simpleloop.c	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Memory Size = 50						
Rand	70.9277%	7263	2977	2927	220	2707
FIFO	70.9082%	7261	2979	2929	213	2716
LRU	72.8125%	7456	2784	2743	88	2646
CLOCK	72.6855%	7443	2797	2747	99	2648
OPT	73.9355%	7571	2669	2619	18	2601
Memory Size = 100						
Rand	72.8418%	7459	2781	2681	61	2620
FIFO	73.0762%	7483	2757	2657	44	2613
LRU	73.7793%	7555	2685	2585	2	2583
CLOCK	73.7207%	7549	2691	2591	5	2586
OPT	74.1895%	7597	2643	2543	0	2543
Memory Size = 150						
Rand	73.5156%	7528	2713	2562	17	2545
FIFO	73.4668%	7523	2717	2567	16	2551
LRU	73.7988%	7557	2683	2533	0	2533
CLOCK	73.7793%	7555	2685	2535	0	2535
OPT	74.1895%	7597	2643	2493	0	2493
Memory Size = 200						
Rand	73.5156%	7528	2712	2512	15	2497
FIFO	73.5449%	7531	2709	2509	12	2497
LRU	73.7988%	7557	2683	2483	0	2483
CLOCK	73.7891%	7556	2684	2484	0	2484
OPT	74.1895%	7597	2643	2443	0	2443

matmul.c	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Memory Size = 50						
Rand	65.5209%	1892187	995725	995675	956175	39500
FIFO	60.9659%	1760642	1127270	1127220	1083228	43992
LRU	63.9452%	1846682	1041230	1041180	1040068	1112
CLOCK	63.9439%	1846645	1041267	1041217	1040102	1115
OPT	79.6581%	2300455	587457	587407	586318	1089
Memory Size = 100						
Rand	88.8183%	2564995	322917	322817	315368	7449
FIFO	62.4797%	1804360	1083552	1063452	1061224	22228
LRU	65.1494%	1881456	1006456	1006356	1005276	1080
CLOCK	63.9526%	1846895	1041017	1040917	1039833	1084
OPT	96.7867%	2795114	92798	92698	91611	1087
Memory Size = 150						
Rand	96.6502%	2791172	96740	96590	94210	2380
FIFO	98.8085%	2853502	34410	34260	32944	1316
LRU	98.8612%	9855025	32887	32737	31657	1080
CLOCK	98.8500%	2854702	33210	33060	31975	1085
OPT	99.0784%	2861297	26615	26465	25378	1087
Memory Size = 200						
Rand	98.0467%	2831501	56411	56211	54604	1607

FIFO	98.8265%	2854023	33889	33689	32434	1255
LRU	98.8616%	2855036	32876	32676	31596	1080
CLOCK	98.8607%	2855009	32903	32703	31622	1081
OPT	99.3329%	2868647	19265	19065	17978	1087

blocked.c	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Memory Size = 50						
Rand	99.6530%	2409745	8391	8314	5827	2514
FIFO	99.7318%	2411651	6485	6435	4175	2260
LRU	99.7842%	2412917	5219	5169	2814	2355
CLOCK	99.7819%	2412863	5273	5223	2877	2346
OPT	99.8466%	2414426	3710	3660	2572	1088
Memory Size = 100						
Rand	99.7785%	2412780	5356	5256	3479	1777
FIFO	99.8206%	2413798	4338	4238	2758	1480
LRU	99.8434%	2414349	3787	3682	2601	1081
CLOCK	99.8329%	2414096	4040	3940	2624	1316
OPT	99.8755%	2415125	3011	2911	1835	1076
Memory Size = 150						
Rand	99.8168%	2413707	4429	4279	2783	1496
FIFO	99.8252%	2413909	4227	4077	2653	1424
LRU	99.8441%	2414366	3770	3620	2559	1061
CLOCK	99.8369%	2414192	3944	3794	2575	1219
OPT	99.8954%	2415607	2529	2379	1299	1080
Memory Size = 200						
Rand	99.8405%	2414280	3856	3656	2308	1348
FIFO	99.8686%	2414959	3177	2977	1878	1099
LRU	99.8471%	2414439	3697	3497	2436	1061
CLOCK	99.8681%	2414946	3190	2990	1928	1062
OPT	99.9058%	2415857	2279	2079	1012	1067

test.c	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Memory Size = 50						
Rand	36.8453%	12324	21124	21074	557	20517
FIFO	36.8961%	12341	21107	21057	535	20522
LRU	39.4104%	13182	20266	20216	88	20128
CLOCK	39.3716%	13169	20279	20229	99	20130
OPT	39.8081%	13315	20133	20083	22	20061
Memory Size = 100						
Rand	38.4597%	12864	20584	20484	221	20263
FIFO	38.4687%	12867	20581	20481	215	20266
LRU	39.7034%	13280	20168	20068	2	20066
CLOCK	39.6885%	13275	20173	20073	5	20068
OPT	40.0054%	13381	20067	19967	7	19960
Memory Size = 150						
Rand	38.9709%	13035	20413	20263	121	20142
FIFO	38.9111%	13015	20433	20283	133	20150

LRU	39.7124%	13283	20165	20015	0	20015
CLOCK	39.7064%	13281	20167	20017	0	20017
OPT	40.1549%	13431	20017	19867	7	19860
Memory Size = 200						
Rand	39.1593%	13098	20350	20150	92	20058
FIFO	39.1264%	13087	20361	20161	97	20064
LRU	39.7124%	13283	20165	19965	0	19965
CLOCK	39.7094%	13282	20166	19966	0	19966
OPT	40.3044%	13481	19967	19767	7	19760

Description of the forth program (test.c):

Our fourth program is very simple, it is just a nested loop in which update information for each slot in the array and each slot has size of 4096 bytes (page size). I choose this program since nested loop that updating information is extremely common, but the algorithms have the worst performance compares to the trace file generated by the provided programs. The memory reference is interesting since the program has spatial locality (most of the time) and temporal locality since each inner iteration updates information for each sequential page. However, once it reaches the end of the allocated array, it goes back to slot at index 0 (performed by the outer loop) of the array, which explains the poor performance of all algorithms since most likely that most of the pages that cover the allocated array are evicted to the disk (either because it is the least recently used or it is the first page brought into memory). Thus, at the second outer iteration, the OS needs to bring most of pages back again. Therefore, even though the program has locality, but the memory accessing pattern form a corner case for most algorithms, indicates that locality does not guarantee success.

First required paragraph:

For FIFO policy, all memory accesses from the given programs do not suffer from Belady's anomaly, that is the hit rate keeps increasing as the memory size increases, thus the overall eviction count decreases as more memory resources available. However, it mostly performs worse than other algorithm since it violates how most programs behave, that is, it evicts the page no matter how many times it is being accessed before. Exact least recently used policy behaves closest to the optimal most of times since it utilize the principle of locality, that is pages used recently most likely be used again in the future; except the case occurs in the trace file from Blocked.c with memory size of 200 frame (it behaves worse than the clock algorithm), which may due to some corner case of memory access pattern in the trace file since locality does not guarantee higher hit ate. Clock algorithm is an approximate LRU algorithm, in which it gives each page a second chance; that is, each time it sees a page with reference bit of one, it would not evict that page until the second time it sees it or evict the page when it has ref bit of zero. Thus, the table statistics show that it mostly has hit rate lower than that from LRU since LRU uses exact bookkeeping information to tell which page to evict; but it mostly performs better than FIFO policy since FIFO does not has locality preferences. For optimal algorithm, it evicts a page that is not being used for the longest period among all the pages in memory, thus we see that it has the highest hit rate, thus lowest eviction count over all written algorithms. For algorithm that evict pages randomly, we expect that there are no edge cases behaviors, thus we see that it performs worse than other algorithms it performs much better with memory size of 100 frames in matmul's trace file, worse performance of other algorithms (except opt) may due to some corner case memory accessing pattern in the trace file again. Nonetheless, all above algorithms have better hit rate as more memory available

Secondary required paragraph:

We see that the exact LRU algorithm has better hit rate as memory size increases. We can attribute this behavior to a few reasons. Firstly, LRU is one of the policies that uses the principle of locality, specifically, LRU obeys the temporal locality, that is recently used pages mostly likely be demanded in the future and that type of memory accessing pattern occurs in most of the programs, thus more memory available implies that it has more recently used pages stored and evicts the one that has not been accessed for longer time compare to less memory available. Secondly, LRU has the stack property, that is, memory with bigger size contains the pages from memory with smaller size, thus it only has the same or even better performance, never worse, that is no Belay's anomaly is possible. Overall, compare to other implemented algorithms, it has better hit rate and the performance is closest to the optimal algorithm. It is better than Rand and FIFO policies since both algorithms does not obey what most programs behave. It is better than Clock algorithm since LRU has more precise bookkeeping information to evict a victim page. For a large trace file like matmul.c, it most likely has different memory accessing pattern, which increase the possibility that a corner case occurs thus LRU performs worse than Rand, but as memory space gets bigger, hit rate has a drastic increase.