1) A) On Intel x86 based PCs, during booting, what does BIOS do?

Power-on self-test (POST), which checks, identifies and initializes system devices such as the CPU, RAM, etc. It also looks for a bootable device.

b) How many sectors does BIOS load from the boot device?

Depending on the system there may be multiple sectors. Hard disks may have multiple partitions. One hard drive has one boot sector.

c) Where in memory does BIOS load the booter?

When booting from a hard disk, BIOS loads the MBR (master boot record) booter to the memory location 0x0000, 0x7C00 and executes it as usual.

In the early days, a PC is only guaranteed to have 64KB of RAM memory. The memory below 0x07C00 is reserved for interrupt vectors, BIOS and BASIC, etc. The first OS usable memory begins at 0x08000. So the booter is loaded to 0x07C00, which is 1KB below 0x08000

2) BIOS loads only 512 bytes of a booter into memory, which is only the beginning part of the booter. How does the booter load the remaining part of the booter into memory?

It sends CPU to execute the startup code of the OS kernel, which starts up the OS.

3) Assume a COMPLETE booter is loaded at the segment 0x9000. WHY do we have to set the CPU's segment registers CS,DS,SS,ES to 0x9000?

It is the beginning address for real memory and follows one segment memory model.

4) How do you find the file /boot/mtx?

Find the inode of /boot, then with that inode we can find the inode of mtx

5) How to load the (disk) blocks of /boot/mtx to the segment 0x1000?

Getblk(blk, buf) //Bios uses ES, buf as memory address. (segment, offset) = (ES, buf)

ES points at 0x1000, so instead of reading a block and writing to a buffer, the block is read and written to (ES, 0) and thus puts all the blocks in memory.