1. Circle the relevant role of an operating system as R, I, or G (i.e., acting as a Referee, Illusionist, or Glue):

(a) R / I / G Physical details of a disk, such as sector size, are hidden.

(b) R / I / G Cut and paste commands work across different applications.

(c) R / I / G Prevent users from accessing each other’s files without permission.

2. Match these three terms to the appropriate blanks below: efficiency, response time, throughput

1. The design goal for batch operating systems was high throughput
2. The design goal for time-sharing operating systems was low response time
3. The design goal for virtual machines is high efficiency

Latency Comparison Numbers (~2012, Jeff Dean)

------------------------------------------------------------------------------------------------------

L1 cache reference 0.5 ns

Branch mispredict 5 ns

L2 cache reference 7 ns 14x L1 cache

Mutex lock/unlock 25 ns

Main memory reference 100 ns 20x L2 cache, 200x L1 cache

Compress 1K bytes with Zippy 3,000 ns 3 us

Send 1K bytes over 1 Gbps network 10,000 ns 10 us

Read 4K randomly from SSD 150,000 ns 150 us ~1GB/sec SSD

Read 1 MB sequentially from memory 250,000 ns 250 us

Round trip within same datacenter 500,000 ns 500 us

Read 1 MB sequentially from SSD 1,000,000 ns 1,000 us 1 ms ~1GB/sec SSD, 4X memory

Disk seek 10,000,000 ns 10,000 us 10 ms 20x datacenter roundtrip

Read 1 MB sequentially from disk 20,000,000 ns 20,000 us 20 ms 80x memory, 20X SSD

Send packet CA->Netherlands->CA 150,000,000 ns 150,000 us 150 ms

Notes:

1 ns = 10^-9 seconds

1 us = 10^-6 seconds = 1,000 ns

1 ms = 10^-3 seconds = 1,000 us = 1,000,000 ns

3. Operating systems can support communication between applications:

* Through the file system, using open(), read(), write(), close(), etc. system calls.
* With messages passed between the applications, using connect(), send(), receive(), disconnect() system calls.
* Through regions of memory shared between the applications, using load and store instructions.

Which method is likely the most efficient? Why?

* Shared memory, as it requires far less total methods

4. Consider this statement (from Wikipedia):

In early versions of DOS, printing was accomplished using the copy command: the file to be printed was “copied” to the file representing the print device. Control returned to the user when the print job completed.

Can an OS support printing in the background and thereby allow control to return to a user almost immediately after he or she issues a print command (or to an application after a system call to print is made) rather than waiting until the printing is complete? If so, suggest a way in which this can be done.

* Yes, queue the requested methods and return access back to the user, and the OS then can “share” the load between executing printing and anything new the user does