1. In a system using modern virtual memory techniques, what is the relationship between a page and a page frame?

A page refers to a fixed length block of virtual memory, while a page frame refers to the physical page frame in physical memory that the page is mapped to.

2. Why are operating system ABIs of importance for convenient application software distribution?

ABIs don't require users to compile the source to run on their machines.

They are able to be run on hardware that supports the ABI.

Different ABIs can run on different hardware that supports the ABI, so having a set of ABIs can support multiple hardware, so convenient to distribute.

Different ABIs can support the same API, so same API can be run on different machines.

3. Why is information hiding a good property in an operating system interface?

OS interfaces can handle variations of, for example, files in a common way that the OS can handle as a single abstraction, (Common formats like a federation framework).

Allows virtualization of process memory, where OS hides details of paging, swapping, etc. from the process.

Allows users of the interface to focus on their job (in case of process, that is running instructions in memory) without needing to handle details.

4. When an operating system performs a context switch between processes, what information must the OS save?

OS saves the current Program counter (of previous process) and general registers of the previous process.

OS also saves the state of the process (blocked, stopped, ...) in the process identifier in process table.

5. What is the purpose of a trap table?

To provide a lookup from a trap (could be represented as a syscall number when calling syscalls) to run the trap handler in kernel mode.
The hardware, upon receiving trap, needs this table to look up the PC/PS of the trap handler to run in the OS and handle the trap.
Thus, it is used when trapping to OS in system calls, exceptions, or interrupts, therefore also used in scheduling.

6. What is a race condition?

A situation where modes of control such as multiple threads run and the outcome depends on the timing of their runs, where it matters which thread executes an instruction first. Outcomes of race conditions are unpredictable in threads- for example, because the time in between instructions varies between runs, and other threads instructions may interleave in different orders, thus causing different outcomes.

Race conditions can cause problems in shared data between threads because of order of getting, updating, and writing to data. They can be solved by combining instructions to be atomic or adding locks/mutexes.

7. Why is blocking a problem for user-mode threads? Why isn't it a problem for kernel-mode threads?

When a user-mode thread blocks, the OS is not knowledgeable that this is a user-mode thread and so the whole process will be blocked, thus blocking other user-mode threads within the same process, even though these user-mode threads could have been run. This prevents all other threads from running and changes the scheduling since the whole process blocks, even though only one thread intended to.
Kernel-mode threads will only block themselves and so other kernel-mode threads can keep running.

8. Why does Shortest Time-To-Completion First (STCF) scheduling require preemption?

STCF needs to run processes that will complete the earliest, first. If the scheduler receives long running processes and has to choose one to run, but then a short running process with much earlier completion time arrives after, it should preempt to run this newly arriving process to do STCF.

Without preemption, early completion jobs may have to wait for processes with later completion times. This isn't STCF, since a shortest-to-completion time process may have to wait and end up running much later, rather than first.

9. When a Unix-system follows a fork with an exec, what resources of the forked process are replaced?

The data / code segment, program stack, program counter, are replaced.

10. What form of fragmentation do we still suffer if we use a paging memory management system? For a segmented paging system, how much fragmentation per segment do we see? (For fixed size segments)

Still suffer slightly from internal fragmentation because pages are allocated to fit the memory request but this may be slightly larger than the request.

For majority of segments there isn't any fragmentation but if total space given by pages is slightly larger than requested memory, there may be some fragmentation less than the fixed size of each segment.