ATTENTION: The goal of these sample questions is to simply direct you when you review relevant topics for the exam. By no means, this set of questions is comprehensive (you still have to review all lectures!). **This also does NOT represent the actual format of the exam, style or number of questions, their difficulty, etc.**

Correct answers **are not** provided deliberately so that you can explore and look for the answer while you are preparing for the exam.

NOTE: I will mostly avoid theoretical questions about concurrent objects (e.g., histories, response/request notations, etc). But you still need to understand what sequential consistency and linearizability is. You also need to know what hardware actually implements. You need to know both blocking and non-blocking progress guarantees/conditions and be able to understand lock-free code (e.g., lock-free stack or queue). You need to know what TAS and CAS are.

Q1: Which of the following is true

A. Linearizability is easier to implement in hardware, while sequential consistency is harder
B. Sequential consistency is easier to implement in hardware, while linearizability is harder
C. Neither sequential consistency nor linearizability are typically implemented by hardware directly, but sequential consistency can typically be obtained by using memory barriers
D. Neither sequential consistency nor linearizability are typically implemented by hardware directly, but linearizability can typically be obtained by using memory fences
E. Neither sequential consistency nor linearizability are achievable in software or hardware

Q2: Which of the following are examples of the concept of “virtual memory”?
A. Paging through page tables
B. Segmentation
C. Disk emulation through qemu
D. /dev file system in Linux
E. Virtual file system

Q3: What is the monolithic OS? Please specify alternatives to this OS design.

Q4 [True or False]: UEFI has its own set of basic device drivers, which is independent from that of an operating system

Q5 [True or False]: UEFI and BIOS are both obsolete systems, neither are used by modern general purpose computer systems

Q6: Recall that TAS (Test-And-Set) is a special instruction which atomically sets a new value and returns and old value that was stored in a shared variable, e.g., can be considered as an **atomic** function

```
int TAS(int *variable, int new) {
   int old = *variable;
   *variable = new;
    return old;
}
```

Please show how you can implement a lock using this operation

Q7: Recall that TSS was historically used to manage tasks (processes, threads) to encapsulate task states when using ***hardware*** context switching. However, it is typically not used in the same manner anymore since ***software*** context switches are preferred. However, you still need to create one TSS segment per each processor (or core). What is the primary purpose of TSS (Task State Segment) when using it with modern x86-64 operating systems?

A. To store user-space stack pointer(s) since the default kernel stack pointers can be unreliable when an interrupt happens while CPU is in user mode
B. To store kernel-space stack pointer(s) since the default user stack pointers can be unreliable when an interrupt happens while CPU is in user mode
C. To store user-space stack pointer(s) since the default kernel stack pointers can be unreliable when an interrupt happens while CPU is in kernel mode
D. To store both user-space and kernel-space stack pointer(s) since all default pointers can be unreliable when an interrupt happens
E. Neither

Q8: What is the primary difference between NUMA (Non-Uniform Memory Access) and UMA (Uniform Memory Access)?

Q9: What is the primary difference between SMP (Symmetric Multiprocessing) and AMP (Asymmetric Multiprocessing)?

Q10 [True or False]: LibC is used to wrap system calls in a platform-independent manner

Q11 [True or False]: LibC does not provide anything beyond ordinary system call wrappers

Q12: Find bug(s) in the code which implements the ***pop*** operation for Treiber's stack. (top is a global variable which points to the top of the stack.) Note that the stack can be empty. Please ignore (i.e., do not consider) memory reclamation and/or the ABA problem. You also do not need to consider how values/objects are stored in the data structure.

```
1:      stack_node * pop () {
2:              stack_node *t;
3:              do {
4:                      t = atomic_load(&top);
5:              } while (!atomic_compare_exchange_strong(&top, &t->next, t));
6:              return t->next;
7:      }
```

Q13 [True or False]: Memory reclamation does not typically solve the ABA problem, so you still need to use ABA tags for queues and stacks to be prevent false mismatches

Q14: What is the difference between RAID 0 and RAID 1?

Q15: Give an example of at least three different components (including separate bits) of page table entries?

Q16: Can interrupts happen while the program is in user space?
Yes / No

Q17: Which of the following is true about page tables?
A. User program page tables reside in user space
B. Kernel portion is not shared across different page tables
C. Kernel portion is shared across different page tables
D. User portion is shared across different page tables (processes)
E. User program page tables reside in kernel space

Q18: Which of the following is true about a journaling file system
A. It is also known as a “virtual file system”
B. The last operation (create file, delete file, etc) can either fully succeed or not be performed at all during power loss
C. The last operation (create file, delete file, etc) can only fully succeed during power loss
D. The last operation (create file, delete file, etc) will never be performed during power loss
E. Neither

Q19: What is the difference between system calls and interrupts?

Q20: Please provide an example when a deadlock is possible (e.g., use more than one lock to demonstrate the condition)

Q21: Consider the following code that is executed in user space

```
void func(const char *str) {
        printf(str);
}
```

What does the 'str' variable contain?
A. Physical address to a read-only string
B. Virtual address to a read-only string
C. Character of type ‘const char’
D. Constant physical address to a string
E. Neither

Q22: What is difference between hard links and soft (symbolic) links

Q23: Please explain the difference between different thread models (1:1, M:N, N:1)

Q24: Suppose you need to set up a periodic APIC timer. Please explain how IDT (Interrupt Descriptor Table) is related to that and what you need to do with IDT.

Q25: Which of the following represent actual physical memory addresses when using hypervisors

A. Virtual addresses
B. Physical addresses
C. Machine addresses
D. Shadow page table
E. Nested page table

Q26: Please explain what is typically stored in file system i-nodes

Q27: Which of the following is true about virtualization with hypervisors

A. Popek and Goldberg were proven to be wrong about virtualization requirements
B. Binary translation is typically impossible, thus paravirtualization is the only option when using virtualization
C. Paravirtualization is preferable for x86-64 due to availability of protection rings (Ring 0, Ring 1, Ring 2, Ring 3)
D. Hardware virtualization was historically used but is now fully obsolete due to paravirtualization
E. Neither

Q28: [True or False] Containers provide better isolation than hypervisors

Q29: Please explain what file descriptors are, where they are stored. Give an example of three (default) file descriptors that are typically available for any program.

Q30: Remember that write() is used to write data to a file. How is it possible that LibC implements printf() through write() to output strings to the terminal.