Exam answers.

1.
a)
Any one of: The user might have to leave a session. The user realises he has started an incorrect command. The process looks as though it is not going to terminate due to deadlock, infinite loop, unavailable resource. etc.
Any one of: The process looks as though it is not going to terminate due to deadlock, infinite loop, unavailable resource. The machine is overloaded with work. Something appears to be wrong with the computer. etc.
Any one of: The machine is being shut down. The process has attempted to execute a privileged instruction or access a resource it does not have the required privileges for. The process has exceeded a resource limitation. etc.

b)
Keyboard interrupt causes a jump to the keyboard interrupt handling routine (indirectly).
Either the handler itself or an input filter on the way to sending the keypress to the process recognizes the key sequence as the one to suspend a process.
A call is made the suspend function.
The state of the process is changed to suspended.
As part of the context switch the running state of the process is preserved in the process control block (or elsewhere).
The process is removed from the ready queue (or equivalent).

c)
Similarities: Both cannot run. They are not scheduled nor in any ready queue. Any resources they currently have they keep.
Differences: A suspended process is not waiting for a resource, it is waiting to be resumed. i.e. It is not in a queue waiting for a resource.

2.
a)
If the user did not have read permission to the buffer memory location, the data could be written to the "file" and then accessed later by the user reading the file.

b)
The buffer is allocated on the stack beneath the return address of the current function. By writing more data than the buffer has been allocated the data above the buffer including the return address can be overwritten. By knowledge of the stack structure and the CPU architecture it is possible to place a system call to start a program on the stack and alter the original return address to jump to that code.

c)
Check the range of arrays used as buffers.

3.
a)
It ensures that only one thread at a time can execute inside any synchronized blocks of a particular object.

b)
They move the current thread into the wait set of the Buffer object. This stops the thread from running until it is notified later. They are there because in these situations the Buffer state is such that the threads cannot continue i.e. either a producer is trying to insert a value into a full buffer or a consumer is trying to remove a value from an empty buffer.

c)
To stop external direct access to the values. We want all access to these variables protected by the synchronized methods. If this was not the case some thread could access and change these variables while another thread was executing within one of the synchronized methods leading to inconsistent state and errors.

d)
The consumer calls notifyAll.
This causes all producers currently blocked to be woken up from the wait() in insert.
One producer is selected by the scheduler to run first.
This one checks the condition on the while loop and carries on and puts its data into the buffer.
All other newly awoken producers run one after another (only one at a time) and check the while condition and go back to waiting.

4.
a)
Deadlock prevention in a distributed environment. Deadlock prevention alone is acceptable.

b)
The wound-wait scheme because a young process gets restarted when it holds a resource wanted by an older one. When it restarts it may wait for the resource but doesn't restart again. In the wait-die scheme a young process keeps restarting whenever the resourse is still held by an older process.

5.
a)
If both have an R there are no available Rs. In this case neither P nor Q is guaranteed enough Rs to complete.

b)
P releases R
Q requests second R
Q finishes
P requests R
P requests second R
P finishes

6.
a)
ACLs. The information list of <domain, rights> is associated with the object.

b)
Capabilities. The information list of <object, rights> is associated with the domain.

c)
ACLs. Capability information can be scattered throughout all domains, including over networks. Changing a required capability invalidates all existing ones.

d)
Capabilities. They can be distributed easily over a network. With ACLs we need to know explicitly which domain is getting the access privilege.

e)
Both. They can both set access rights in any desired way.

7.
a)
Less flexibility. Both acls and capabilities allow very specific control over access. This is in at least two ways, the types of access allowed and who or what has the access. e.g. It is impossible to give a group of users one type of access, another group a different type of access and no access to anyone else. This is easy to do with either acls or capabilities.

b)
Simplicity, especially of administration.

8.
a)
Any two of:
To help with speed mismatches. A slow device might buffer its output so that a fast device doesn't have to deal with it until there is a reasonable amount of data.
To help with different data transfer sizes. The buffer gets data from one device and dishes it out to the other device in chunks it can deal with.
To cache data. Data from a disk drive could be kept in a buffer cache until the space is needed for something else. In the meantime any access of that data can come from the buffer rather than having to access the disk again.
To preserve copy semantics. Data from a user level buffer can be copied to a kernel buffer on a write call to stop the user modifying the data after the call and before the actual write.
etc.

b)
When data is going to be available very soon. e.g. Disk drives that don't have DMA or terminal concentrators for large numbers of terminals.