1)

a. Main memory is not suitable for permanent program storage or backup purposes, because of the fact that main memory stores data via an electrical charge, and without an electrical charge (e.g., the computer is turned off), and the fact that memory cells were not designed to keep a charge in them, the main memory will not be able to store anything. The only way to keep data stored in main memory, is to leave the computer turned on all the time which is simply not sustainable.
b. There are quite a few disadvantages to storing information on a magnetic disk drive as opposed to main memory, but I think the main one would be the drive's lifespan. While magnetic disk drives are much slower than main memory, it is still useable, just it might be a tad annoying to do so. The real issue is that magnetic disk drives have quite a few moving parts in them, which just like anything else, can wear out. If the arm holding the read-write head were to collide with the surface of the drive – something that can very reasonably happen, it would dig a hole in itself, and thus destroy a large amount of data. This is an issue that main memory simply doesn't have due to it's lack of moving parts.

2)

a. The dining-philosophers problem is a problem where you have five silent philosophers sitting at a round table with bowls of spaghetti. In between each philosopher, is a fork. Philosophers are allowed to eat only if forks are available on either side of them, and if they aren't (e.g. someone on either side of them is using that side's fork), they must sit and think until both are available.
b. This relates to operating systems, as the problem here represents concurrent processes with a set number of resources. In this problem, the philosophers represent programs that all want to run (eat) at the same time, however each program requires a certain amount of resources (forks) to properly execute, and under these circumstances, not all programs can run at the same time. A solution to this problem would represent how to allocate resources optimally between processes.