CS622 Assignment#1 Solution Sketch

1. Please refer to the posted codes (inclusive.c, nine.c, exclusive.c). The generated results are shown in Table 1.

Table 1. L2 and L3 cache miss count

App.	Inclusive		NINE		Exclusive	
	L2	L3	L2	L3	L2	L3
bzip2	5398166	1446388	5397576	1445846	5397576	889221
gcc	3036461	1373402	3029809	1366248	3029809	1242824
gromacs	336851	170531	336724	170459	336724	159302
h264ref	969678	342146	965624	333583	965624	143681
hmmer	1743421	391226	1735322	376344	1735322	300046
sphinx3	8820349	8207362	8815130	8205144	8815130	7220776

The general trend in L2 cache misses: Going from inclusive to NINE, the L2 cache misses drop in number. This is because in NINE, the inclusion victims arising from back-invalidations are absent. However, notice that the difference is very small indicating that back-invalidations are not a big problem. Going from NINE to exclusive, the L2 cache misses remain unchanged in number.

The general trend in L3 cache misses: Going from inclusive to NINE, the L3 cache misses drop in number. This is because in NINE, the inclusion victims arising from back-invalidations are absent. As a result, the accesses to these victims get satisfied in the L2 cache even though these blocks are not present in the L3 cache. In an inclusive L3 cache, these accesses miss in both L2 and L3 caches increasing their the miss counts of both L2 and L3 caches. Going from NINE to exclusive, the L3 cache misses drop significantly in number. This is because an exclusive hierarchy offers bigger effective capacity.

2. Please refer to the posted codes (cold.c, inclusive-FA-MIN.c, inclusive-FA-LRU.c, inclusive.c). The generated results are shown in Table 2. The results for inclusive-FA-LRU were not asked, but I show them here just to show why we need to use the MIN algorithm to correctly estimate capacity misses. FA stands for fully-associative and SA stands for set-associative. Although our focus is on L3 cache misses, I have included the L2 cache misses also in the results. Note that in all these results, the L2 cache is same as in the previous problem.

Table 2. L2 and L3 cache miss count

App.	Cold	Inclusive-FA-MIN		Inclusive-FA-LRU		Inclusive-SA-LRU	
		L2	L3	L2	L3	L2	L3
bzip2	119753	5351953	536836	5397991	1361401	5398166	1446388
gcc	773053	2946879	939289	3035483	1369924	3036461	1373402
gromacs	107962	321640	143254	336940	169368	336851	170531
h264ref	63703	960590	111605	969235	335880	969678	342146
hmmer	75884	1732342	153447	1743222	377024	1743421	391226
sphinx3	122069	8680310	3068580	8818925	8387248	8820349	8207362

We make three observations. First, the L2 cache miss count generally increases very slowly as we go from Inclusive-FA-MIN to Inclusive-FA-LRU to Inclusive-SA-LRU (there are exceptions, though). This is primarily due to inclusion victims. Second, the L3 cache miss count in Inclusive-FA-LRU is marginally less than Inclusive-SA-LRU for all applications except sphinx3. In sphinx3, Inclusive-FA-LRU has more L3 cache misses than Inclusive-SA-LRU. This indicates that the LRU policy prevents the fully-associative cache from attaining its full potential. Third, Inclusive-FA-MIN has the least number of L3 cache misses. From the L3 cache miss counts of Inclusive-FA-MIN and Inclusive-SA-LRU, we derive the capacity and conflict miss counts for the Inclusive-SA-LRU configuration. The number of L3 cache misses in Inclusive-SA-LRU is cold+capacity+conflict and the number of L3 cache misses in Inclusive-FA-MIN is cold+capacity. See Table 3 on the next page.

Note that in this calculation, we have classified all misses arising due to mistakes of the replacement policy as conflict misses. The alternative option would be to have a fourth category of misses called replacement misses.

Table 3. Cold, capacity, and conflict misses in the inclusive set-associative L3 cache

App.	Cold	Capacity	Conflict	TOTAL
bzip2	119753	417083	909552	1446388
gcc	773053	166236	434113	1373402
gromacs	107962	35292	27277	170531
h264ref	63703	47902	230541	342146
hmmer	75884	77563	237779	391226
sphinx3	122069	2946511	5138782	8207362