1. Nooks (10 pts)

Give two fault scenarios that Nooks’s method of implementing isolation and recovery could not handle well. Say briefly why.

Here are a few scenarios that Nooks couldn’t handle well; the first one is explicitly acknowledged in the paper. A brief reason is given.

- Nooks cannot handle if an extension maliciously tries to execute privileged instructions, because it lets extensions run in kernel mode. Consequently, an extension could simply reprogram the page tables to circumvent Nooks’ restrictions.
- Nooks cannot handle well reproducible software-related failures. For instance, if the filesystem driver crashes because the filesystem meta data was left in an inconsistent state after a previous crash, this crash will occur over and over.
- Nooks cannot handle well failures that are due to software bugs that are input-independent. If there’s a bug in your list implementation, it will still be there after Nooks kills and recover an extension that encountered the bug.
- Nooks cannot be applied to scenarios that involve essential system state. For instance, wrapping the scheduler as a Nooks extension would make little sense, since recovery after a failure would wipe out the queues maintained by the scheduler, which likely make the system unable to continue.
- Nooks by itself cannot detect when an extension has failed due to deadlock or livelock. (Note though that the paper points out that a recovery manager might use software mechanisms to decide when an extension is livelocked (or deadlocked) and could then terminate the extension.)

2. Journaling vs Soft Updates (10 pts)

Which of the following statements about journaling and soft update systems is true or false (2 pts each, no penalty for guessing)?

a) Soft updates provide for slower recovery after a crash because they need to replay their log.

    TRUE       FALSE

    FALSE. Soft updates do not keep a log, journaling fs do.
b) Soft updates track dependency information on a per-block basis.

   TRUE   FALSE

**FALSE. Dependencies are tracked on a per-pointer basis, because tracking them on a per-block basis would create cyclic dependencies.**

c) Soft updates always write as many or fewer blocks as the journaling version of FFS.

   TRUE   FALSE

**FALSE. When rolling back updates to avoid cyclic dependencies, soft updates may perform an extra write the journaling version would not incur. Even if you account for the write to the log that the journaling version must do, the unrolling of multiple cyclic dependencies within one block can trigger a series of additional writes, whereas the corresponding log entries (being smaller in size) can likely be coalesced into one write. (This question was more intricate than warranted for a true/false type of question, my apologies for that.)**

d) Journaling guarantees atomicity for certain operations, such as rename.

   TRUE   FALSE

**TRUE. Journaling keeps an explicit log of all operations.**

e) Journaling file systems do not typically synchronize meta-data operations with actual data write operations.

   TRUE   FALSE

**TRUE. Otherwise, the data writes would have to be included in the log, which would write most data twice.**

3. Mondriaan Memory (8 pts)

a) (4 pts) The Asbestos paper states that although the implementation of its event processes is optimized for processes that modify very little memory, the memory overhead of an event process is at least a single page of memory to hold the event process’s user-level state. Could Asbestos benefit from Mondriaan’s more fine-grained memory protection capabilities to reduce this memory overhead further? Justify your answer!

**No – Mondriaan provides protection domains within the same address space. Asbestos event processes require that each event process has their own address space (note that the ep_ programming model guarantees that a pointer value that was created in the base process is valid in the event processes also.)**
b) (4 pts) Would Nooks benefit from Mondriaan Memory Protection?

Absolutely. Nooks extensions run in the kernel address space within their own protection domain, exactly the model for which Mondriaan was designed.

4. Kernels (10 pts)

a) (4 pts) Why does Asbestos use a Mach-like microkernel design?

A major goal of Asbestos is to control information flow between processes to implement mandatory access control. In a microkernel-design, all access to services is provided through messages that are exchanged between a process and servers, which makes it easy for Asbestos to control simply by assigning labels to ports.

b) Extensible OS such as SPIN or the Exokernel have been criticized for the cost they must invest to solve “unnatural” problems that they themselves created. Name two examples of such problems.

The exokernel, by refusing to allow any management decisions to be made inside the kernel, must support mechanisms for safely interpreting the code that makes management decisions, such as UDFs. This lack of semantics can make it extremely difficult to implement correct meta-data ordering.

SPIN, because of its relying on type-safety for extensibility, must manage issues such as M3/C crossing, garbage collection; it must provide guards against long running extensions and defend against possible buggy extensions that might end up using too many system resources.

Note that the question asked for problems that were created by those OSs themselves, that is, problems that are due to, or at least related to, the extensibility model those OSs provide.

5. SFI (12 pts)

a) (3 pts) Why does the SFI paper use the term fault domain and not protection domain?

Protection would imply guarding against reading other domain’s memory; SFI only protects against writes to other domains.

b) (3 pts) What’s the equivalent of changing page tables (in a hardware-based system) in SFI?

On a domain crossing, SFI must reload the dedicated registers that hold the segment/shift identifier of the current domain with the values valid for the target
domain, just like a hardware-based OS must switch page tables when crossing domains.

c) (6 pts) The Nooks paper points out that although SFI solves the problem of isolation, it does not solve the problem of recovery. Name an approach by which SFI could be extended to perform recovery. Name one advantage and one disadvantage of the approach when applied to SFI.

SFI could be extended to implement an undo log. Before each write, it could record the write in a log that would store the old and the new value. This would require one additional dedicated register to hold the current location in the log, plus some way to identify the bounds of the log space. In this way, if a fault occurs, all writes could be undone. An advantage of this approach is that it naturally fits the SFI model of interposing on writes. A disadvantage is that it would log all state, even changes that wouldn’t have to be undone, so it would likely incur high overhead. (Note that the VINO system didn’t choose to go that route.)