Part 1: -

1) If single processor takes 20 unit of time to complete a task, and then we run the same task on P processors. Then the ratio of time required to compute with single processor by time required to compute with P processors is parallel speedup. That is ratio of time taken to run on a single processor and time taken on multiple processors. In the above-mentioned example, if we run the task on multiple processors (5 for example) and the time it takes is 10 units. Then parallel speedup would be 20 / 10 = 2. In short, its Ts/T_p.
2) Parallel efficiency is used to measure how much parallel processors are we using in our processing. It is ratio of speedup factor and the number of resources (parallel processor). In our example above it is 1/5 * 2 = 0.4 (Efficiency = 1/p * parallel speedup). This means only 40 % of parallel processors are being used currently.
3) Amdahl's Law is equal to 1/((1-p) +p/s). It basically gives us the theoretical speed up of the execution of the whole task. Here s is the speedup of the part of the task that benefits from increased number of processors. It also states that instructions that are being run in sequence limit the max speedup. Hence, if we plot a graph of speedup against number of processors, then we won't get the ideal "linear graph" because of this limiting factor. According to Amdahl's law Maximum Speed up=1/ (1- (P/100)). Where P is percentage parallel tasks

Part 2: -

Amdahl's Law = 1/((1-p) +p/s)

P = 95%

Maximum Speed up=1/ (1- (P/100))

= 1/ (1- (95/100))

= 20

He will get speedup of 20 times then what he got before making 95% parallel.

Part 3: -

S = 1/((1-p) +p/s).

p = 0.5

s = 4

So,

1/ ((1-0.5) + 0.5/4))

= 1/ (0.5 + 0.125)

= 1.6

Part 4: -

Latency is the wall clock time; it impacts small payloads and messages and is usually measured in microseconds. By wall clock time it means the time it takes for data to be transferred from one point to another.

Bandwidth is the data transfer rate; it impacts large payloads and is measured in GB/s. In other words, it is the maximum amount of data being transferred in a fixed amount of time.

Throughput is the operation completion rate. That is how much information a system can process in a fixed time.

Part 5: -

There are 6 types of memories in the computer memory hierarchy. At the top of the hierarchy is the most expensive yet fastest memory. Data persists in it if power is on. It comes in the form of registers, which provide very fast, small yet expensive storage at CPU/GPU level. Then we get immediate term memory, or the caches. They are also very fast and expensive and provide small amount of storage. Then comes the affordable random-access memory. It provides, medium sized storage, and is fast but not as fast as registers or caches. It also retains data if it's powered on.

After ram comes the large capacity USB flash drives. They are cheap but slow and can store data even when power is off. Another power off storage that comes after flash drives is the hard drives. They provide very large storage but are slow and very cheap. All day to day used data such as OS is stored here. It is a midterm storage device. At the end of the hierarchy is tape backup. It is long term, slow, affordable, and provides very large storage.

Part 6: -

1) High performance computing with multi cores uses general purpose processor architecture. In this processor architecture their multiple processor cores in a single integrated circuit. Each core has its own caches to store data. This gives enhanced performance with memory and I/O channel support to speed up data transfer.
2) High performance computing with GPU uses special purpose processor called GPU. They have numerous cores and use special kind of memory. Here things closer to core are computed faster. They are designed to for special computing such as matrix multiplication. They have special tensor cores for such purposes. Parallel computation can be done on GPUs as they have thousands of cores. This type of computing is usually used in Artificial Intelligence and machine learning applications.

Part 7: -

Matrix multiplication is easier to parallelize than Monte Carlo method. This is because in matrix multiplication we can divide the matrix in n number of arrays and send them to n worker nodes. Each node can perform this calculation and together the multiplication will take much less time. However, in Monte Carlo method it is difficult to divide the work in subtasks, hence it is slightly difficult to achieve parallelization in it.

Part 8: -

SIMT means single instruction multiple threads, SIMD means single instruction multiple data and MIMD means multiple instruction multiple data. In SIMT a single instruction is processed by multiple threads whereas in SIMD single instruction processes multiple data. MIMD stands for multiple instructions multiple data. In MIMD multiple instructions manage multiple data at the same time. MIMD is much more efficient than SIMD in terms of speed and performance.

Part 9: -

The operating system that I am familiar with is the Windows. It has the "task manager". Which can be used to determine which resource is being used by which software. It gives us detailed data about how many processes are running, how many threads are running, And it also shows how much resource each process is consuming. The user can also terminate the process if they want from this task manager as well.

Part 10: -

Concurrency occurs due to context switching of processes on the CPU. Here the application processes more than one task at a time. And is used to reduce response time on a single core CPU, as they could normally run one process at a time. Concurrency gives an illusion of parallelism and speeds up the processing on a single CPU device.

In parallelism multiple tasks are run simultaneously. And multiple core CPUs are required for this purpose. Since it does a lot of things simultaneously, it speeds up the computational speed of a device.