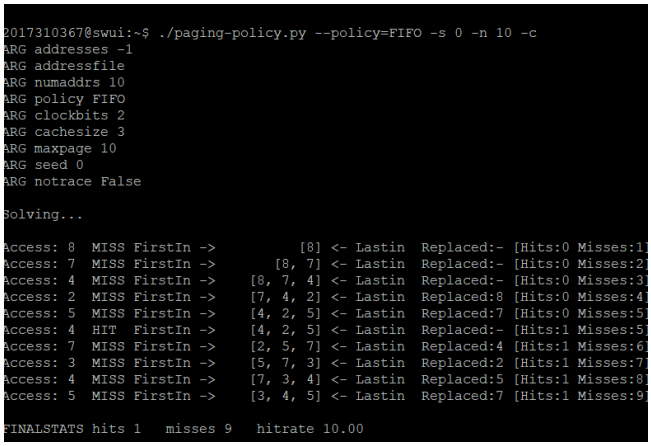
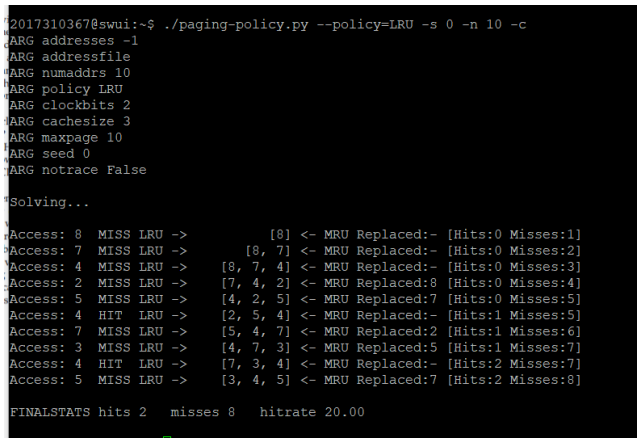
1. Generate random addresses with the following arguments: -s 0 -n 10, -s 1 -n 10, and -s 2 -n 10. Change the policy from FIFO, to LRU, to OPT. Compute whether each access in said address traces are hits or misses.



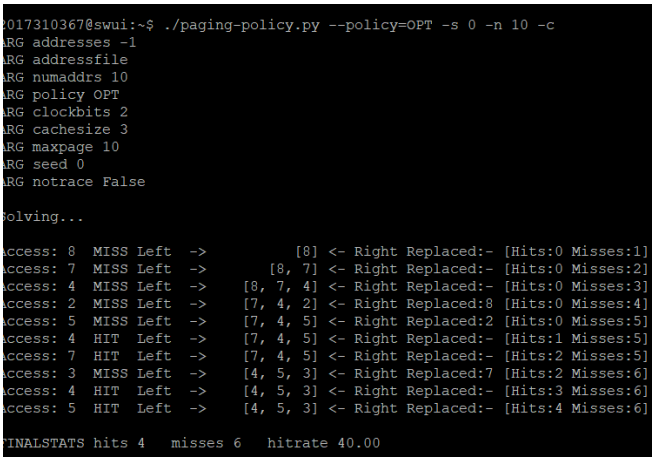
FIFO when seed is 0.

Number of hits is 1, Number of misses is 9, therefore hit-rate is 10%.



LRU when seed is 0.

Number of hits is 2, Number of misses is 8, therefore hit-rate is 20%.



OPT when seed is 0.

Number of hits is 4, Number of misses is 6, therefore hit-rate is 40%.

```
2017310367@swui:~$ ./paging-policy.py --policy=FIFO -s 1 -n 10 -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS FirstIn ->          [1] <- Lastin  Replaced:-  [Hits:0 Misses:1]
Access: 8  MISS FirstIn ->       [1, 8] <- Lastin  Replaced:-  [Hits:0 Misses:2]
Access: 7  MISS FirstIn ->    [1, 8, 7] <- Lastin  Replaced:-  [Hits:0 Misses:3]
Access: 2  MISS FirstIn ->    [8, 7, 2] <- Lastin  Replaced:1  [Hits:0 Misses:4]
Access: 4  MISS FirstIn ->    [7, 2, 4] <- Lastin  Replaced:8  [Hits:0 Misses:5]
Access: 4  HIT  FirstIn ->    [7, 2, 4] <- Lastin  Replaced:-  [Hits:1 Misses:5]
Access: 6  MISS FirstIn ->    [2, 4, 6] <- Lastin  Replaced:7  [Hits:1 Misses:6]
Access: 7  MISS FirstIn ->    [4, 6, 7] <- Lastin  Replaced:2  [Hits:1 Misses:7]
Access: 0  MISS FirstIn ->    [6, 7, 0] <- Lastin  Replaced:4  [Hits:1 Misses:8]
Access: 0  HIT  FirstIn ->    [6, 7, 0] <- Lastin  Replaced:-  [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

FIFO when seed is 1.

Number of hits is 2, Number of misses is 8, therefore hit-rate is 20%.

```
017310367@swui:~$ ./paging-policy.py --policy=LRU -s 1 -n 10 -c
RG addresses -1
RG addressfile
RG numaddrs 10
RG policy LRU
RG clockbits 2
RG cachesize 3
RG maxpage 10
RG seed 1
RG notrace False

olving...

ccess: 1  MISS LRU ->          [1] <- MRU Replaced:-  [Hits:0 Misses:1]
ccess: 8  MISS LRU ->       [1, 8] <- MRU Replaced:-  [Hits:0 Misses:2]
ccess: 7  MISS LRU ->    [1, 8, 7] <- MRU Replaced:-  [Hits:0 Misses:3]
ccess: 2  MISS LRU ->    [8, 7, 2] <- MRU Replaced:1  [Hits:0 Misses:4]
ccess: 4  MISS LRU ->    [7, 2, 4] <- MRU Replaced:8  [Hits:0 Misses:5]
ccess: 4  HIT  LRU ->    [7, 2, 4] <- MRU Replaced:-  [Hits:1 Misses:5]
ccess: 6  MISS LRU ->    [2, 4, 6] <- MRU Replaced:7  [Hits:1 Misses:6]
ccess: 7  MISS LRU ->    [4, 6, 7] <- MRU Replaced:2  [Hits:1 Misses:7]
ccess: 0  MISS LRU ->    [6, 7, 0] <- MRU Replaced:4  [Hits:1 Misses:8]
ccess: 0  HIT  LRU ->    [6, 7, 0] <- MRU Replaced:-  [Hits:2 Misses:8]

INALSTATS hits 2   misses 8   hitrate 20.00
```

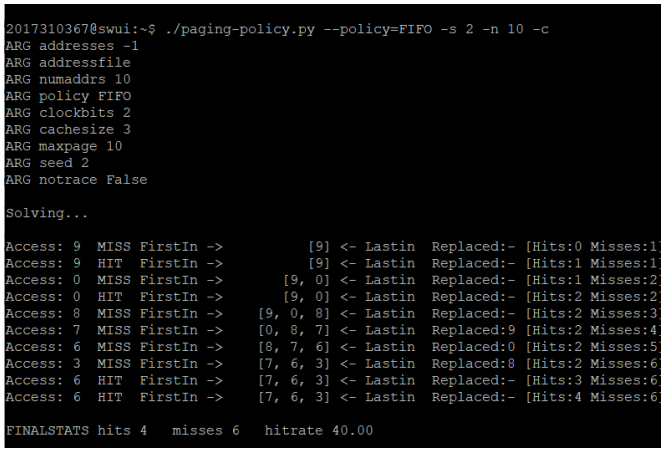
LRU when seed is 1.

Number of hits is 2, Number of misses is 8, therefore hit-rate is 20%.

```
2017310367@swui:~$ ./paging-policy.py --policy=OPT -s 1 -n 10 -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy OPT
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS Left  ->          [1] <- Right Replaced:-  [Hits:0 Misses:1]
Access: 8  MISS Left  ->       [1, 8] <- Right Replaced:-  [Hits:0 Misses:2]
Access: 7  MISS Left  ->    [1, 8, 7] <- Right Replaced:-  [Hits:0 Misses:3]
Access: 2  MISS Left  ->    [1, 7, 2] <- Right Replaced:8  [Hits:0 Misses:4]
Access: 4  MISS Left  ->    [1, 7, 4] <- Right Replaced:2  [Hits:0 Misses:5]
Access: 4  HIT  Left  ->    [1, 7, 4] <- Right Replaced:-  [Hits:1 Misses:5]
Access: 6  MISS Left  ->    [1, 7, 6] <- Right Replaced:4  [Hits:1 Misses:6]
Access: 7  HIT  Left  ->    [1, 7, 6] <- Right Replaced:-  [Hits:2 Misses:6]
Access: 0  MISS Left  ->    [1, 7, 0] <- Right Replaced:6  [Hits:2 Misses:7]
Access: 0  HIT  Left  ->    [1, 7, 0] <- Right Replaced:-  [Hits:3 Misses:7]

FINALSTATS hits 3   misses 7   hitrate 30.00
```

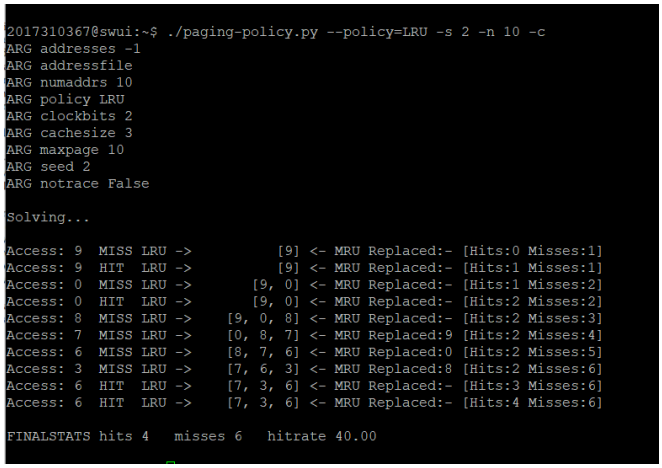
OPT when seed is 1.

Number of hits is 3, Number of misses is 7, therefore hit-rate is 30%.



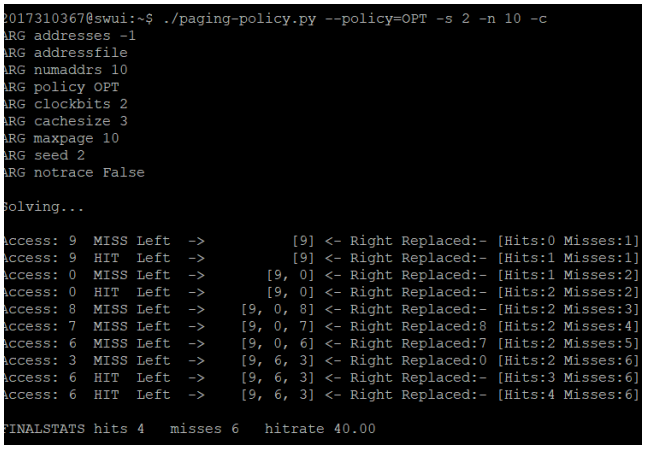
FIFO when seed is 2.

Number of hits is 4, Number of misses is 6, therefore hit-rate is 40%.



LRU when seed is 2.

Number of hits is 4, Number of misses is 6, therefore hit-rate is 40%.

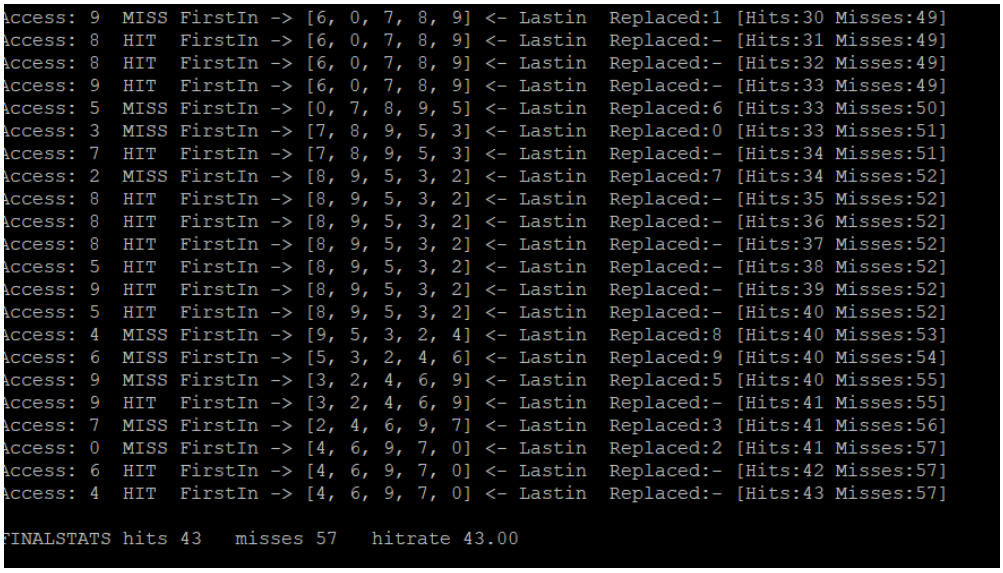


OPT when seed is 2.

Number of hits is 4, Number of misses is 6, therefore hit-rate is 40%.

Overall, OPT showed the best hit-rate, followed by LRU and FIFO.

2. For a cache of size 5, generate worst-case address reference streams for each of the following policies: FIFO, LRU, and MRU (worst-case reference streams cause the most misses possible. For the worst case reference streams, how much bigger of a cache is needed to improve performance dramatically and approach OPT?



When the cache size was 5 and number of streams is 100, FIFO hit-rate showed that

seed 0: 43%

seed 1: 48%

seed 2: 53%

The worst-case reference stream is seed 0.

```
ccess: 1  HIT  LRU -> [7, 8, 9, 6, 4, 3, 5, 2, 1] <- MRU Replaced:- [Hits:55 Misses:16]
ccess: 1  HIT  LRU -> [7, 8, 9, 6, 4, 3, 5, 2, 1] <- MRU Replaced:- [Hits:56 Misses:16]
ccess: 6  HIT  LRU -> [7, 8, 9, 4, 3, 5, 2, 1, 6] <- MRU Replaced:- [Hits:57 Misses:16]
ccess: 6  HIT  LRU -> [7, 8, 9, 4, 3, 5, 2, 1, 6] <- MRU Replaced:- [Hits:58 Misses:16]
ccess: 4  HIT  LRU -> [7, 8, 9, 3, 5, 2, 1, 6, 4] <- MRU Replaced:- [Hits:59 Misses:16]
ccess: 0  MISS LRU -> [8, 9, 3, 5, 2, 1, 6, 4, 0] <- MRU Replaced:7 [Hits:59 Misses:17]
ccess: 7  MISS LRU -> [9, 3, 5, 2, 1, 6, 4, 0, 7] <- MRU Replaced:8 [Hits:59 Misses:18]
ccess: 8  MISS LRU -> [3, 5, 2, 1, 6, 4, 0, 7, 8] <- MRU Replaced:9 [Hits:59 Misses:19]
ccess: 9  MISS LRU -> [5, 2, 1, 6, 4, 0, 7, 8, 9] <- MRU Replaced:3 [Hits:59 Misses:20]
ccess: 8  HIT  LRU -> [5, 2, 1, 6, 4, 0, 7, 9, 8] <- MRU Replaced:- [Hits:60 Misses:20]
ccess: 8  HIT  LRU -> [5, 2, 1, 6, 4, 0, 7, 9, 8] <- MRU Replaced:- [Hits:61 Misses:20]
ccess: 9  HIT  LRU -> [5, 2, 1, 6, 4, 0, 7, 8, 9] <- MRU Replaced:- [Hits:62 Misses:20]
ccess: 5  HIT  LRU -> [2, 1, 6, 4, 0, 7, 8, 9, 5] <- MRU Replaced:- [Hits:63 Misses:20]
ccess: 3  MISS LRU -> [1, 6, 4, 0, 7, 8, 9, 5, 3] <- MRU Replaced:2 [Hits:63 Misses:21]
ccess: 7  HIT  LRU -> [1, 6, 4, 0, 8, 9, 5, 3, 7] <- MRU Replaced:- [Hits:64 Misses:21]
ccess: 2  MISS LRU -> [6, 4, 0, 8, 9, 5, 3, 7, 2] <- MRU Replaced:1 [Hits:64 Misses:22]
ccess: 8  HIT  LRU -> [6, 4, 0, 9, 5, 3, 7, 2, 8] <- MRU Replaced:- [Hits:65 Misses:22]
ccess: 8  HIT  LRU -> [6, 4, 0, 9, 5, 3, 7, 2, 8] <- MRU Replaced:- [Hits:66 Misses:22]
ccess: 8  HIT  LRU -> [6, 4, 0, 9, 5, 3, 7, 2, 8] <- MRU Replaced:- [Hits:67 Misses:22]
ccess: 5  HIT  LRU -> [6, 4, 0, 9, 3, 7, 2, 8, 5] <- MRU Replaced:- [Hits:68 Misses:22]
ccess: 9  HIT  LRU -> [6, 4, 0, 3, 7, 2, 8, 5, 9] <- MRU Replaced:- [Hits:69 Misses:22]
ccess: 5  HIT  LRU -> [6, 4, 0, 3, 7, 2, 8, 9, 5] <- MRU Replaced:- [Hits:70 Misses:22]
ccess: 4  HIT  LRU -> [6, 0, 3, 7, 2, 8, 9, 5, 4] <- MRU Replaced:- [Hits:71 Misses:22]
ccess: 6  HIT  LRU -> [0, 3, 7, 2, 8, 9, 5, 4, 6] <- MRU Replaced:- [Hits:72 Misses:22]
ccess: 9  HIT  LRU -> [0, 3, 7, 2, 8, 5, 4, 6, 9] <- MRU Replaced:- [Hits:73 Misses:22]
ccess: 9  HIT  LRU -> [0, 3, 7, 2, 8, 5, 4, 6, 9] <- MRU Replaced:- [Hits:74 Misses:22]
ccess: 7  HIT  LRU -> [0, 3, 2, 8, 5, 4, 6, 9, 7] <- MRU Replaced:- [Hits:75 Misses:22]
ccess: 0  HIT  LRU -> [3, 2, 8, 5, 4, 6, 9, 7, 0] <- MRU Replaced:- [Hits:76 Misses:22]
ccess: 6  HIT  LRU -> [3, 2, 8, 5, 4, 9, 7, 0, 6] <- MRU Replaced:- [Hits:77 Misses:22]
ccess: 4  HIT  LRU -> [3, 2, 8, 5, 9, 7, 0, 6, 4] <- MRU Replaced:- [Hits:78 Misses:22]

INALSTATS hits 78   misses 22   hitrate 78.00
```

When the cache size was 5 and number of streams is 100, LRU hit-rate showed that

seed 0: 78%

seed 1: 83%

seed 2: 82%

The worst-case reference stream is seed 0.

```
ccess: 0  HIT  LRU -> [9, 1, 5, 6, 0] <- MRU Replaced:- [Hits:54 Misses:72]
ccess: 7  MISS LRU -> [9, 1, 5, 6, 7] <- MRU Replaced:0 [Hits:54 Misses:73]
ccess: 2  MISS LRU -> [9, 1, 5, 6, 2] <- MRU Replaced:7 [Hits:54 Misses:74]
ccess: 1  HIT  LRU -> [9, 5, 6, 2, 1] <- MRU Replaced:- [Hits:55 Misses:74]
ccess: 6  HIT  LRU -> [9, 5, 2, 1, 6] <- MRU Replaced:- [Hits:56 Misses:74]
ccess: 3  MISS LRU -> [9, 5, 2, 1, 3] <- MRU Replaced:6 [Hits:56 Misses:75]
ccess: 0  MISS LRU -> [9, 5, 2, 1, 0] <- MRU Replaced:3 [Hits:56 Misses:76]
ccess: 1  HIT  LRU -> [9, 5, 2, 0, 1] <- MRU Replaced:- [Hits:57 Misses:76]
ccess: 5  HIT  LRU -> [9, 2, 0, 1, 5] <- MRU Replaced:- [Hits:58 Misses:76]
ccess: 1  HIT  LRU -> [9, 2, 0, 5, 1] <- MRU Replaced:- [Hits:59 Misses:76]
ccess: 2  HIT  LRU -> [9, 0, 5, 1, 2] <- MRU Replaced:- [Hits:60 Misses:76]
ccess: 7  MISS LRU -> [9, 0, 5, 1, 7] <- MRU Replaced:2 [Hits:60 Misses:77]
ccess: 4  MISS LRU -> [9, 0, 5, 1, 4] <- MRU Replaced:7 [Hits:60 Misses:78]
ccess: 3  MISS LRU -> [9, 0, 5, 1, 3] <- MRU Replaced:4 [Hits:60 Misses:79]
ccess: 4  MISS LRU -> [9, 0, 5, 1, 4] <- MRU Replaced:3 [Hits:60 Misses:80]
ccess: 0  HIT  LRU -> [9, 5, 1, 4, 0] <- MRU Replaced:- [Hits:61 Misses:80]
ccess: 3  MISS LRU -> [9, 5, 1, 4, 3] <- MRU Replaced:0 [Hits:61 Misses:81]
ccess: 4  HIT  LRU -> [9, 5, 1, 3, 4] <- MRU Replaced:- [Hits:62 Misses:81]
ccess: 1  HIT  LRU -> [9, 5, 3, 4, 1] <- MRU Replaced:- [Hits:63 Misses:81]
ccess: 1  HIT  LRU -> [9, 5, 3, 4, 1] <- MRU Replaced:- [Hits:64 Misses:81]
ccess: 8  MISS LRU -> [9, 5, 3, 4, 8] <- MRU Replaced:1 [Hits:64 Misses:82]
ccess: 5  HIT  LRU -> [9, 3, 4, 8, 5] <- MRU Replaced:- [Hits:65 Misses:82]
ccess: 2  MISS LRU -> [9, 3, 4, 8, 2] <- MRU Replaced:5 [Hits:65 Misses:83]
ccess: 6  MISS LRU -> [9, 3, 4, 8, 6] <- MRU Replaced:2 [Hits:65 Misses:84]
ccess: 8  HIT  LRU -> [9, 3, 4, 6, 8] <- MRU Replaced:- [Hits:66 Misses:84]

INALSTATS hits 66   misses 84   hitrate 44.00
```

When the cache size was 5 and number of streams is 100, MRU hit-rate showed that

seed 0: 48%

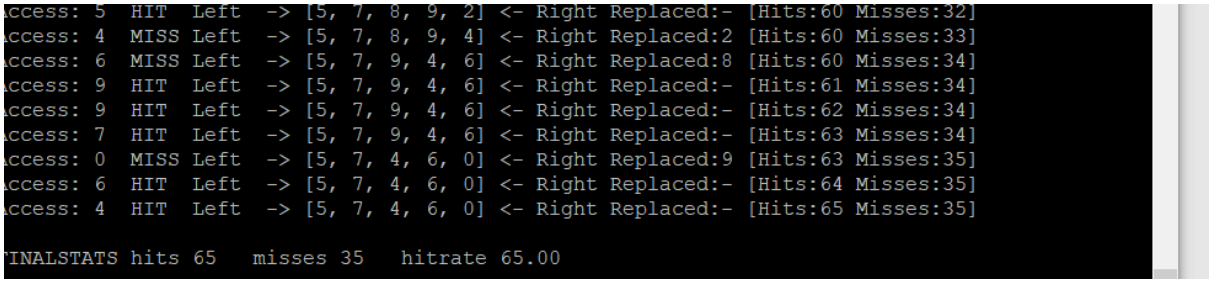
seed 1: 44%

seed 2: 48%

The worst-case reference stream is seed 1.

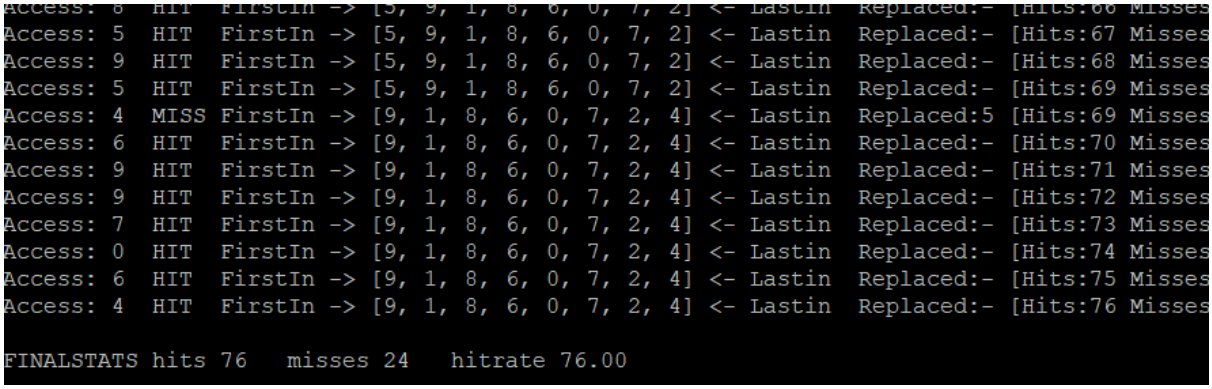
Therefore, the seed 0 is the worst-case reference stream of FIFO and LRU, and the seed 1 is the worst-case reference stream of MRU.

This is OPT of each worst-case reference stream



OPT hit-rate is 65% when seed is 0 (the worst-case reference stream of FIFO and LRU)

Case of FIFO



When the cache size was gradually raised, the hit-rate was 60 percent when the cache size was 7, and 76 percent when the cache size was 8. Considering OPT's hit-rate is 65%, cache size has to be 8 to improve performance dramatically and approach OPT.

Case of FIFO

Case of FIFO

Case of LRU

```
Access: 5  HIT  LRU -> [4, 0, 9, 3, 7, 2, 8, 5] <- MRU Replaced:- [Hits:61 Misses:29]
Access: 9  HIT  LRU -> [4, 0, 3, 7, 2, 8, 5, 9] <- MRU Replaced:- [Hits:62 Misses:29]
Access: 5  HIT  LRU -> [4, 0, 3, 7, 2, 8, 9, 5] <- MRU Replaced:- [Hits:63 Misses:29]
Access: 4  HIT  LRU -> [0, 3, 7, 2, 8, 9, 5, 4] <- MRU Replaced:- [Hits:64 Misses:29]
Access: 6  MISS LRU -> [3, 7, 2, 8, 9, 5, 4, 6] <- MRU Replaced:0 [Hits:64 Misses:30]
Access: 9  HIT  LRU -> [3, 7, 2, 8, 5, 4, 6, 9] <- MRU Replaced:- [Hits:65 Misses:30]
Access: 9  HIT  LRU -> [3, 7, 2, 8, 5, 4, 6, 9] <- MRU Replaced:- [Hits:66 Misses:30]
Access: 7  HIT  LRU -> [3, 2, 8, 5, 4, 6, 9, 7] <- MRU Replaced:- [Hits:67 Misses:30]
Access: 0  MISS LRU -> [2, 8, 5, 4, 6, 9, 7, 0] <- MRU Replaced:3 [Hits:67 Misses:31]
Access: 6  HIT  LRU -> [2, 8, 5, 4, 9, 7, 0, 6] <- MRU Replaced:- [Hits:68 Misses:31]
Access: 4  HIT  LRU -> [2, 8, 5, 9, 7, 0, 6, 4] <- MRU Replaced:- [Hits:69 Misses:31]

FINALSTATS hits 69   misses 31   hitrate 69.00
```

When the cache size was gradually raised, the hit-rate was 69 percent when the cache was 8. Considering OPT's hit-rate is 65%, cache size has to be 8 to improve performance dramatically and approach OPT.

```
ccess: 7  HIT  Left  -> [2, 7, 1, 3, 5] <- Right Replaced:- [Hits:100 Misses:37
ccess: 4  MISS Left  -> [2, 1, 3, 5, 4] <- Right Replaced:7 [Hits:100 Misses:38
ccess: 3  HIT  Left  -> [2, 1, 3, 5, 4] <- Right Replaced:- [Hits:101 Misses:38
ccess: 4  HIT  Left  -> [2, 1, 3, 5, 4] <- Right Replaced:- [Hits:102 Misses:38
ccess: 0  MISS Left  -> [1, 3, 5, 4, 0] <- Right Replaced:2 [Hits:102 Misses:39
ccess: 3  HIT  Left  -> [1, 3, 5, 4, 0] <- Right Replaced:- [Hits:103 Misses:39
ccess: 4  HIT  Left  -> [1, 3, 5, 4, 0] <- Right Replaced:- [Hits:104 Misses:39
ccess: 1  HIT  Left  -> [1, 3, 5, 4, 0] <- Right Replaced:- [Hits:105 Misses:39
ccess: 1  HIT  Left  -> [1, 3, 5, 4, 0] <- Right Replaced:- [Hits:106 Misses:39
ccess: 8  MISS Left  -> [1, 3, 5, 4, 8] <- Right Replaced:0 [Hits:106 Misses:40
ccess: 5  HIT  Left  -> [1, 3, 5, 4, 8] <- Right Replaced:- [Hits:107 Misses:40
ccess: 2  MISS Left  -> [1, 3, 5, 8, 2] <- Right Replaced:4 [Hits:107 Misses:41
ccess: 6  MISS Left  -> [1, 3, 5, 8, 6] <- Right Replaced:2 [Hits:107 Misses:42
ccess: 8  HIT  Left  -> [1, 3, 5, 8, 6] <- Right Replaced:- [Hits:108 Misses:42

INALSTATS hits 108   misses 42   hitrate 72.00
```

OPT hit-rate is 72% when seed is 1 (the worst-case reference stream of MRU)

Case of MRU

```
Access: 5  HIT  LRU -> [4, 0, 9, 3, 7, 2, 8, 5] <- MRU Replaced:- [Hits:61 Misses:29]
Access: 9  HIT  LRU -> [4, 0, 3, 7, 2, 8, 5, 9] <- MRU Replaced:- [Hits:62 Misses:29]
Access: 5  HIT  LRU -> [4, 0, 3, 7, 2, 8, 9, 5] <- MRU Replaced:- [Hits:63 Misses:29]
Access: 4  HIT  LRU -> [0, 3, 7, 2, 8, 9, 5, 4] <- MRU Replaced:- [Hits:64 Misses:29]
Access: 6  MISS LRU -> [3, 7, 2, 8, 9, 5, 4, 6] <- MRU Replaced:0 [Hits:64 Misses:30]
Access: 9  HIT  LRU -> [3, 7, 2, 8, 5, 4, 6, 9] <- MRU Replaced:- [Hits:65 Misses:30]
Access: 9  HIT  LRU -> [3, 7, 2, 8, 5, 4, 6, 9] <- MRU Replaced:- [Hits:66 Misses:30]
Access: 7  HIT  LRU -> [3, 2, 8, 5, 4, 6, 9, 7] <- MRU Replaced:- [Hits:67 Misses:30]
Access: 0  MISS LRU -> [2, 8, 5, 4, 6, 9, 7, 0] <- MRU Replaced:3 [Hits:67 Misses:31]
Access: 6  HIT  LRU -> [2, 8, 5, 4, 9, 7, 0, 6] <- MRU Replaced:- [Hits:68 Misses:31]
Access: 4  HIT  LRU -> [2, 8, 5, 9, 7, 0, 6, 4] <- MRU Replaced:- [Hits:69 Misses:31]

FINALSTATS hits 69   misses 31   hitrate 69.00
```

When the cache size was gradually raised, the hit-rate was 69 percent when the cache was 8. Considering OPT's hit-rate is 72%, cache size has to be 8~9 to improve performance dramatically and approach OPT

3. Generate a random trace (use python or perl). How would you expect the different policies to perform on such a trace?

4. Now generate a trace with some locality. How can you generate such a trace? How does LRU perform on it? How much better than RAND is LRU? How does CLOCK do? How about CLOCK with different numbers of clock bits?

5. Use a program like valgrind to instrument a real application and generate a virtual page reference stream. For example, running valgrind --tool=lackey --trace-mem=yes ls will output a nearly-complete reference trace of every instruction and data reference made by the program ls. To make this useful for the simulator above, you'll have to first transform each virtual memory reference into a virtual page-number reference (done by masking off the offset and shifting the resulting bits downward). How big of a cache is needed for your application trace in order to satisfy a large fraction of requests? Plot a graph of its working set as the size of the cache increases.