1.Task 2 table:

- `tracefile: tr-simpleloop.ref`

Algorithm	Hit rate	Hit count	Miss count	Overall eviction count	Clean eviction count	Dirty eviction count
RAND						
m=50	70.8822	7247	2977	2927	223	2704
m=100	72.7993	7443	2781	2681	61	2620
m=150	73.4742	7512	2712	2562	17	2545
m=200	73.4742	7512	2712	2512	15	2497
FIFO						
m=50	70.8725	7246	2978	2928	213	2715
m=100	73.0340	7467	2757	2657	44	2613
m=150	73.4253	7507	2717	2567	16	2551
m=200	73.5035	7515	2709	2509	12	2497
LRU						
m=50	72.7895	7442	2782	2732	87	2645
m=100	73.7285	7538	2686	2586	2	2584
m=150	73.7578	7541	2683	2533	0	2533
m=200	73.7578	7541	2683	2483	0	2483
CLOCK						
m=50	72.7700	7440	2784	2734	87	2647
m=100	73.6796	7533	2691	2591	5	2586
m=150	73.7578	7541	2683	2533	0	2533
m=200	73.7480	7540	2684	2484	0	2484
OPT						
m=50	73.8948	7555	2669	2619	18	2601
m=100	74.1491	7581	2643	2543	0	2543
m=150	74.1491	7581	2643	2493	0	2493
m=200	74.1491	7581	2643	2443	0	2443

- `tracefile: tr-matmul.ref`

Algorithm	Hit rate	Hit count	Miss count	Overall eviction count	Clean eviction count	Dirty eviction count
RAND						
m=50	94.6636	6528	368	318	153	165
m=100	97.3028	6710	186	86	9	77
m=150	97.8973	6751	145	0	0	0
m=200	97.8973	6751	145	0	0	0
FIFO						

m=50	94.8231	6539	357	307	139	168
m=100	97.4768	6722	174	74	0	74
m=150	97.8973	6751	145	0	0	0
m=200	97.8973	6751	145	0	0	0
LRU						
m=50	96.0992	6627	269	219	80	139
m=100	97.8103	6745	151	51	0	51
m=150	97.8973	6751	145	0	0	0
m=200	97.8973	6751	145	0	0	0
CLOCK						
m=50	96.0847	6626	270	220	83	137
m=100	97.7378	6740	156	56	0	56
m=150	97.8973	6751	145	0	0	0
m=200	97.8973	6751	145	0	0	0
OPT						
m=50	97.5348	6726	170	120	18	102
m=100	97.8973	6751	145	45	0	45
m=150	97.8973	6751	145	0	0	0
m=200	97.8973	6751	145	0	0	0

- `tracefile: tr-blocked.ref`

Algorithm	**Hit rate**	**Hit count**	**Miss count**	**Overall eviction count**	**Clean eviction count**	**Dirty eviction count**
RAND						
m=50	94.5375	6663	385	335	165	170
m=100	97.2900	6857	191	91	11	80
m=150	97.9001	6900	148	0	0	0
m=200	97.9001	6900	148	0	0	0
FIFO						
m=50	94.7361	6677	371	321	150	171
m=100	97.4461	6868	180	80	0	80
m=150	97.9001	6900	148	0	0	0
m=200	97.9001	6900	148	0	0	0
LRU						
m=50	96.0131	6767	281	231	85	146
m=100	97.7866	6892	156	56	0	56
m=150	97.9001	6900	148	0	0	0
m=200	97.9001	6900	148	0	0	0
CLOCK						
m=50	95.9705	6764	284	234	90	144
m=100	97.7299	6888	160	60	0	60

m=150	97.9001	6900	148	0	0	0
m=200	97.9001	6900	148	0	0	0
OPT						
m=50	97.4745	6870	178	128	21	107
m=100	97.9001	6900	148	48	0	48
m=150	97.9001	6900	148	0	0	0
m=200	97.9001	6900	148	0	0	0

- `tracefile: tr-myprog.ref`

Algorithm	**Hit rate**	**Hit count**	**Miss count**	**Overall eviction count**	**Clean eviction count**	**Dirty eviction count**
RAND						
m=50	95.5308	6947	325	175	114	161
m=100	97.6210	7099	173	73	7	66
m=150	98.0336	7129	143	0	0	0
m=200	98.0336	7129	143	0	0	0
FIFO						
m=50	95.3933	6937	335	285	130	155
m=100	97.6760	7103	169	69	0	69
m=150	98.0336	7129	143	0	0	0
m=200	98.0336	7129	143	0	0	0
LRU						
m=50	96.7959	7039	233	183	59	124
m=100	97.9785	7125	147	47	0	47
m=150	98.0336	7129	143	0	0	0
m=200	98.0336	7129	143	0	0	0
CLOCK						
m=50	96.5455	6959	249	199	68	131
m=100	97.8635	7054	154	54	0	54
m=150	98.0336	7129	143	0	0	0
m=200	98.0336	7129	143	0	0	0
OPT						
m=50	97.7173	7106	166	116	18	98
m=100	98.0336	7129	143	43	0	43
m=150	98.0336	7129	143	0	0	0
m=200	98.0336	7129	143	0	0	0

2. Describe the fourth program

We generated two array of integers and each array has size 4096. We then randomly assign number for them, add the first array to the second, and perform a quick sort on the second array.

3. compare the various algorithms

We have compared each algorithm's hit rate and we have found that the hit rate of FIFO is almost the same as the hit rate of RAND, LRU hit rate is a little bit higher than the previous two, ClOCK hit rate is almost the same as LRU hit rate, and OPT hit rate is almost the highest one among the five algorithms. Thus, OPT is the most optimal approach, since it always chooses the page that will not use for the longest time in the future. When the memory size increases, the hit rate of FIFO may decrease. We also found that when the memory size increases, the miss count and the eviction count in each algorithm will decrease every time we perform the trace file test.

4. LRU

We have noticed that the hit rate of the LRU policy increases as the size of memory increases in the test of each trace file. This approach is to evict the least-recently-used entry, in other words, if we need to evict a page, LRU choose the entry that has not recently been used for the longest time. As a result, if we increase the memory size, we are more likely to store more recently used pages. Thus, it takes the advantage of locality in the memory-reference stream and increases the hit rate. We have also found that in the matmul case, the eviction of LRU decreases when the memory size increases. LRU approach is better than RAND, and it does not have belady anomy.