1). The time required to copy the file using *read_write* varies with the size of the buffer specified. Smaller buffer sizes take longer. The time required for *memmap* varies much less regardless of how you perform the copy. Discuss why this is, and in your discussion consider the influence of: (1) the overhead of the read() and write() system calls and (2) the number of times the data is copied under the two methods.

Read and write system calls move data from the hardware to a buffer, then to the user, after used by the user, it will return back to the hardware storage. The process is really inefficient. However, the mmap removes part of the overhead of the data path from buffer to storage and back to hardware, which make the operations more efficient. The user will be able to access data directly from disk and buffers.

2). When you use the *read_write* command as supplied, the size of the copied file varies with the size of the buffer specified. When you use the *memmap* command implemented the size of the source and destination files will match exactly. This is because there is a mistake in the *read_write* code. What is the mistake, and how can it be corrected?

Read_write fille allocates memory inefficiently which potentially causes the running is slow. However, if the read command tracks the size of the input reads, so we write that much of the bytes will probably make it faster.