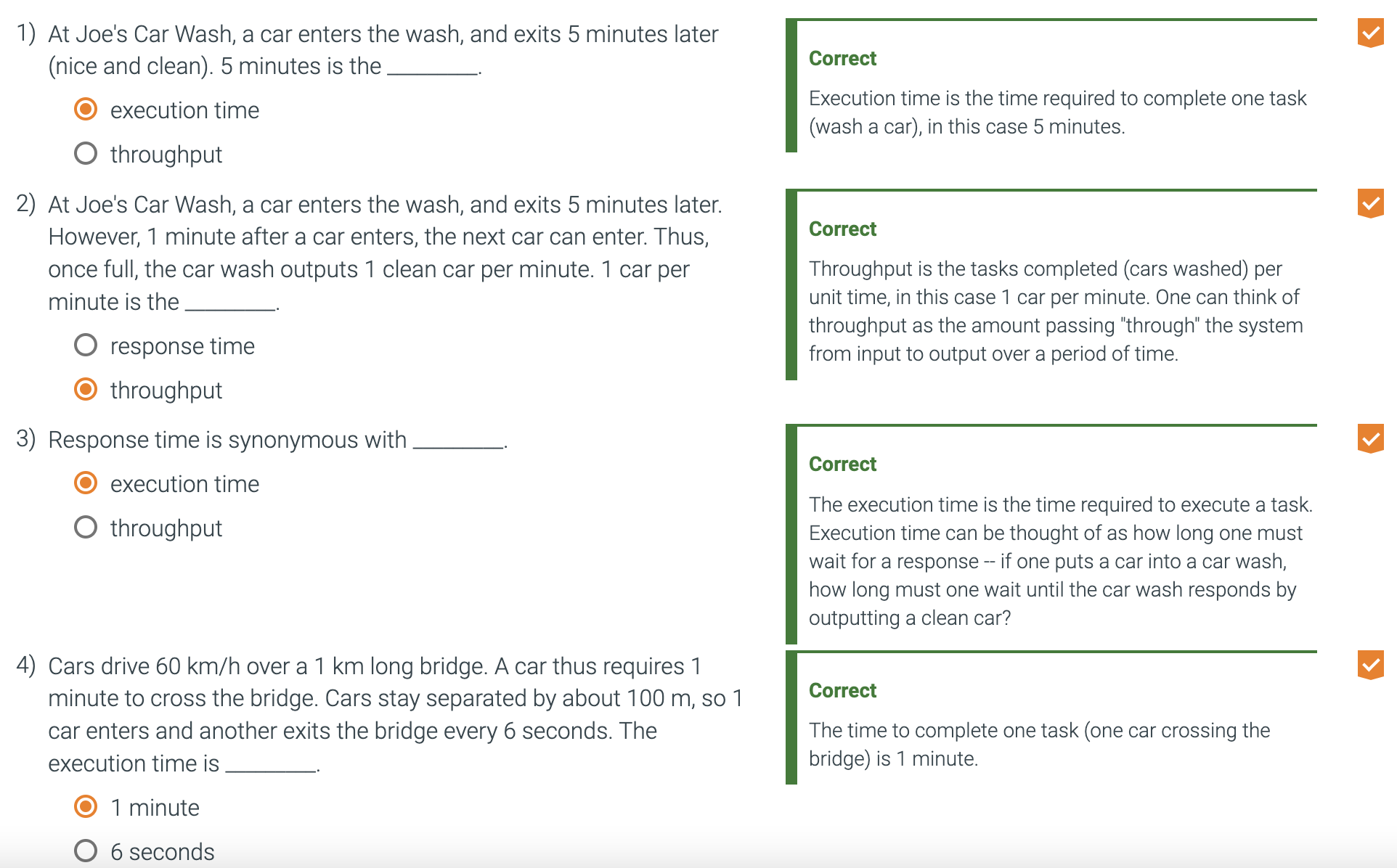
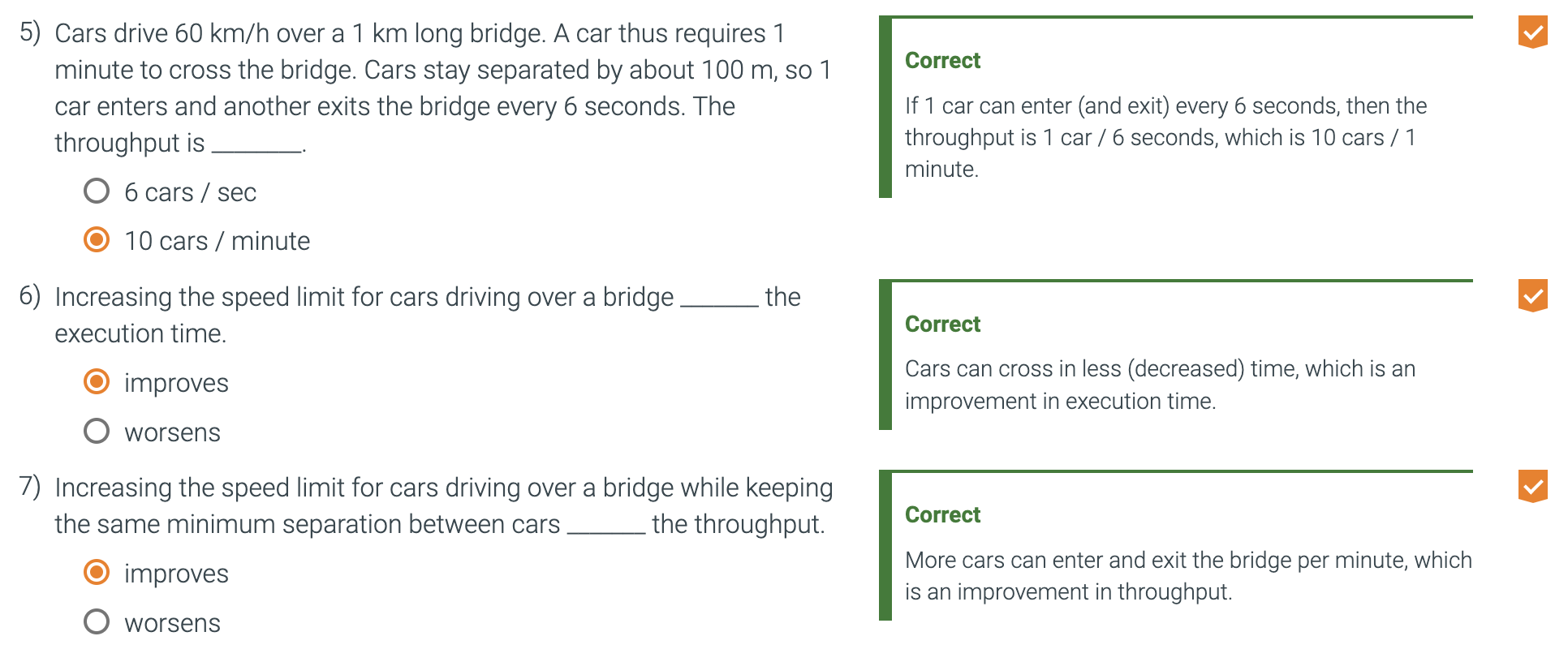
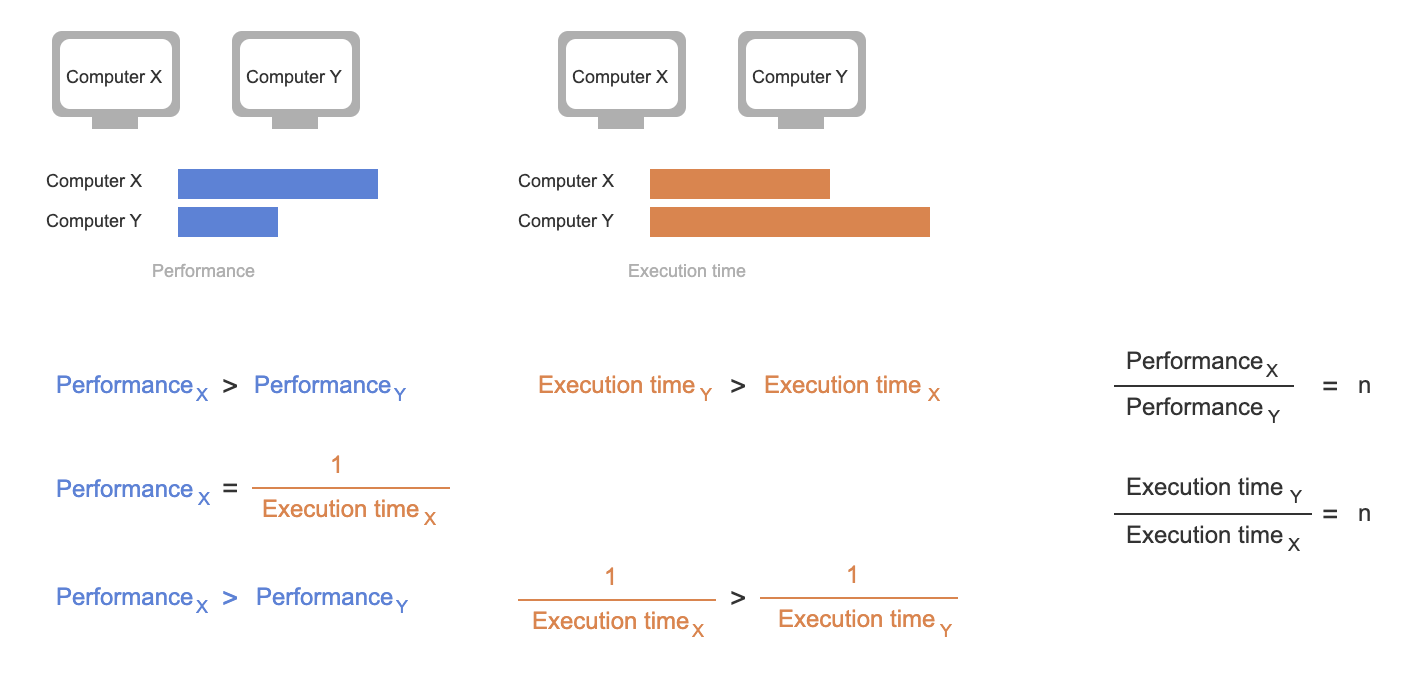
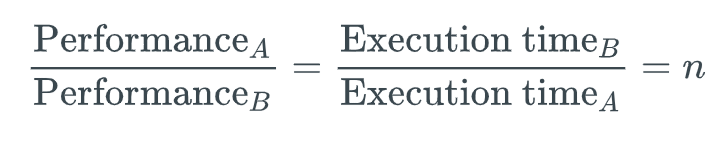
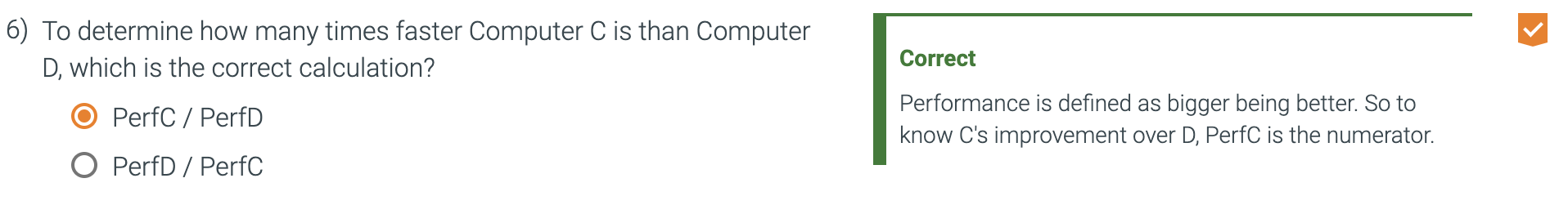
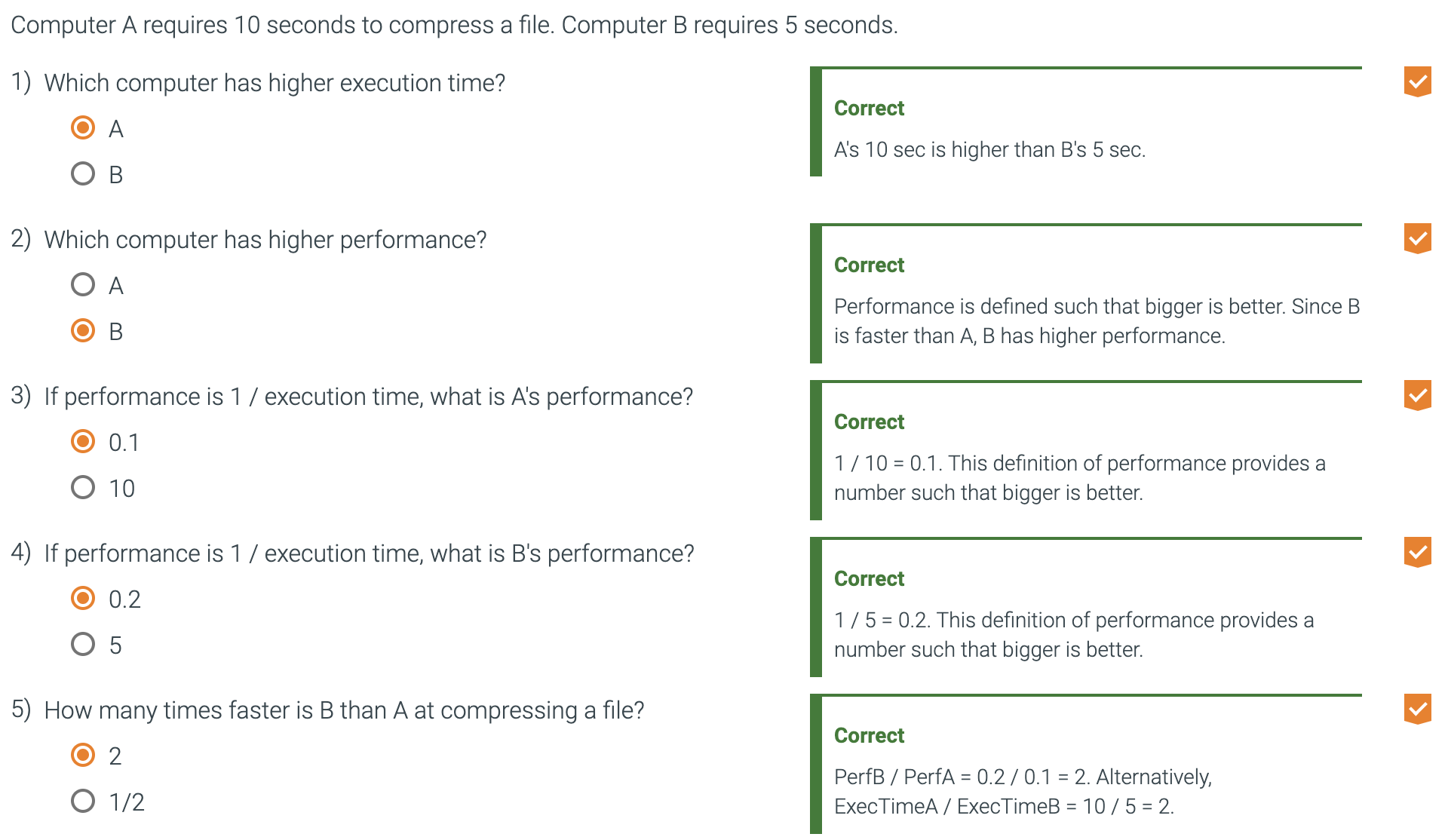
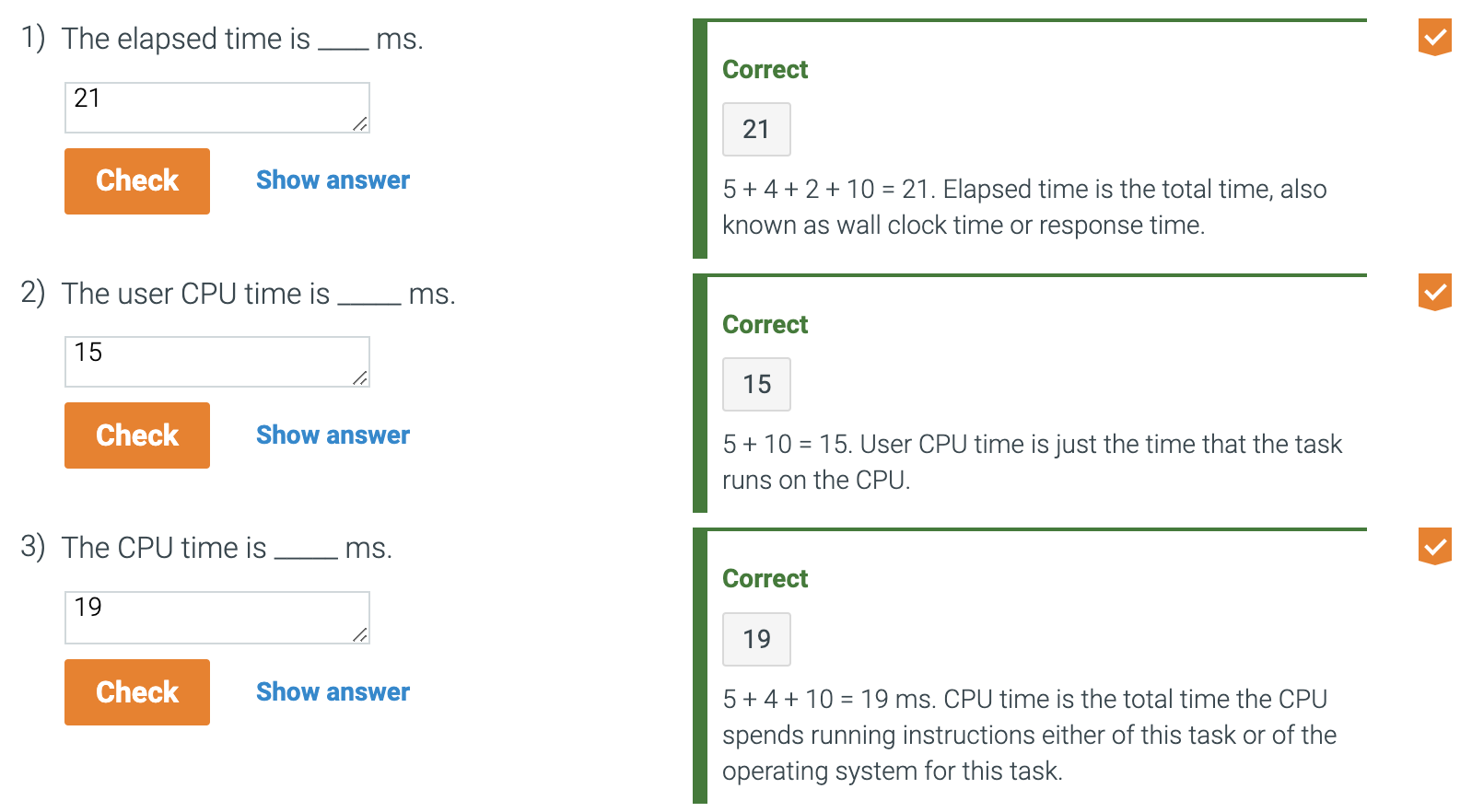
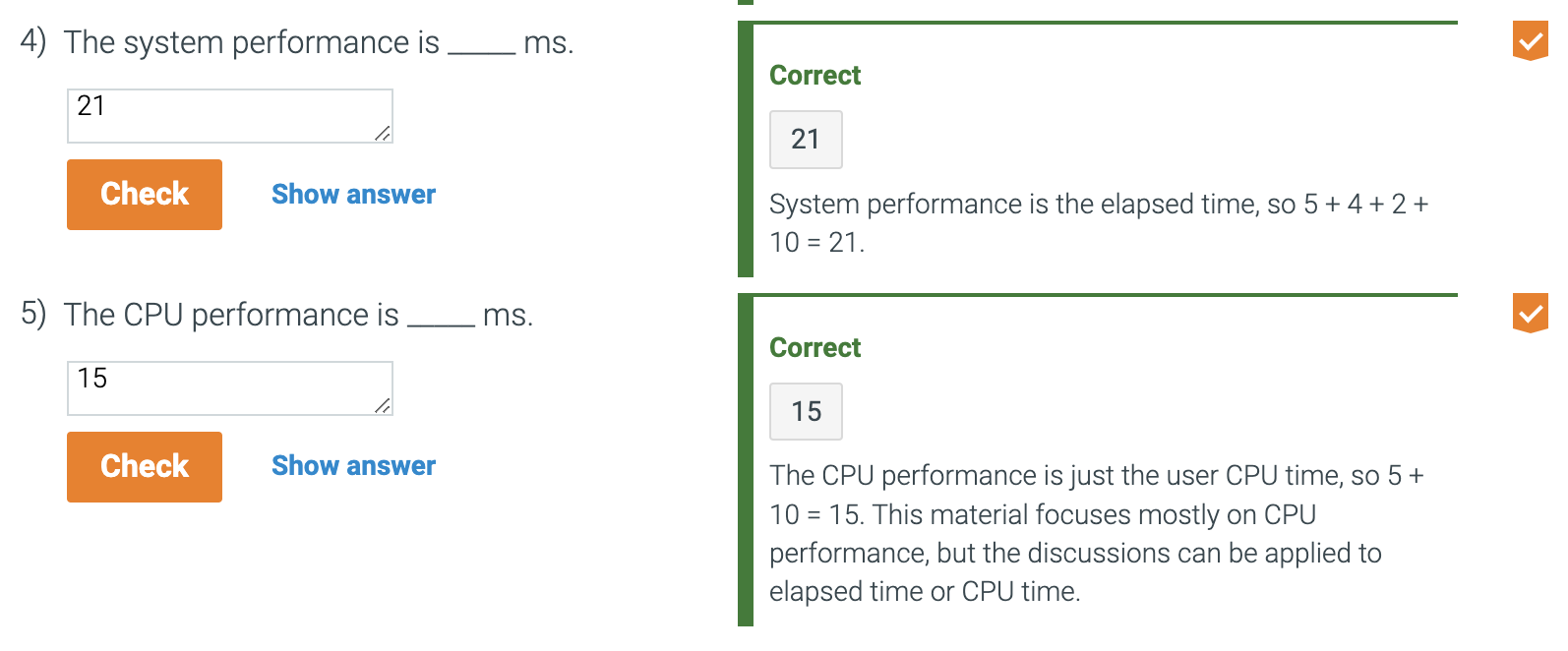
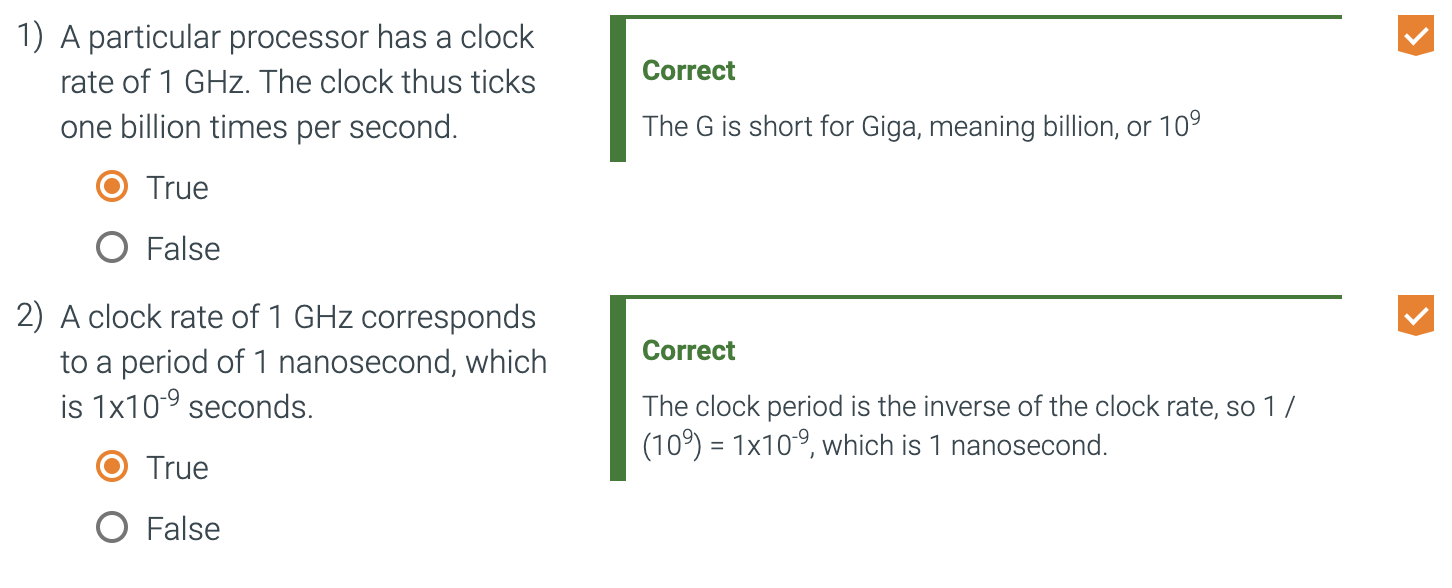
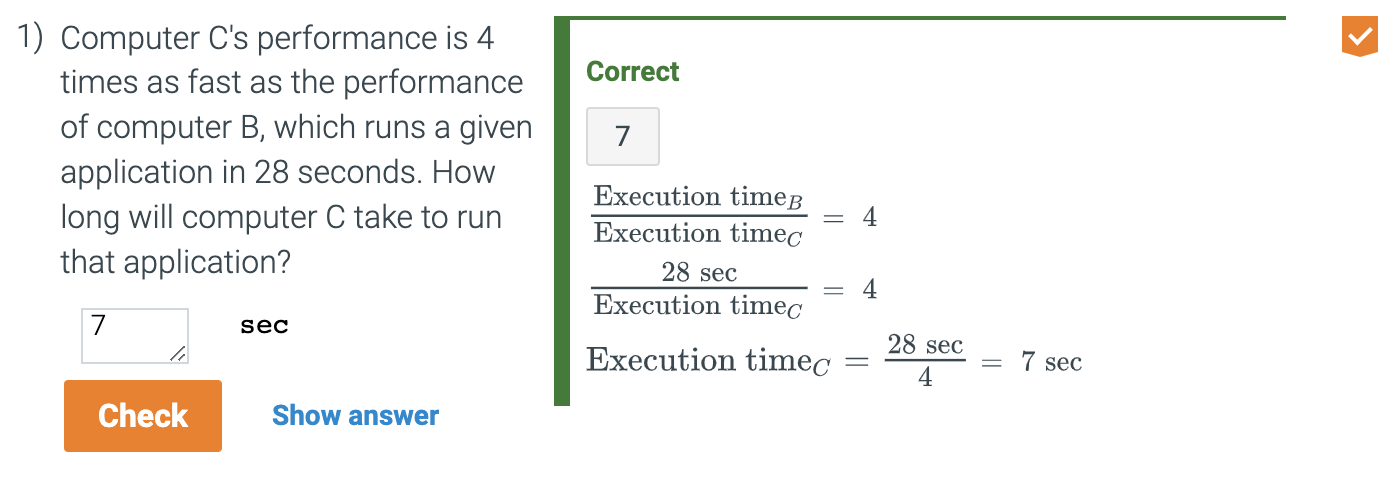
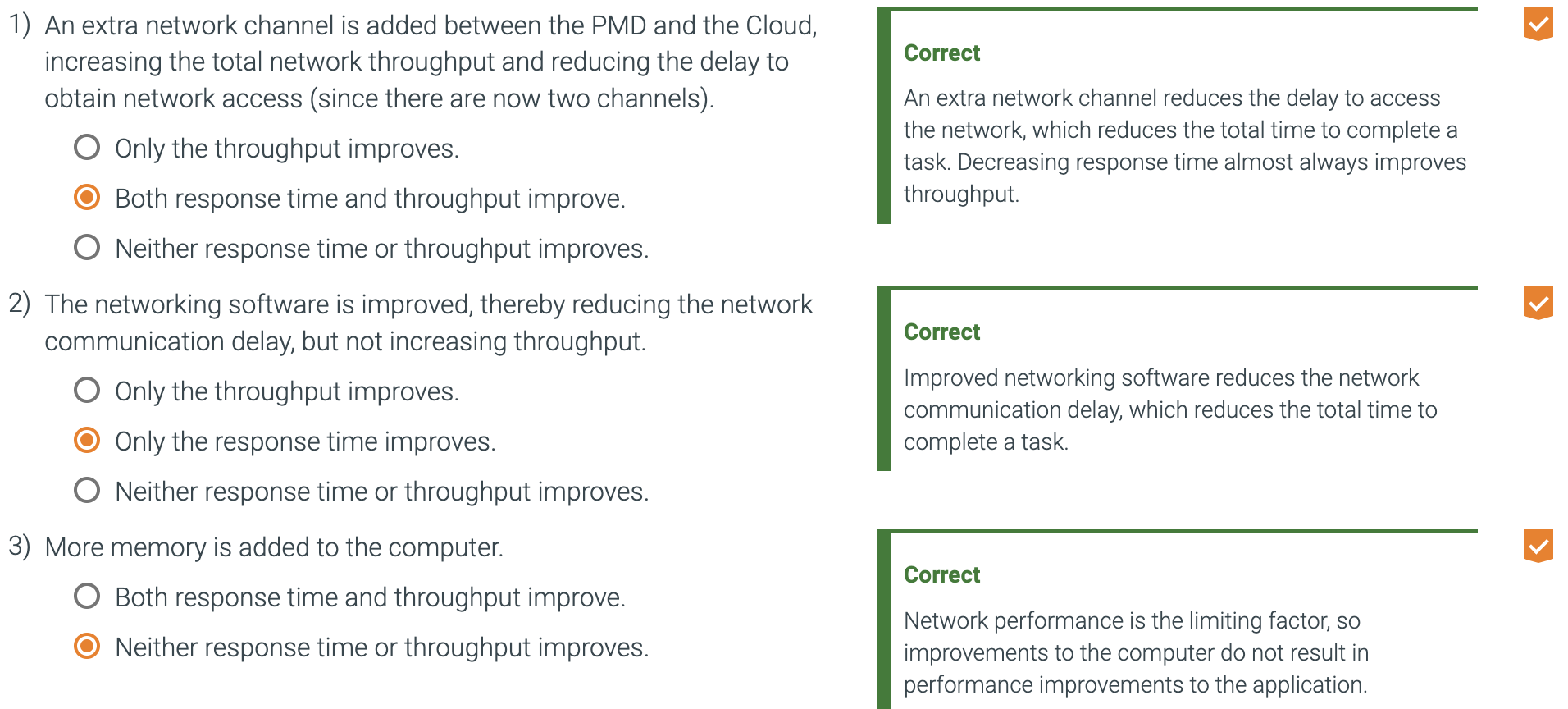
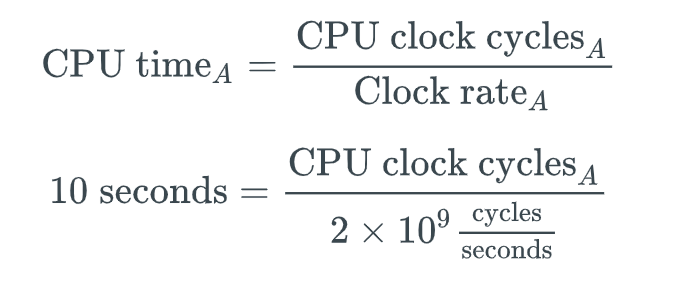
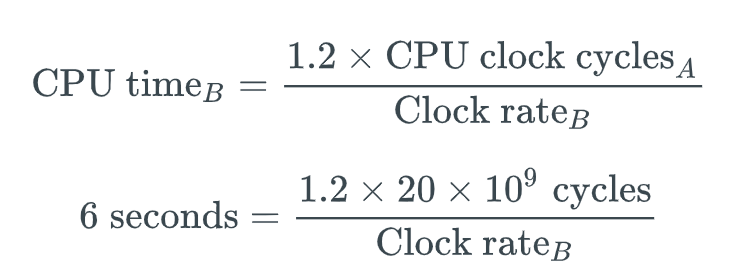
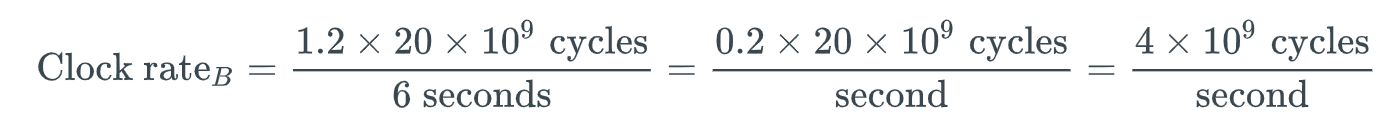
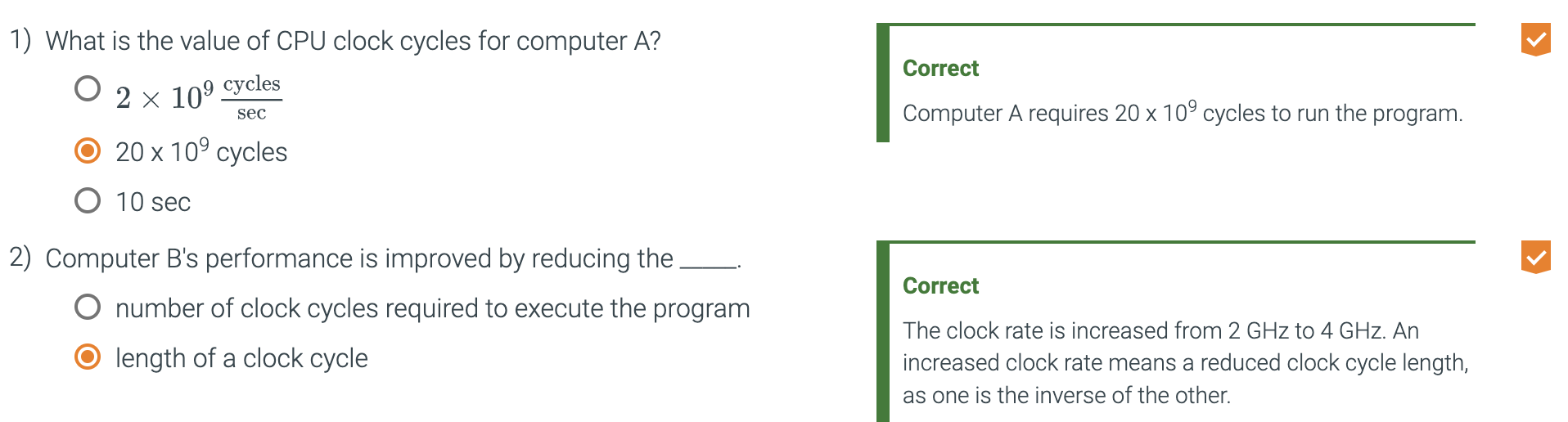
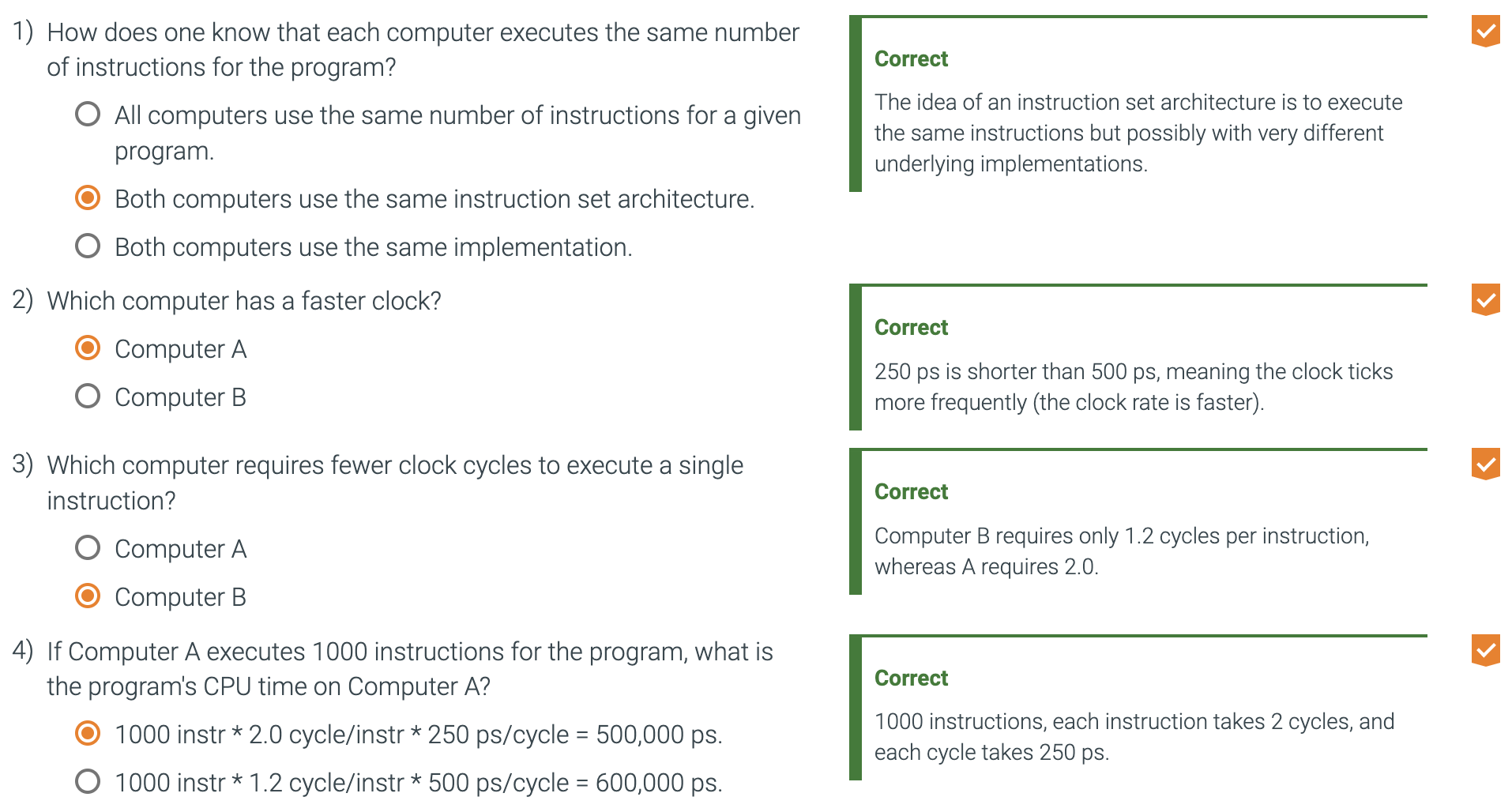
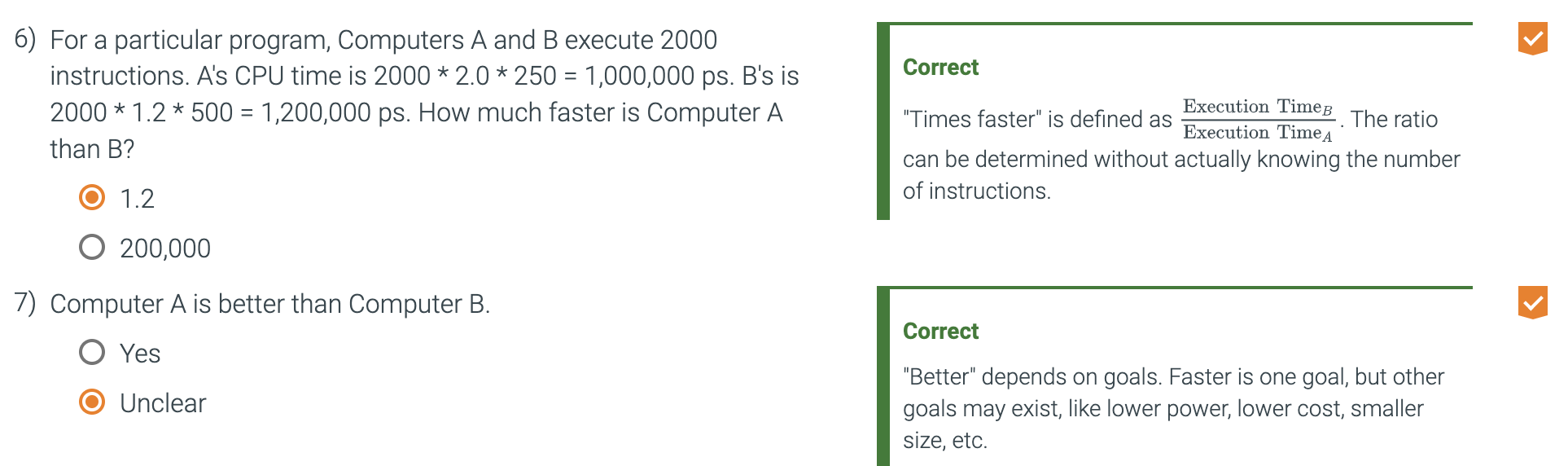
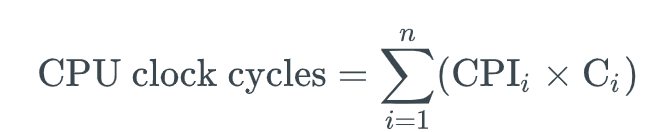
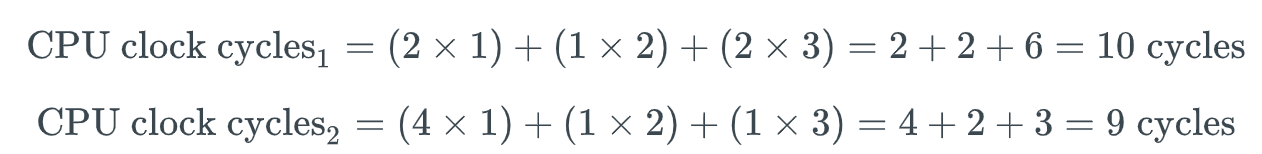
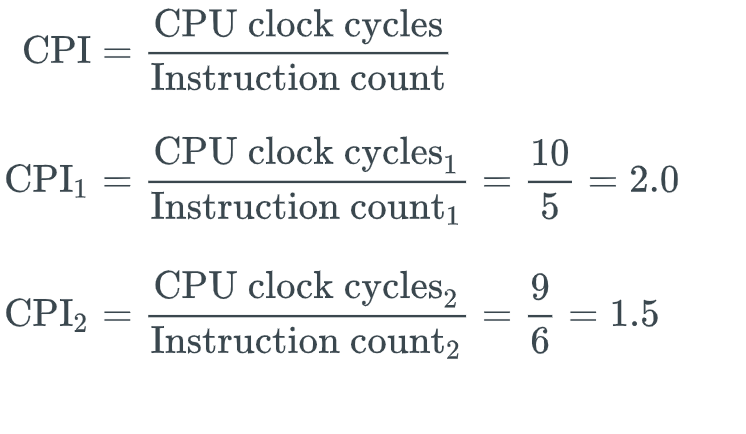
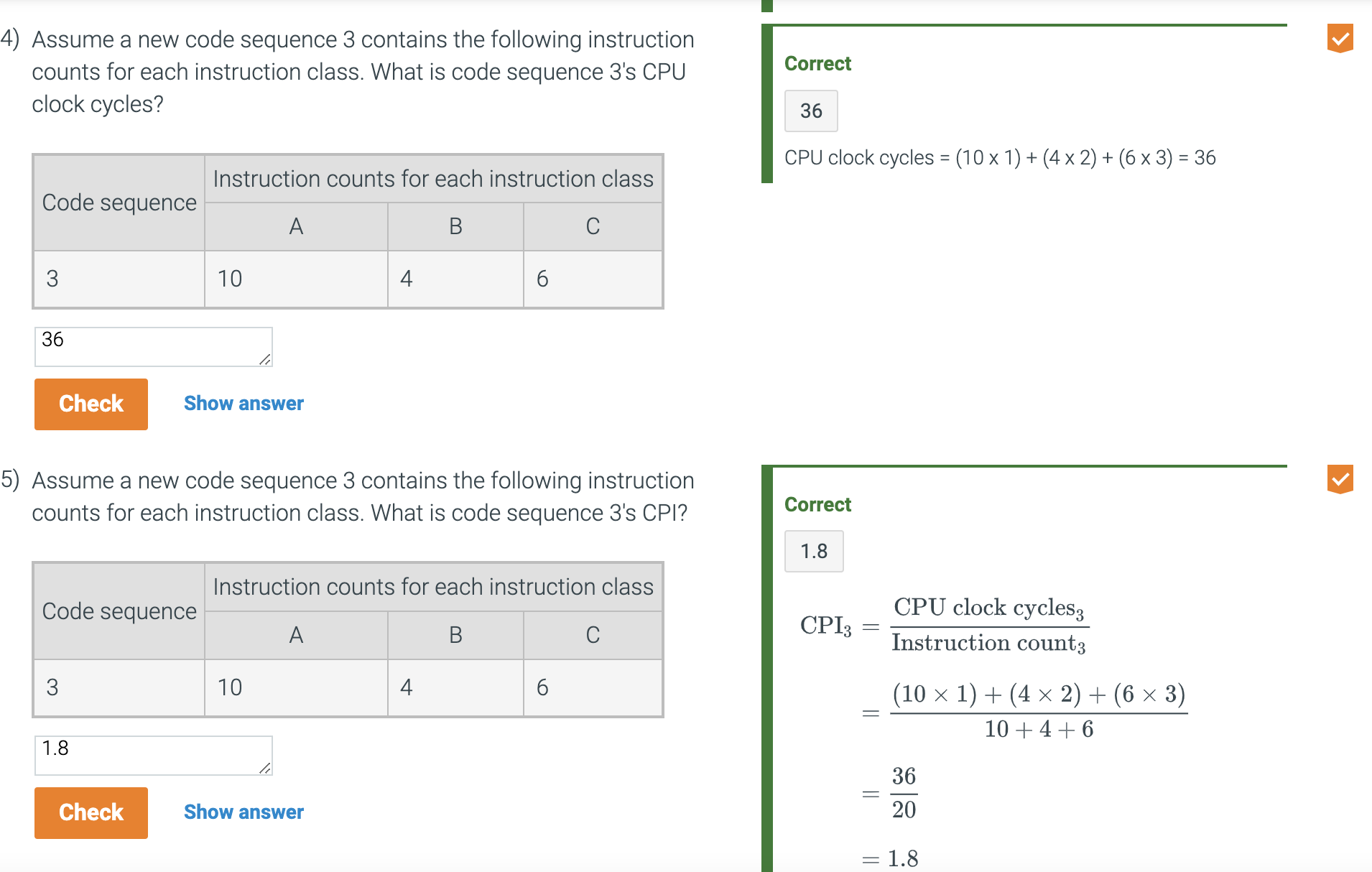
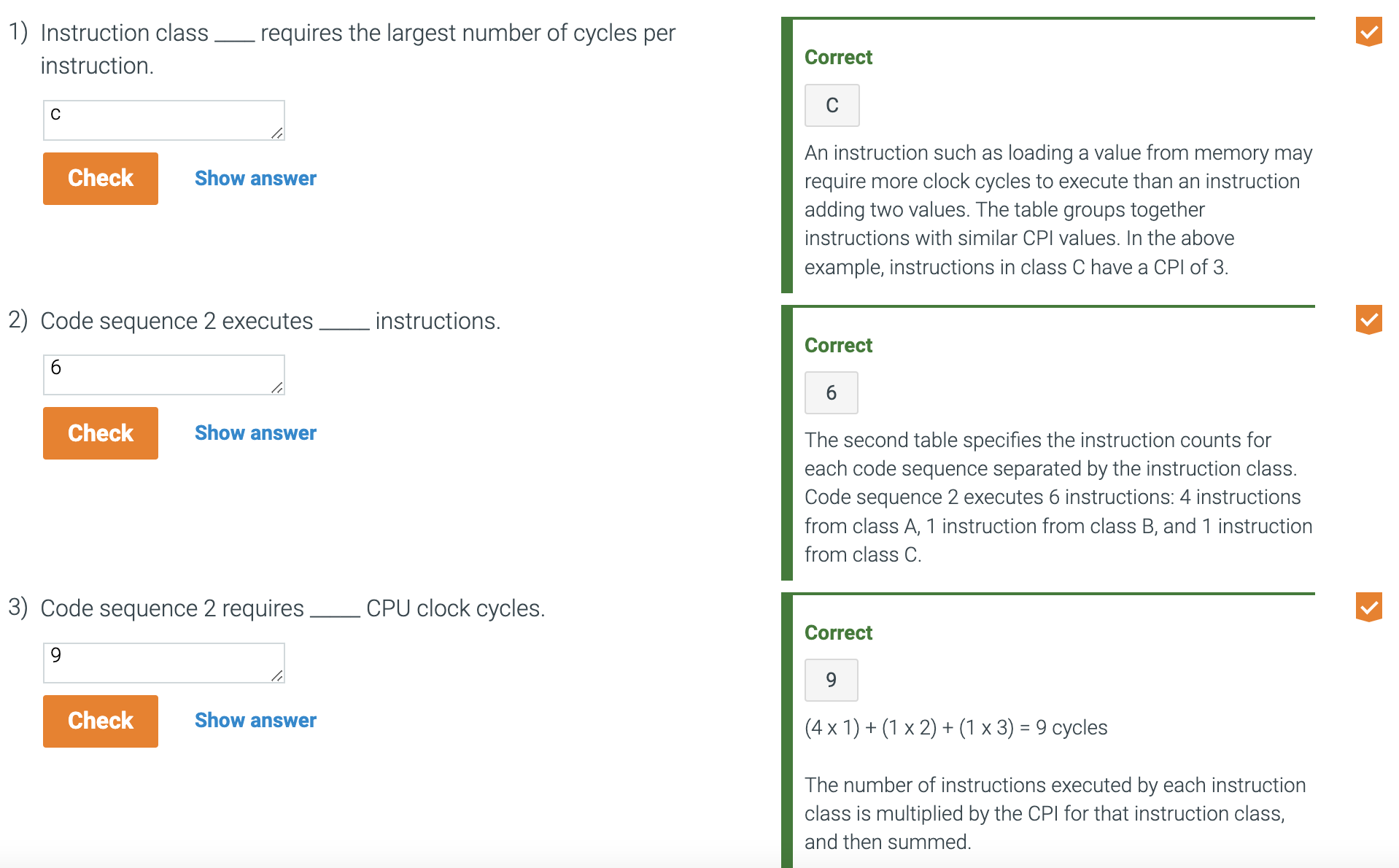
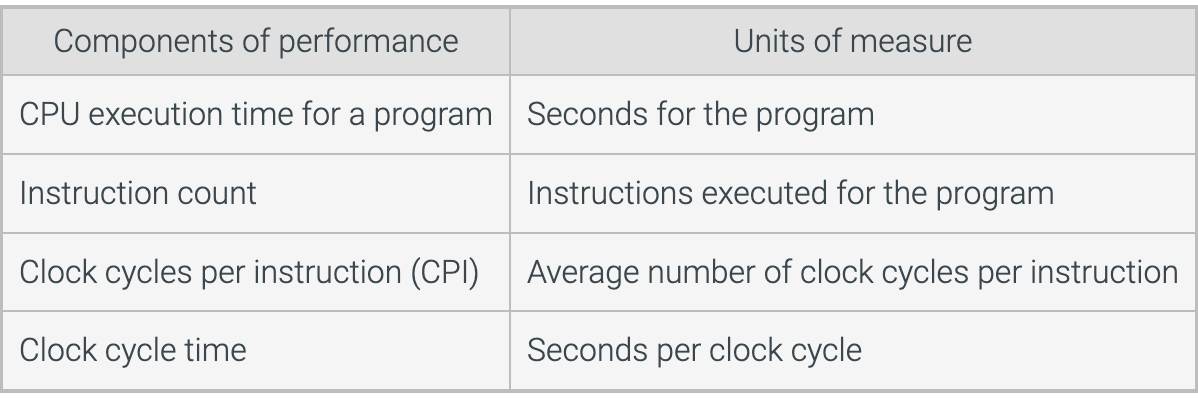
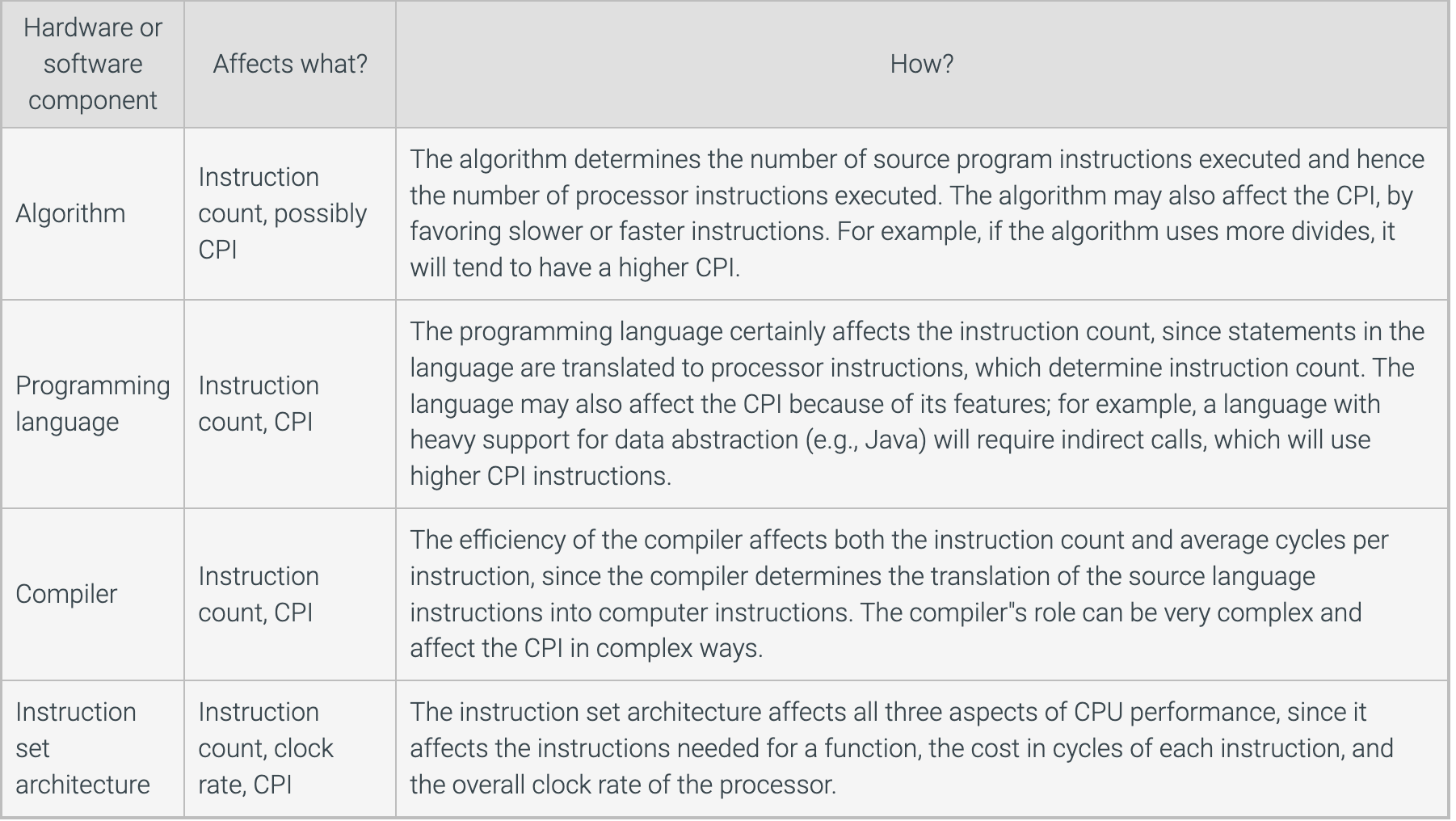
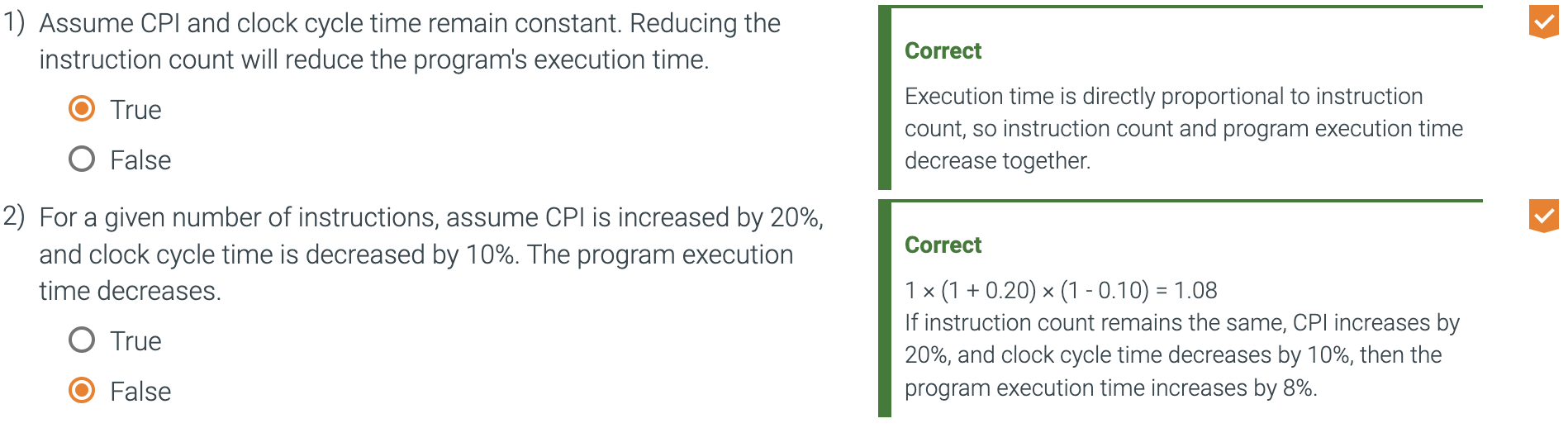
**Chapter 1.6 Notes**

* Introduction 1.6
  + T = execution time
  + IC = instruction count
  + CPI = cycles per instruction
  + CCT = clock cycle time
  + CR = clock rate (note that CR = 1 / CCT)
  + T = IC \* CPI \* CCT or, T = IC \* CPI / CR
  + Sanity check: just as in chemistry, physics, etc., do a dimensional analysis:
    - instructions \* cycles/instructions \* seconds/cycle => result will be in seconds
  + Speedup = old execution time / new execution time.
    - Note that the speedup is unitless.
    - Speedup = ( IC\_old \* CPI\_old \* CCT\_old ) / ( IC\_new \* CPI\_new \* CCT\_new )
    - See Lecture Notes 2 for examples
  + **Workload** = program and input
  + **Configuration** = CPU, cache, memory, I/O, OS, compiler, and optimizations
  + **Elapsed time** = program execution + I/O + wait, which is important to user
  + **Execution time** = user time + system time (but note that OS self measurement may be inaccurate)
  + **CPU performance** = user time on unloaded system, which is important to computer architect
  + **Response time (latency)** – time, between start and completion of a unit of work
  + **Throughput (bandwidth)** – rate, which is work units done per unit time, e.g., transactions/sec
    - Note that response time and throughput are reciprocals, i.e., throughput = 1 / response time
  + Marketing measures – the quest for single a marketing number usually leads to problems!
    - **MFLOPS** = millions floating point operations in program / (execution time \* 106)
    - **MIPS** = millions of instructions per second count / (execution time \* 106) = CR / (CPI \* 106)
      * Note that MIPS as a performance measure is not comparable across different computer designs when the instructions between the designs vary significantly in complexity.
    - **MIPS** = Microprocessor without Interlocked Pipeline Stages (DIFFERENT THAN THE ONE ABOVE!)
    - RISC
    - CISC
  + As an individual computer user, we are interested in reducing response time.
    - Response time
      * The time between the start and completion of a task - also referred to as execution time.
      * The total time required for the computer to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, CPU execution time, and so on.
  + Datacenter managers are often interested in increasing throughput or bandwidth.
    - Throughput
      * The total amount of work done in a given time. Also called bandwidth.
      * Another measure of performance, it is the number of tasks completed per unit time.
    - In most cases we need different performance metrics to benchmark personal mobile devices.
      * Mobile devices are more focused on response time
      * Servers are more focused on throughput.



* + For a given task, Computer X has less execution time (is faster) than Computer Y.
  + Desired is a measure of "performance" where bigger is better ("more performance" means faster).
  + One such performance measure is the inverse of execution time.
  + The performance of two computers can be related quantitatively. Computer X is n times as fast as computer Y.
  + The execution time of two computers can be related quantitatively. The execution time on computer Y is n times as long as computer X.
* Example:
  + If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?
    - We know that A is n times as fast as B if
    - Thus the performance ratio is 15/10 = 1.5
    - A is 1.5 times as fast as B.
    - We can also say that computer B is 1.5 times slower than computer A.
  + We usually say "improve performance" or "improve execution time" when we mean "increase performance" and "decrease execution time."
  + Performance = 1/ execution time
  + Time is the measure of computer performance:
    - The computer that performs the same amount of work in the least time is the fastest.
    - Program execution time is measured in sections per program.
    - The most straightforward definition of time is called wall clock time, response time, or elapsed time.
    - These all mean the total time to complete a task
  + We often want to distinguish between the elapsed time and the time over which the processor is working on our behalf.
  + CPU Execution time/CPU Time recognizes this.
    - CPU time is the time the CPU spends computing for this task and does not include time spent waiting for I/O or running other programs.
      * The response time experienced by the user will be the elapsed time of the program, not the CPU time.
    - CPU can be further divided into **User CPU Time** (time spend in the program) and **System CPU Time** (time spent in the operating system performing tasks on behalf of the program)
    - We use ‘system performance’ to refer to elapsed time on unloaded system’ and ‘cpu performance’ to refer to user cpu time.
* 1.6.5 Work
  + Elapsed time is all time combined
    - 5+4+2+10 = 21 ms
  + User CPU Time is only user time
    - 5+10 = 15 ms
    - This is because the task runs for 5 ms then 10 ms. this does not count time waited or paused.
  + CPU time is the total time the cpu spends running or operating the task. So when it is not idle
    - 5+4+10 = 19 ms
  + The system performance is the elapsed time
    - 21 ms
  + CPU Performance is the user cpu time
    - 15 ms
* Almost all computers are constructed using a clock that determines when events take place in the hardware.
  + These discrete time intervals are called clock cycles.
    - Or ticks, clock ticks, clock periods, clocks, cycles.
  + The length of a clock period is referred to as both the time for a complete clock cycle (250 picoseconds/250 ps) and the clock rate (4 gigahertz/4 GHz)
* Clock cycle
  + Time for one clock period, usually of the processor clock which runs at a constant rate.
* Clock period
  + The length of each clock cycle.
* CPU execution time for a program = CPU clock cycles for a program X Clock cycle time
* CPU execution time for a program = CPU clock cycles for a program / clock rate
* Example
  + Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?
    - First, find the number of clock cycles required for the program on A:
    - So CPU Clock Cycles A = 10 seconds X 2 X 10^9 cycles/second = 20 X 10^9 Cycles.
    - CPU Time B can be found using this:
    - To run a program in 6 seconds, B must have twice the clock rate of A.
* Execution time equals the number of instructions executed multiplied by the average time per instruction. 
  + The number of clock cycles required for a program can be written as
  + CPU Clock Cycles = Instructions for a program X Average clock cycles per instruction
* CPI (Clock cycles per instruction)
  + An average of all the instructions executed in the program.
  + CPI Provides one way of comparing two different implementations of the same instruction set architecture.
  + Average number of clock cycles per instruction for a program or program fragment.
* ps stands for picoseconds
* ns stands for nanoseconds
* Instruction count
  + The number of instructions executed by the program.
* CPU time
  + Instruction count X CPI X Clock Cycle Time
  + Or (Instruction count X CPI) / Clock Rate
    - Clock rate is the inverse of clock cycle time
* Next will be the basic components of performance and how each is measured.
* Instruction count depends on the architecture, not on the exact implementation.
* The CPI depends on a variety of design details in the computer as well as a mix of instruction types executed.
* Today's processors can vary their clock rates, so we would need to use the average clock rate for a program.
* 