OS Lab8 Page Replacement

1. Basic requirements (70)

   After discussion, this lab requires you to write five page replacement algorithms.

   1) FIFO: Using a FIFO list to maintain the element in cache. (**10**)
   2) Min: When page miss happens, you need to replace the one which we won't use for longest time. (15)
   3) LRU: When page miss happens, you need to replace the one which is least recent used. (15)
   4) Clock: Using circular linked list to simulate a clock. The hand is always point at the next position of last replacement. When we check whether the element in the list, we don't move the hand. And during the check procedure, the valid bit does not change. But if the element in the list, we finally change its valid bit to 1. When miss occurs, we start from the hand, and replace the first element whose valid bit is 0. During this procedure, remember to set the valid bit to 0. And after replacement, don't forget to increase the hand. (30) See the following example:

      Consider this input: cache size = 3, n = 7, 1 3 2 4 3 2 1

      Initially, all valid bits are 0. We denoted X(y_z) X is position of the list, y is the page index, z is valid bit.

      1 comes, and doesn't in list. We add 1. Now list: 1(1_1),2(0_0),3(0_0) now the hand is point at the next position of 1, which is 2.

      3 comes, and doesn't in list. We add 3. Now list: 1(1_1), 2(3_1), 3(0_0) now the hand is point at the next position of 2, which is 3.

      2 comes, and doesn't in list. We add 2. Now list: 1(1_1), 2(3_1), 3(2_1) now the hand is point at the next position of 3, which is 1.

      4 comes, and doesn't in list. We start from the hand, and finally replace 1. Now list: 1(4_1), 2(3_0), 3(2_0) now the hand is point at the next position of 1, which is 2.

      3 comes, and in the list. We only change the valid bit of 3. Now list: 1(4_1), 2(3_1), 3(2_0). Now the hand is still at position 2.

      2 comes, and in the list. We only change the valid bit of 2. Now list: 1(4_1), 2(3_1), 3(2, 1). Now the hand is still at position 2.

      1 comes, and doesn't in the list, we start from the hand(2), and finally replace 2. Now list: 1(4_0), 2(1_1), 3(2, 0).

      Hit ratio: 2 / 7 = 28.57%

The allowed languages are C/C++. The compile command is: g++ -std=c++11.
We will give you three test cases: 1.in(10000 pages), 2.in(100000 pages), 3.in(100000 pages)

Explanation for input test cases:

The first line will be an integer K, which is the cache size.
The second line will be an integer A, which is the algorithm index.
The third line will be an integer N, which is the number of pages.
Then there will be N integers, represent the index of the query pages in order.

Please set the algorithm as the following format:

0-FIFO, 1-LRU, 2-Min, 3-Clock

Finally, your code will run on system test cases. (E.g. N = 1000000 and not generate randomly)

You should output the hit ratio as "Hit ratio = xx.xx%" (without quotes) at last. Please remember, your result should be rounded up to 2 the decimal places. For example, 10.25% 9.99%.

2. Bonus (20)

If you implement above algorithm correctly, and have good efficiency, you will get 20 bonus mark. Efficiency is judged by your running time.

2. Lab report(30)
   a) Write the report carefully and concretely. (20)
   b) We will give you three test cases. If you can complete the form correctly, you will get 10 points.