Project 2: Sample Runs

Sample Inputs

input/seq1.txt:

```
C D C E D C F C
```

input/seq2.txt:

```
A Q1 S1 S2
A Q1 S3 S4 S5
A Q1 T1 S6 S7 S8
A T1 S9 Q1 S10
A S11 S12 T1
A
```

Notes:

- For each input, you can run the simulator with different cache sizes and different replacement policies.
- Make sure to design more input sequences to test your program!
- Input file format to follow for the UI to work:
    - Addresses are represented by strings separated by whitespaces (space, tab, newline).
    - Addresses can be more than one character.

Sample Run 1

- Provide an input file to run the simulator.
- FIFO Cache, size 3, Normal mode.

```
>java Simulator input/seq1.txt
Select the cache to simulate:
 1-FIFO Cache; 2-LRU Cache; 3-LFU Cache.
 Option: 1
Select cache size to simulate (positive integer in [1,256]): 3
Simulating a FIFO Cache of size 3.
-----------------------------------------
Access 0: C - Miss
cache content after access:
C
cache not full, next to replace: null
-----------------------------------------
Access 1: D - Miss
cache content after access:
C D
cache not full, next to replace: null
-----------------------------------------
Access 2: C - Hit
cache content after access:
C D
cache not full, next to replace: null
-----------------------------------------
Access 3: E - Miss
cache content after access:
C D E
cache full, next to replace: C
-----------------------------------------
Access 4: D - Hit
cache content after access:
C D E
cache full, next to replace: C
-----------------------------------------
Access 5: C - Hit
cache content after access:
C D E
cache full, next to replace: C
-----------------------------------------
Access 6: F - Miss
cache content after access:
D E F
cache full, next to replace: D
-----------------------------------------
Access 7: C - Miss
cache content after access:
E F C
cache full, next to replace: E
-----------------------------------------
Access 8: A - Miss
cache content after access:
F C A
cache full, next to replace: F
-----------------------------------------
Hit Rate: 30.00%
```

Pick a cache replacement policy to apply.

Pick a cache size to simulate.

Report cache hit/miss for each access. Show cache contents after processing this access.

FIFO cache: items are displayed from the earliest arrival to the latest arrival.

When cache is full, next cache miss will trigger a cache replacement. Report which item will be evicted on next cache miss.

Updates triggered by this cache miss:
- F is loaded in as the latest arrival;
- C is evicted;
- D is now the earliest arrival and next candidate to be replaced.

Report hit rate when the whole sequence is completed.
- Hit Rate = num. of hits / num. of accesses

Sample Run 2

- LRU Cache, size 3, Detailed mode.

Use "-d" to start in detailed mode.

```
>java Simulator input/seq1.txt -d
Select the cache to simulate:
 1-FIFO Cache; 2-LRU Cache; 3-LFU Cache.
 Option: 2
Select cache size to simulate (positive integer in [1,256]): 3
Simulating a LRU Cache of size 3.
-----------------------------------------
Access 0: C - Miss
cache content after access:
C
cache not full, next to replace: null
-----------------------------------------
Access 1: D - Miss
cache content after access:
C D
cache not full, next to replace: null
-----------------------------------------
Access 2: C - Hit
cache content after access:
D C
cache not full, next to replace: null
-----------------------------------------
Access 3: E - Miss
cache content after access:
D C E
cache full, next to replace: D
-----------------------------------------
Access 4: D - Hit
cache content after access:
C E D
cache full, next to replace: C
-----------------------------------------
Access 5: C - Hit
cache content after access:
E D C
cache full, next to replace: E
-----------------------------------------
Access 6: F - Miss
cache content after access:
D C F
cache full, next to replace: D
-----------------------------------------
Access 7: C - Hit
cache content after access:
D F C
cache full, next to replace: D
-----------------------------------------
Access 8: A - Miss
cache content after access:
F C A
cache full, next to replace: F
-----------------------------------------
Hit Rate: 40.00%
Accesses: <F:1>,<A:1>,<C:4>,<D:2>,<E:1>
Hits: <C:3>,<D:1>
```

LRU cache: items are displayed from the least recently accessed to most recently accessed.

Cache content / block ordering updated although this is a cache hit:

- D is accessed earlier in time while C is the most recently accessed item.

LRU item is the one to be replaced on a cache miss when cache is full.

Detailed mode will report the number of accesses and number of hits for each address in the given sequence at the end of the simulation.

Note: the order of items printed out depends on the hash code of the string, the insertion order, and the length of the hash table buckets.

Sample Run 3

- LFU Cache, size 5, Detailed mode.

```
>java Simulator input/seq2.txt -d
Select the cache to simulate:
 1-FIFO Cache; 2-LRU Cache; 3-LFU Cache.
 Option: 3
Select cache size to simulate (positive integer in [1,256]): 5
Simulating an LFU Cache of size 5.
-----------------------------------------
Access 0: A - Miss
cache content after access:
<A,1>
cache not full, next to replace: null
-----------------------------------------
Access 1: Q1 - Miss
cache content after access:
<A,1> <Q1,1>
cache not full, next to replace: null
-----------------------------------------
Access 2: S1 - Miss
cache content after access:
<A,1> <Q1,1> <S1,1>
cache not full, next to replace: null
-----------------------------------------
Access 3: S2 - Miss
cache content after access:
<A,1> <Q1,1> <S1,1> <S2,1>
cache not full, next to replace: null
-----------------------------------------
Access 4: A - Hit
cache content after access:
<Q1,1> <S1,1> <S2,1> <A,2>
cache not full, next to replace: null
-----------------------------------------
Access 5: Q1 - Hit
cache content after access:
<S1,1> <S2,1> <A,2> <Q1,2>
cache not full, next to replace: null
-----------------------------------------
Access 6: S3 - Miss
cache content after access:
<S1,1> <S2,1> <S3,1> <A,2> <Q1,2>
cache full, next to replace: S1
-----------------------------------------
Access 7: S4 - Miss
cache content after access:
<S2,1> <S3,1> <S4,1> <A,2> <Q1,2>
cache full, next to replace: S2
-----------------------------------------
Access 8: S5 - Miss
cache content after access:
<S3,1> <S4,1> <S5,1> <A,2> <Q1,2>
cache full, next to replace: S3
-----------------------------------------
Access 9: A - Hit
cache content after access:
<S3,1> <S4,1> <S5,1> <Q1,2> <A,3>
cache full, next to replace: S3
-----------------------------------------
```

For LFU cache, we record the access count for each entry as <address, count>.

For LFU cache, blocks are displayed from the least frequently accessed to most frequently accessed, i.e. lowest access count to highest access count.

Blocks with the same access count value are displayed from LRU to MRU.

Updates triggered by this cache hit:
- The access count of A is incremented to 2.
- It has the highest access frequency and hence order of blocks updated as well.

Block with the lowest access count and that was accessed earliest in time is the one to be replaced.

Updates triggered by this cache miss:
- S4 is the new block accessed. It is added into cache with a count=1. It is accessed the latest for all blocks with a count=1 and hence displayed as the last one (after <S3,1>).
- Cache is full, S1 is evicted.

```
Access 10: Q1 - Hit
cache content after access:
<S3,1> <S4,1> <S5,1> <A,3> <Q1,3>
cache full, next to replace: S3
-----------------------------------------
Access 11: T1 - Miss
cache content after access:
<S4,1> <S5,1> <T1,1> <A,3> <Q1,3>
cache full, next to replace: S4
-----------------------------------------
Access 12: S6 - Miss
cache content after access:
<S5,1> <T1,1> <S6,1> <A,3> <Q1,3>
cache full, next to replace: S5
-----------------------------------------
Access 13: S7 - Miss
cache content after access:
<T1,1> <S6,1> <S7,1> <A,3> <Q1,3>
cache full, next to replace: T1
-----------------------------------------
Access 14: S8 - Miss
cache content after access:
<S6,1> <S7,1> <S8,1> <A,3> <Q1,3>
cache full, next to replace: S6
-----------------------------------------
Access 15: A - Hit
cache content after access:
<S6,1> <S7,1> <S8,1> <Q1,3> <A,4>
cache full, next to replace: S6
-----------------------------------------
Access 16: T1 - Miss
cache content after access:
<S7,1> <S8,1> <T1,1> <Q1,3> <A,4>
cache full, next to replace: S7
-----------------------------------------
Access 17: S9 - Miss
cache content after access:
<S8,1> <T1,1> <S9,1> <Q1,3> <A,4>
cache full, next to replace: S8
-----------------------------------------
Access 18: Q1 - Hit
cache content after access:
<S8,1> <T1,1> <S9,1> <A,4> <Q1,4>
cache full, next to replace: S8
-----------------------------------------
Access 19: S10 - Miss
cache content after access:
<T1,1> <S9,1> <S10,1> <A,4> <Q1,4>
cache full, next to replace: T1
-----------------------------------------
Access 20: A - Hit
cache content after access:
<T1,1> <S9,1> <S10,1> <Q1,4> <A,5>
cache full, next to replace: T1
-----------------------------------------
Access 21: S11 - Miss
cache content after access:
<S9,1> <S10,1> <S11,1> <Q1,4> <A,5>
cache full, next to replace: S9
-----------------------------------------
```

```
Access 22: S12 - Miss
cache content after access:
<S10,1> <S11,1> <S12,1> <Q1,4> <A,5>
cache full, next to replace: S10
-----------------------------------------
Access 23: T1 - Miss
cache content after access:
<S11,1> <S12,1> <T1,1> <Q1,4> <A,5>
cache full, next to replace: S11
-----------------------------------------
Access 24: A - Hit
cache content after access:
<S11,1> <S12,1> <T1,1> <Q1,4> <A,6>
cache full, next to replace: S11
-----------------------------------------
Hit Rate: 30.77%
Accesses: <S4:1> <T1:3>,<S5:1>,<A:6> <S6:1>,<S7:1>,<S1:1> <S8:1> <S10:1>,<Q1:4> <S2:1>
<S9:1> <S11:1>,<S3:1> <S12:1>
Hits: <A:5>,<Q1:3>
```

Note: the order of items printed out depends on the hash code of the string, the insertion order, and the length of the hash table buckets.

For example:

- "A", hashCode = 65, mod 7=2 ➔ mapped to hash table buckets[2]
- "Q1", hashCode = 49+81*31 = 2560, mod 7=5 ➔ mapped to hash table buckets[5]
- Hence info for "A" is displayed before info for "Q1".