(1) Explain how the NachOS FS manage and find free block space? Where is this information stored on the raw disk (which sector)?

FileHeader::Allocate 中，依據file size算出需要存在幾個sector裡面，並依據所需sector數量，呼叫 freeMap->FindAndSet()，尋找在bitmap中還空閒的bit編號，同時也就是空閒的sector編號。

在FileSystem的建構子中，可以看到FreeMapSector，而這個常數的值為0，也就是存在sector 0。

(2) What is the maximum disk size that can be handled by the current implementation? Explain why.

32 track \* 32 sectors/track = 1024 sectors.

1024 sectors \* 128 bytes/sector = 128KB.

But files have a size limit of about 4KB, and current file system support only ten files, so lesser than 40KB will be actually used.

(3) Explain how the NachOS FS manage the directory data structure? Where is this information stored on the raw disk (which sector)?

在FileSystem的建構子中，如果格式化的flag為true，則會呼叫Directory的建構子，並建立一個能存放10個檔案的簡單空資料夾。 如果沒有要格式化，則從DirectorySector讀取資料夾資訊。

DirectorySector = 1，所以是存在sector 1。

(4) Explain what information is stored in an inode, and use a figure to illustrate the disk allocation scheme of current implementation.

Inode -> FileHeader in NachOS, so the information stored in it are numBytes, numSectors, dataSectors.

(No figure for now) (Please go to FileSystem::Create() )

The scheme is Indexed Allocation with only direct block.

10 files Indexed with 0~9.

(5) Why is a file limited to 4KB in the current implementation?

Since each File header will be stored in a sector, this means only 128 bytes of information can be stored in it.

128 – sizeof(numBytes) – sizeof(numSectors) = 120 bytes.

120 bytes / sizeof(int) = 30.

So for each file, the number of sectors it occupied cannot be more than 30, which result in 30 \* 128 bytes = 3840 bytes, about 4KB.