Simpleloop Tables:

Rand							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	70.9706	7239	2961	2911	131	2780
	100	72.8922	7435	2765	2665	39	2626
	150	73.3922	7486	2714	2564	10	2554
	200	73.4412	7491	2709	2509	10	2499
FIFO							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	71.0392	7246	2954	2904	122	2782
	100	73.0098	7447	2753	2653	32	2621
	150	73.3922	7486	2714	2564	8	2556
	200	73.4706	7494	2706	2506	6	2500
Clock							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	72.7549	7421	2779	2729	67	2662
	100	73.6961	7517	2683	2583	2	2581
	150	73.7157	7519	2681	2531	0	2531
	200	73.7157	7519	2681	2481	0	2481
LRU							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	72.8039	7426	2774	2724	65	2659
	100	73.7059	7518	2682	2582	2	2580
	150	73.7255	7520	2680	2530	0	2530
	200	73.7255	7520	2680	2480	0	2480
OPT							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	57.598	5875	4325	4275	848	3427
	100	72.8824	7434	2766	2666	71	2595
	150	74.1176	7560	2640	2490	0	2490
	200	74.1176	7560	2640	2440	0	2440

Matmul Tables:

Rand		Matmul					
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	65.5219	1892183	995681	995631	478516	517115
	100	88.7762	2563736	324128	324028	158429	165599
	150	96.6794	2791971	95893	95743	46873	48870
	200	98.0426	2831338	56526	56326	27521	28805
FIFO							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	60.9658	1760609	1127255	1127205	541679	585526
	100	62.4795	1804322	1083542	1083442	530670	552772
	150	98.8086	2853457	34407	34257	16665	17592
	200	98.8266	2853978	33886	33686	16250	17436
CLOCK							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	63.9448	1846639	1041225	1041175	520102	521073
	100	65.3103	1886073	1001791	1001691	500466	501225
	150	98.798	2853151	34713	34563	16942	17621
	200	98.8612	2854978	32886	32686	15994	16692
LRU							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	63.9449	1846643	1041221	1041171	520102	521069
	100	65.1489	1881411	1006453	1006353	502791	503562
	150	98.8613	2854980	32884	32734	16018	16716
	200	98.8617	2854991	32873	32673	15985	16688
OPT							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	60.8014	6085	3923	3873	1632	2241
	100	86.8905	8696	1312	1212	432	780
	150	98.3114	9839	169	19	6	13
	200	98.4013	9848	160	0	0	0

USED INPUT ./runit matmul 10 FOR OPT RESULTS

Blocked Tables:

Rand							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.6573	2409849	8287	8237	3040	5197
	100	99.7807	2412832	5304	5204	1857	3347
	150	99.8185	2413748	4388	4238	1535	2703
	200	99.8404	2414276	3860	3660	1320	2340
FIFO							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.7317	2411649	6487	6437	2113	4324
	100	99.8208	2413802	4334	4234	1397	2837
	150	99.8254	2413913	4223	4073	1366	2707
	200	99.8689	2414965	3171	2971	1001	1970
CLOCK							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.7617	2412374	5762	5712	1674	4038
	100	99.822	2413831	4305	4205	1332	2873
	150	99.8438	2414358	3778	3628	1324	2304
	200	99.8673	2414926	3210	3010	1057	1953
LRU							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.7846	2412927	5209	5159	1433	3726
	100	99.8436	2414354	3782	3682	1327	2355
	150	99.8443	2414370	3766	3616	1318	2298
	200	99.8473	2414443	3693	3493	1281	2212
OPT							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	73.092	5210	1918	1868	891	977
	100	95.5247	6809	319	219	93	126
	150	97.9377	6981	147	0	0	0
	200	97.9377	6981	147	0	0	0

USED INPUT ./runit blocked 10 25 FOR OPT RESULTS

Childcreate Tables:

Rand							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.1319	55839	405	355	110	245
	100	99.6325	56121	207	107	12	95
	150	99.7302	56176	152	2	0	2
	200	99.7302	56176	152	0	0	0
FIFO							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.281	55923	405	355	110	245
	100	99.6733	56144	184	84	0	84
	150	99.7231	56172	156	6	0	6
	200	99.7302	56176	152	0	0	0
CLOCK							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.4763	56033	295	245	69	176
	100	99.7106	56165	163	63	0	63
	150	99.7284	56175	153	3	0	3
	200	99.7302	56176	152	0	0	0
LRU							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	99.5136	56054	274	224	61	163
	100	99.7106	56165	163	63	0	63
	150	99.7302	56176	152	2	0	2
	200	99.7302	56176	152	0	0	0
OPT							
	mem	Hit rate	Hit count	Miss count	Evict count	Clean evict count	Dirty evict count
	50	59.8956	33738	22590	22540	10331	12209
	100	78.611	44280	12048	11948	5302	6646
	150	99.4195	56001	327	177	87	90
	200	99.7302	56176	152	0	0	0

Fourth Program:

We used a c program from our second year CS course CSC209 which used the fork to generate children processes. We ran the program to create 50,000 children processes and we were curious to see how the replacement algorithms would handle a large use of memory in parallel processes and whether the randomness of the execution caused by processor time affected the memory. The result was that the algorithms did very well with the 50,100,150,200 slots of memory given, even the random algorithm did well. We believe this was the result because we were not doing anything complex with the children besides the fact that we were running them in parallel.

Observations:

We found that in all cases the hit rate followed the trend FIFO<CLOCK<LRU with Rand sometimes giving better results with a smaller amount of memory and always worse results than the others given a larger amount of memory. We noticed that performance of all algorithms would increase given additional memory which is to be expected, less swaps are required, no matter which algorithm we choose. The random algorithm behaved very strangely in the matmul program, gaining a bigger hit rate than the other algorithms given 50 and 100 slots in physical memory and then performing worse than the others given more memory, as expected. We were surprised to see that the difference in algorithms although different in execution did not produce great variability amongst each other. In effect, running the random algorithm was only slightly less effective than running the LRU which outperformed all other algorithms given the most memory. It is surprising because LRU uses knowledge of the pass to make more accurate predictions of the future but simply running a random algorithm proved almost as effective for these simple programs.

FIFO vs LRU:

LRU would consistently outperform FIFO on all programs and with any amount of memory because LRU leverages probability distributions better than FIFO which does not account for anything. If any process happens very frequently, LRU is able to hold on to it for longer than FIFO and therefore save time on extra swaps. As we see in these tests, this makes a difference, but that difference is pretty small. On the other hand, any improvement on the average of thousands of programs can save minutes or even hours of total processing time so it seems that LRU is superior than FIFO at least of the test programs we ran.