(1) Describe how processes differ from programs by mentioning at least three differences between them.

**Location:** programs are stored in storage devices (e.g., hard drives) while processes are stored in the main (RAM) memory.

**Static vs. dynamic:** programs are static (they remain unchanged in storage devices), while processes are dynamic (the size of processes continuously changes in memory).

**Contents:** programs consist of (pretty much) only programs codes (machine codes or processor instructions), while processes consist of many more components, such as the heap area, stuck area, the program codes, and even more.

(2) What does “PCB” stand for? Why do operating systems need PCB?

“PCB” stands for “Process Control Block”

There are a couple of different reasons for operating systems to create a PCB for each process:

- Since processes are dynamic, operating systems need information about the current state of each process to manage them (how much CPU time and memory space have been allocated, and etc.)

- Since it is possible to create multiple processes from the same program, without a PCB (for each process), operating systems will not be able to recognize each process (“PID (process ID)” is the information that is a part of a PCB).

(3) What is “monolithic structure OS” (define the one)? Mention at least one primary advantage and disadvantage.

Monolithic structure operating systems are those that consist of a collection of procedures (or functional modules) that have no structure (in a flat structure)

**Disadvantage** (any one of the followings):
• Extremely time- and labor-consuming to apply bug fixes
• Poor flexibility (adding a new functional module requires re-compile the whole system)
• Require a large memory footprint (a large minimum memory requirement) since all the functions available in an operating system will be loaded to the memory, even though you do not need many of them.

Advantage:

• Operating systems in the monolithic structure usually have good (short) response time and (high) throughput because:
  - Most of the functional modules available are loaded to the memory when an operating system is loaded to the memory.
  - Functional modules are called just as “subroutines” without high calling overhead.

(4) Many operating systems use “external commands”. What are they? What is the primary reason to adopt them? What is the primary difference between “external commands” and “micro-kernel architecture”?

What: External commands are services (or functional modules) in operating systems, which are implemented as a binary executable file which is loaded to the memory and also kicked out of the memory on-demand basis.

Why: The binary executables for such external commands will not be loaded to the main memory when an operating system starts (boots), to avoid occupying a large memory space in a computer system.

Difference: external commands: they are still executed as a part of an operating system (in the KERNEL mode)

Micro-kernel architecture: they are executed in USER mode

(5) What are the advantages in using “micro-kernel architecture”? What is the primary disadvantage in “micro-kernel architecture”?

Advantages:

• smaller memory footprint (the required memory space for having an OS is small)
• Easy bug fixes (bug fixes can be performed simply by over-writing program files for the service modules that have bugs)
• Better robustness (since many service modules, which used to be executed as a part of an operating system, are executed in the user mode of a processor - just because they are now executed in the user mode, when they crash (because of bugs), such bugs will not crash the entire OS)

Disadvantage:

• Communication overhead slows down execution of system calls