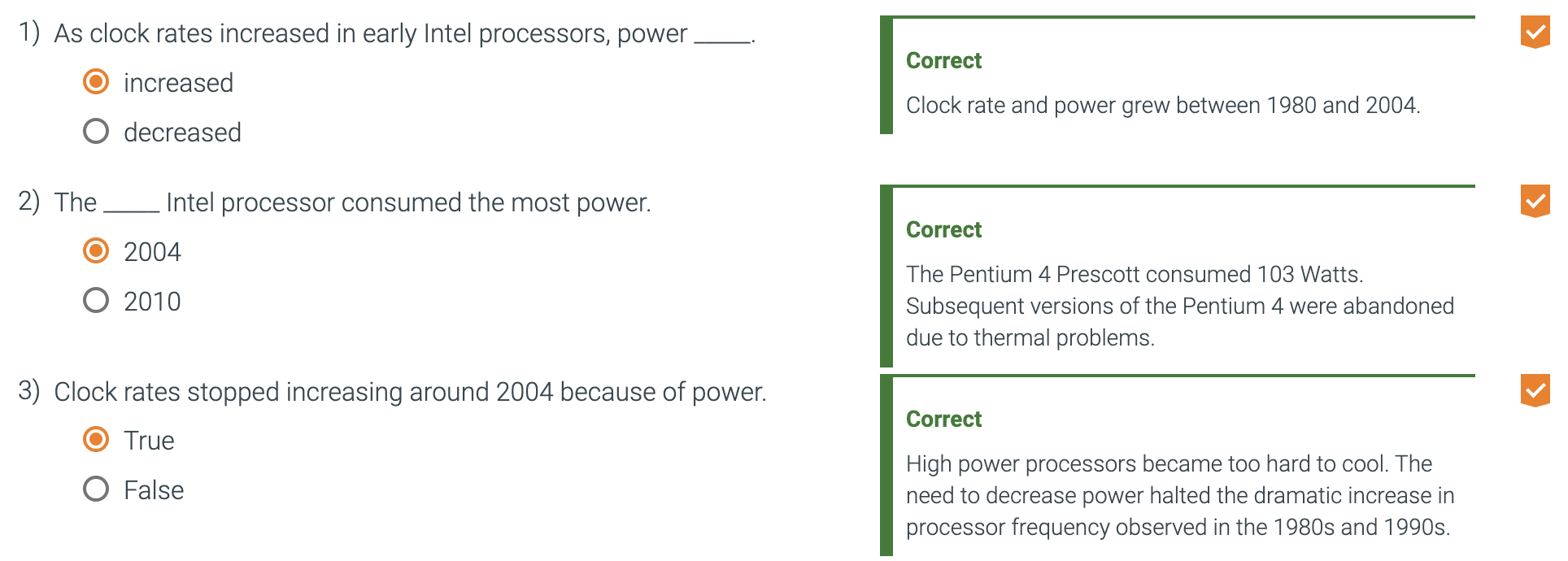
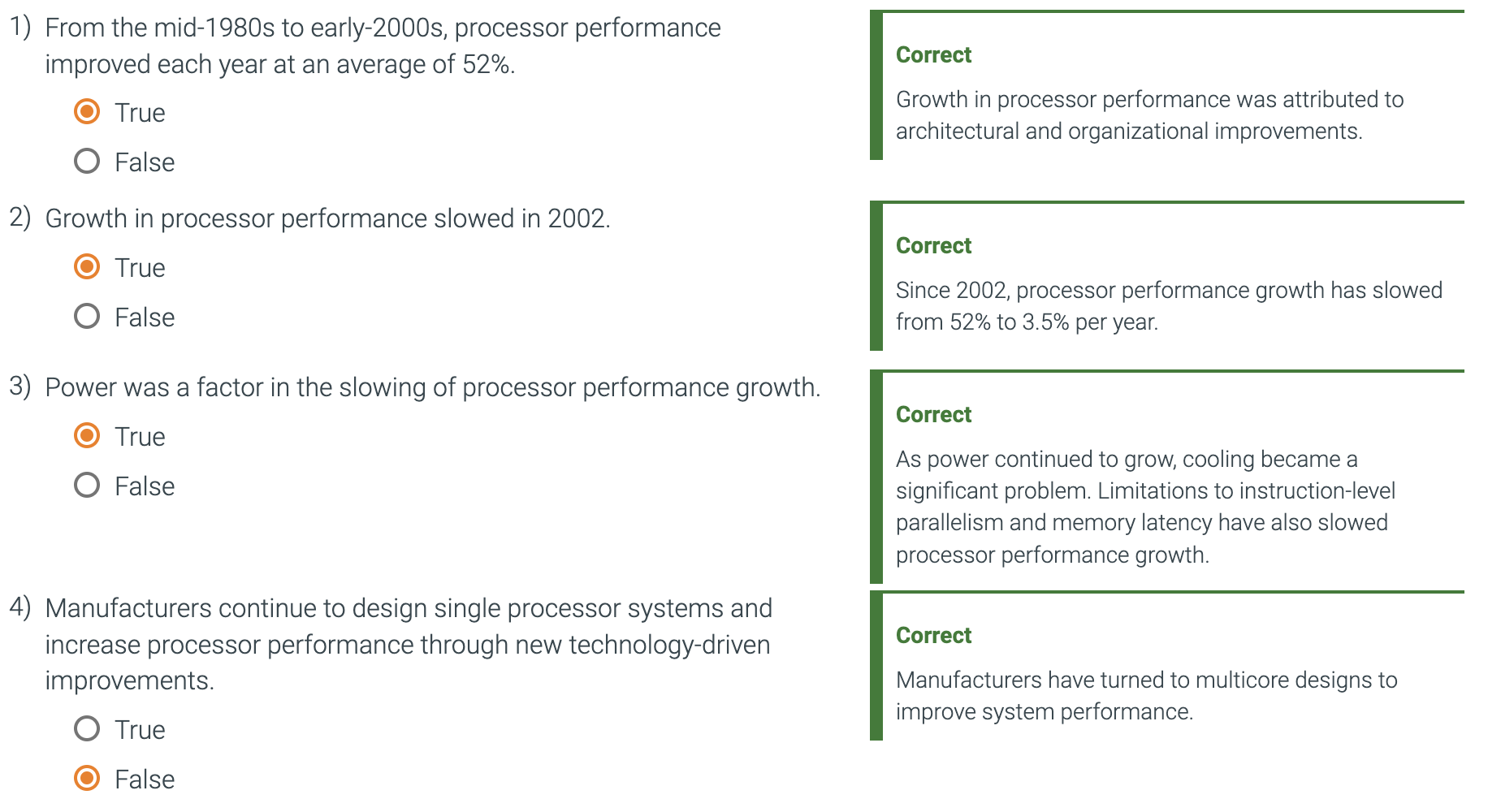
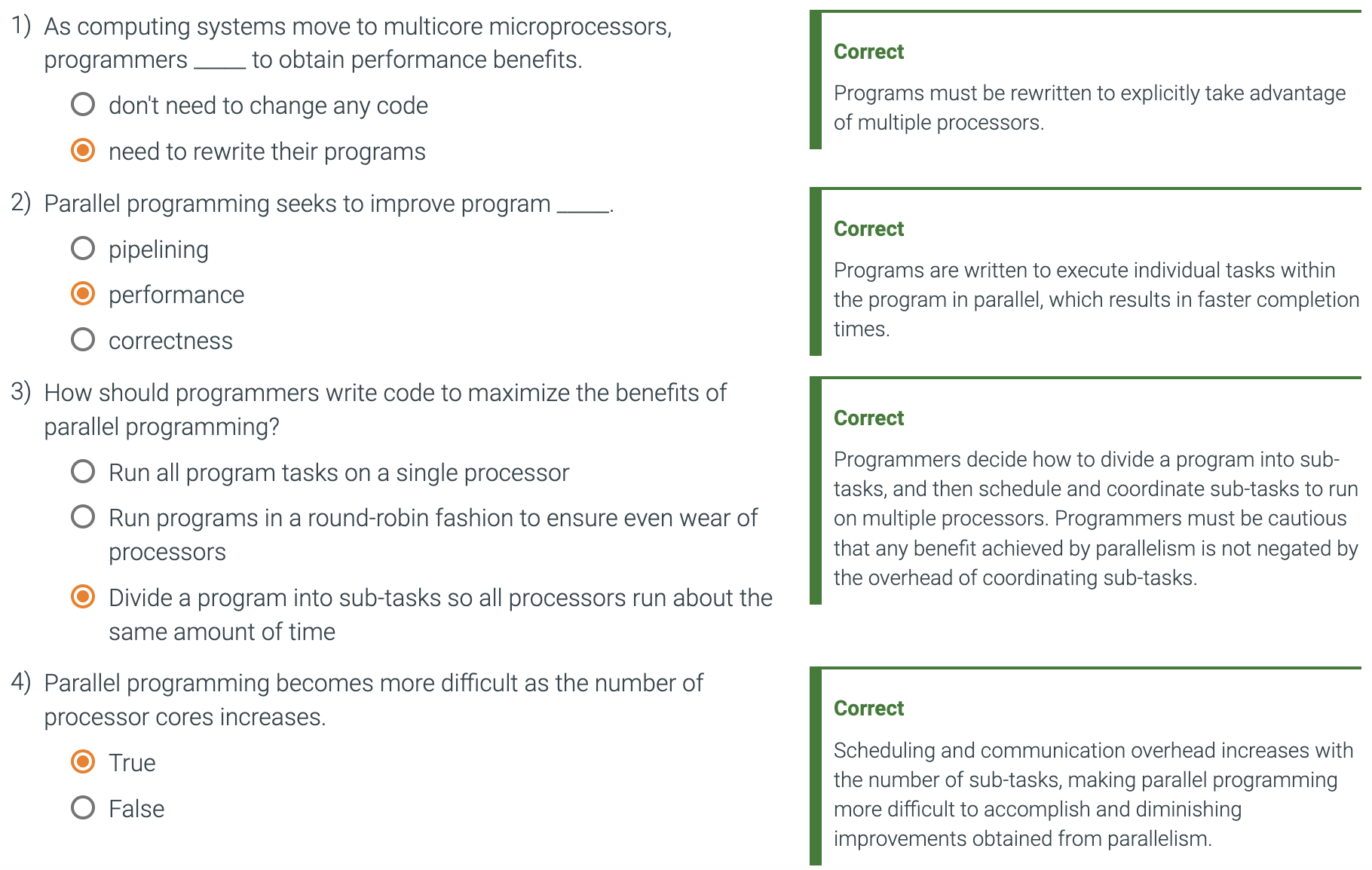
**Chapter 1.7 - 1.10 Notes**

* Lecture Notes for 1.7 through 1.10
  + The power density for single-core microprocessors grew too great and made them become much too hot/overheat.
    - This led to the wide use of multicore processors.
    - This is many processors on one die.
  + Programming choices for program parallelism
    - Explicit Program Parallelism
      * MPI
        + Message passing interface
        + This is Explicit Program Parallelism
        + It is very difficult to implement
    - Implicit program parallelism
      * Threading Building Blocks
        + Library routines designed to exploit multicore
        + Threads share the same memory space but independent units of scheduling, and are independent.
        + They can be on the same processors.
        + The best case is you would have multiple threads running at the same time on different cores.
        + The scheduling is often done by the library
      * OpenMP
        + Programmer uses compiled directives (pragmas) to identify parallelism
      * Intel C++ Compiler Parallelization
        + Automatically identify loops that contain parallelism
    - No change in programming and instead hope for enough independent users for good throughput.
  + Benchmarks
    - Performance is not necessarily correlated with the number of instructions executed, nor is it necessarily correlated with the clock rate.
      * Selected SPEC scores show an inverse relationship of performance to clock rate frequency when comparing across different processor families.
    - Kernels: key, small pieces of real programs
      * Linpack (1985)
        + Collection of fortran program segments used in solving systems of linear equations.
        + Linpack in the form of HPL is used as the benchmark for the top 500 supercomputers
      * Livermore Loops (1986)
        + Collection of 24 innermost loops found in scientific applications run at LLNL
    - Synthetic Programs: created to match execution profiles
      * Whetstone (1976)
        + Designed to simulate arithmetic intensive scientific operations
      * Dhrystone (1984)
        + Designed to simulate systems programming applications
    - SPEC (Systems Performance Evaluation Cooperative) (1988)
      * Started to provide benchmarks for workstations
      * SPEC89 was a suite of ten real programs that ran for 5-10 min each
        + SPECmark was the geometric mean
      * SPEC92 separated into SPECint and SPECfp
      * SPEC95 introduced baseline measured to reduce compiler optimization flag dependencies
      * Then
        + SPEC CPU2000
        + SPEC CPU2006
        + SPEC CPU2017
    - TPC (Transaction Processing Council) (1988)
      * Started to provide benchmarks for transaction processing
        + TPC-C : on-line transaction processing (credit,debit)
        + TPC-E : on-line transaction processing (brokerage firm model)
        + TPC-H : decision support (ad hoc queries)
    - EEMBC (Embedded Microprocessor Benchmark Consortium) (1997)
      * Started to provide benchmarks for embedded processors
      * Developed coremark benchmark to replace dhrystone
      * processor benchmarks for various application areas
      * system benchmarks for various application areas
  + Behavioral Simulation
    - A simulator for an instruction set architecture (ISA) is written in a high-level language (HLL) or a hardware design language (HDL).
    - Has built in language operators such as +, - to mimic behavior of computer implementing ISA
    - Primary goal is instruction-accurate simulation, but also cycle-accurate simulation to estimate performance
  + Structural Simulation using HDL
    - Refines a processor design down to the level of circuit and gate elements, which are usually chosen from a component library
    - **A computer-aided design (CAD)** tool can easily synthesize manufacturing design files from such a description, and circuit timing can be done using this type of description
* The power wall 1.7
  + The increase in clock rate and power of microprocessors are positively correlated.
    - The reason for their recently slowing growth is that we have run into a practical power limit for cooling commodity microprocessors.
    - **CMOS (Complementary metal oxide semiconductor)**
      * The dominant technology for integrated circuits.
      * The primary source of energy consumption for CMOS is dynamic energy (energy that is consumed when transitions switch states from 0 to 1 and vice versa)
      * The power required per transition is the product of energy of a transition and the frequency of transitions.
        + Frequency switched is a function of the clock rate
        + The capacitive load per transistor is a function of both the number of transistors connected to an output (**the fanout**) and the technology which determines the capacity of both wires and transistors.
* The sea change: the switch from uniprocessors to multiprocessors 1.8
  + Since 2002, the improvement in response time of programs for desktop microprocessors has slowed from a factor of 1.5 per year to 1.03 per year
  + Since 2002, the limits of power, available instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 3.5% per year.
  + As of 2006, all desktop and server companies are shipping microprocessors with multiple processors per chip.
    - Companies refer to processors as ‘core’ and microprocessors as ‘multicore’.
    - A quadcore contains four processors, or four cores.
  + Slowing of performance
    - From the mid-1980s to 2000s, processor performance improved 52% annually. Performance growth was largely attributable to more advanced architectural and organizational ideas.
    - In 2005, the limits of power, available instruction-level parallelism, and long memory latency slowed uniprocessor performance. Multiprocessor systems began to appear.
    - Growth in each processor's performance has slowed to about 3.5% annually in 2018.
    - Future systems may contain numerous processors.
  + Pipelining
    - Runs programs faster by overlapping the execution of instructions.
    - One example of instruction level parallelism.
    - The parallel nature of the hardware is abstracted away so the programmer and compiler can think of the hardware as executing instructions sequentially.
  + Explicitly Parallel Programs
    - Performance programming.
    - Why is it hard to write?
      * Increased difficulty in programming.
        + Program has to be correct, solve an important problem, provide a useful interface to people or programs that invoke it, and be fast.
      * Fast for parallel hardware means that the programmer must divide an application so each processor has the same amount to do at the same time.
        + Also overhead of scheduling and coordination does not take away the potential performance.
  + Real Stuff: Benchmarking the Intel Core i7 1.9
    - Benchmarks
      * Programs specifically chosen to measure performance.
      * A program selected for use in comparing computer performance.
    - Workload
      * A set of programs run on a computer that is either the actuall collection of applications run by a user or constructed from real programs to approximate such a mix.
      * A typical workload specifies both the programs and the relative frequencies.
    - In order to make the common case fast, we need to know accurately which case is common.
    - SPEC (System performance evaluation cooperative)
      * In 1989, SPEC created a benchmark set focusing on processor performance. (SPEC89)
      * The latest is SPEC CPU2017
        + Consists of a set of 10 integer benchmarks (SPECspeed 2017 Floating Point) and 13 floating point benchmarks (SPECspeed 2017 Floating point)
        + Floating points include structured grid codes for finite element modeling, particle method codes for molecular dynamics and sparse linear algebra codes for fluid dynamics.
      * SPEC reports a single number to summarize all 10 integer benchmarks
        + Dividing execution time of a reference processor by the execution time of the measured computer; normalizes the execution time measurements.
        + This yields SPECratio

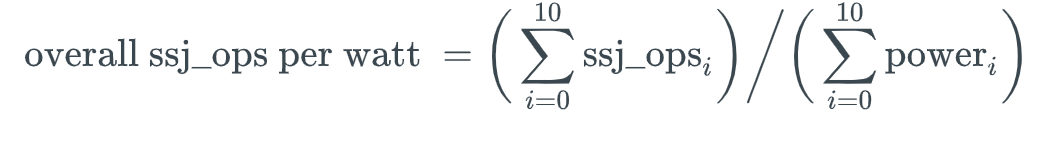
Is the inverse of execution time.

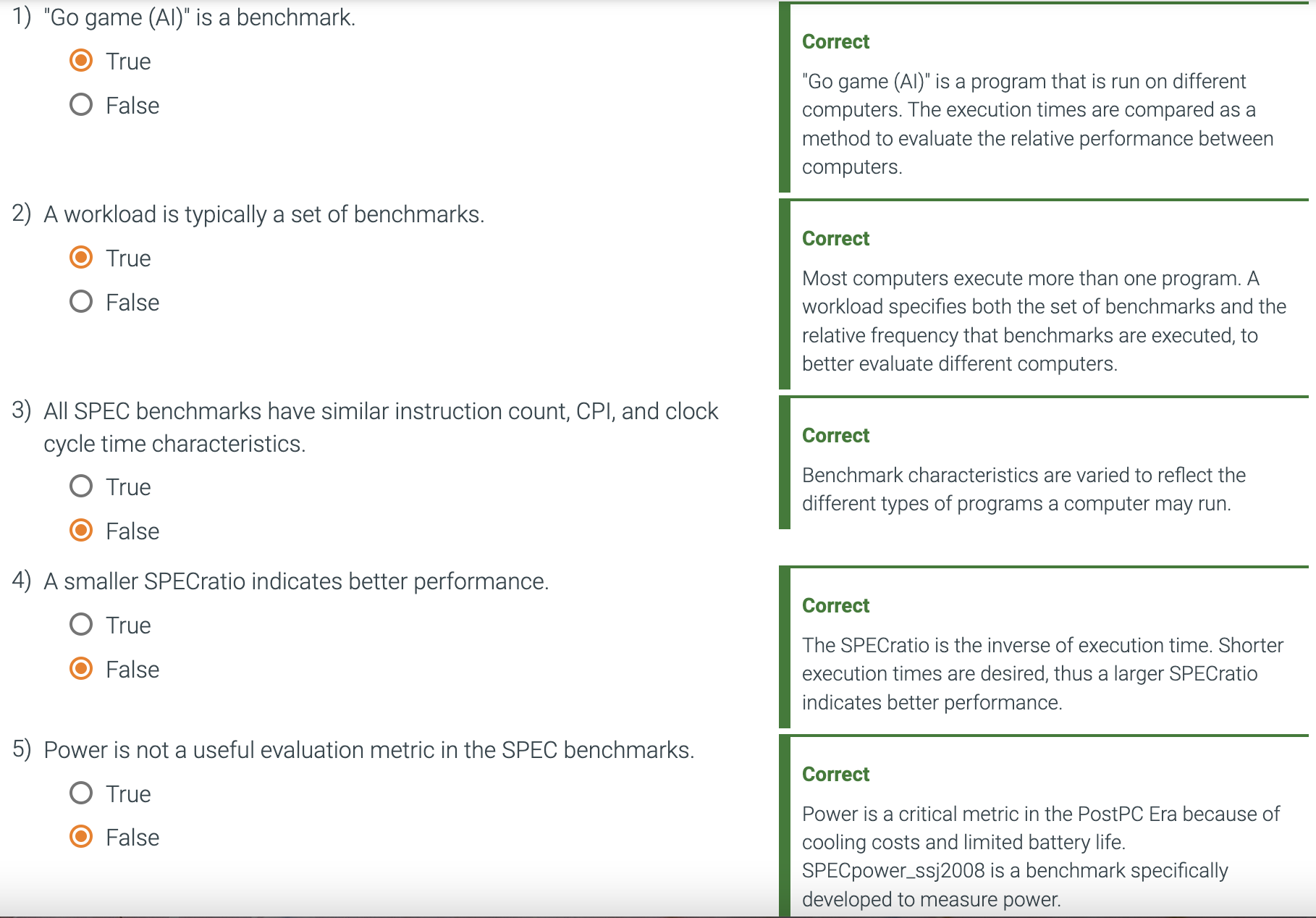
A SPECspeed 2017 summary measurement is obtained by taking the geometric mean of the SPECratios.

* + - * Formula for geometric mean is
        + Execution time ratio is the execution time, normalized to the reference computer, for the ith program of a total of n in the workload
      * SPEC power benchmark
        + Reports power consumption of servers at different workload levels, divided into 10% increments over a period of time.
        + Started with SPECJBB2005 (for java business applications)

Exercises the processors, caches, and main memory as well as the Java virtual machine, compiler, garbage collector, and pieces of the operating system

Performance is measured in throughput, units are business operations per second.

This is the overall ssj\_ops per watt

* + Going faster: matrix multiply in Python 1.10
    - To accelerate Python, programmers commonly call highly optimized libraries instead of writing the code in Python itself.
    - If we used the Numpy library instead, a 960x960 matrix multiply would take much less than 1 second instead of 5 minutes.