John King

HW #4

1)
a)#pragma omp parallel for num_threads(thread_count)
b)it would not parallelize because it won't know when the loop will terminate before the loop starts because of the && (!flag).
c)#pragma omp parallel for num_threads(thread_count)
d)#pragma omp parallel for num_threads(thread_count)
e)it would not parallelize because it won't know when the loop will terminate before the loop starts because of the break statement.
f)dotp=0;

```
#pragma omp parallel for num_threads(thread_count)
for (i=0; i < n; i++) {
   term = a[i] * b[i];
#pragma omp critical
   dotp += alpha * term;
}
```

g)#pragma omp parallel for num_threads(thread_count)
h)assuming that n>2k it will not parallelize because of data dependencies.

2)

n	p	q	n^2/p	n/p	n[b]/log(q)
1000	1	1	1000000	1000	undefined
2000	4	2	1000000	500	1000
4000	16	4	1000000	250	500
8000	64	8	1000000	125	333.3333
16000	256	16	1000000	62.5	250
32000	1024	32	1000000	31.25	200
64000	4096	64	1000000	15.625	166.66
128000	16384	128	1000000	7.8125	142.857

3)
a)The traditional benchmark is all about flops and little else. The author says that this was because it was designed in the 70's and back then problems mostly cared about flops. Models

that are being created now focus more on problems that involve lots of message passing, like a sparse matrix. This new benchmark takes this into consideration and weighs its score heavily on the capabilities of the network and memory access.

b)
i) The Cray system (Hopper) was by far the most variable.
ii)The leading cause of this, according to the authors, is interference from other jobs running on the same network.
iii)Instead of using a totally automated scheduler for batch jobs the Hopper used a system in which the users set the amount of work a batch job would do in order to assure that results could be saved every day. This lead to more variable results, less average work, slower time until completion, and the need for more batch slots.