1 Question 1

1.1 Task a)

Using `bsub "grep . /sys/devices/system/cpu/cpu0/cache/index*/* "` we get the results listed in table 1.

	L1		L2	L3
Cache Type	Data	Instruction	Unified	Unified
total size (KB)	32	32	256	8192
cache line size (KB)	64	64	64	64

Table 1: L1, L2 and L3 cache key figures of Euler compute node.

1.2 Tasks b), c) and d)

The results of running the compiled program on Euler are shown in figure 1. When jumping randomly through the array (task b), the performance drops are at the KB marks which could be expected based on table 1: Sharp drops at `64 KB` and `8192 KB` as the array doesn't fit in the next level cache. From that point on the random jumps trash the newly read cache at each jump (except when the random jump is within the cache line, the chance for which gets increasingly small as N increases).
Sequential reads (task c) obviously suffer no performance drop as the data ahead always gets read into the cache in due time and thus is available with almost no delay. Jumping by 16 integers (which equals 64 bit as the size of an `int` in 64-bit `C++` was found to be 4 bit) gives performance drops at the marks seen in the random task, but they are much less pronounced: While we're able to read from cache lines, reading the increasingly large arrays makes memory access a bottle neck.

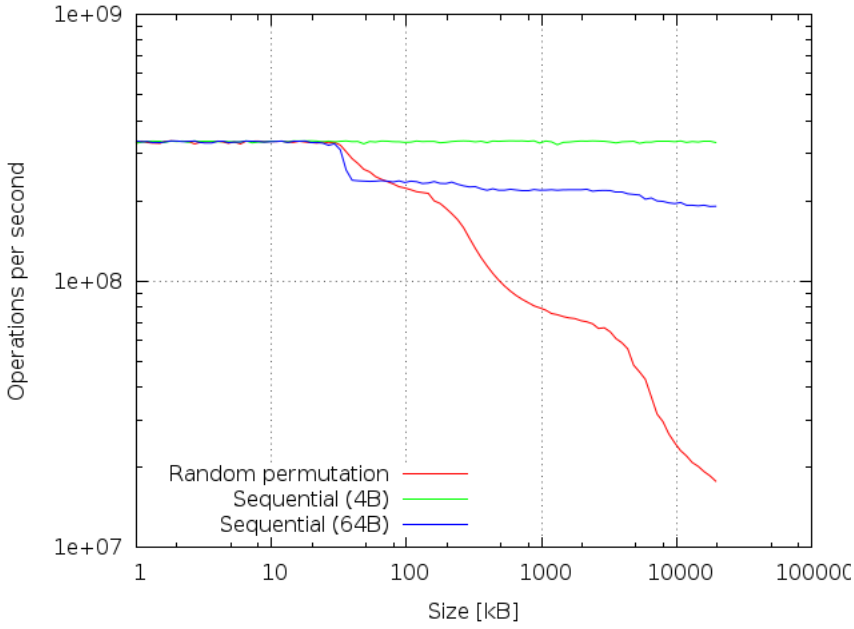

Figure 1: Operations per second when reading from differently sized `int` arrays on Euler compute cluster.

2 Question 2

Ran out of time :)

3 Question 3

Ran out of time :)