Memory Management

locked

by *IEEEXtreme*

Dr. X develops various highly optimized code for applications ranging from smart watches to autonomous vehicles using Wintel Operating Systems and Processors. For its next generation of Operating Systems, Wintel decided to change the page replacement algorithm from First-In-First-Out (FIFO) to Least Recently Used (LRU) in order to improve the performance. In an OS, page replacement algorithms decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated.

In a FIFO page replacement, the OS keeps track of all the pages in memory in a queue, with the most recent arrival at the back, and the oldest arrival in front. When a page needs to be replaced, the page at the front of the queue (the oldest page) is selected.

In a LRU page replacement, the OS maintains a list containing all the pages in memory. At the back of this list is the least recently used page and at the front is the most recently used page. Whenever a new page is accessed, the OS checks if the page is present in the list or not. If the page is listed, the page is moved to the front of the list. If the page is not listed, then the OS would evict the least recently used page from the list and add the new page to the front of the list.

Dr. X is excited by the prospect of improved performance for LRU page replacement so that he can advertise the improved performance of his applications. But Dr. X is aware of the fact that not all of his applications will get a performance boost because of the change. So he has hired you to develop a program to determine if an application can be advertised or not. Your program will monitor the various data accesses by the application and should determine the number of times that the pages will have to be replaced. If the number of page replacements with the LRU page replacement algorithm is less than with the FIFO page replacement algorithm, then Dr. X will be able to advertise the application.

Input Format

The input begins with an integer t, $1 \le t \le 20$, which gives the number of test cases in the input.

Each test case begins with a line containing three space-separated integers p, s and n, where p is the number of pages present in the OS for a program, s is the size of each page and n is the number of memory accesses by the application.

The next n lines will contain the various addresses accessed by the application.

Note: In order to solve this challenge, you will need to identify the page that corresponds to the addresses listed. The page number is given by the formula: floor([address]/s).

Constraints

p and s will always be a power of 2

$1 \le p \le 128$, $128 \le s \le 4096$, $1 \le n \le 600$

The memory addresses range from 0 to $2^{31}-1$, inclusive

Output Format

For each test case, you should output a single line with three space-separated values:

- The first value should be either `yes` or `no`: If the application can be advertised, then the output should be `yes`. If not, the output should be `no`.
- The second value should be the number of page replacements under the FIFO page replacement approach.
- The last value should be the number of page replacements under the LRU page replacement approach.

Sample Input

```
2
4 1024 5
0
1024
2048
3076
4096
2 128 7
```

```
0
255
127
256
60
1024
120
```

Sample Output

```
no 1 1
yes 3 2
```

Explanation

First Test case

For the first test case, there are only four pages which can be allocated and the size of each page is 1024 bytes. The addresses 0, 1024, 2048, 3076 and 4096 correspond to the 0th, 1st, 2nd, 3rd and 4th page.

The page replacements are shown with asterisks (*) in the following tables.

FIFO Page Replacement:

Cycle	0	1	2	3	4	5
Page Requested	-	0	1	2	3	4
OS Page 0	-	0	0	0	0	4*
OS Page 1	-	-	1	1	1	1
OS Page 2	-	-	-	2	2	2
OS Page 3	-	-	-	-	3	3

LRU Page Replacement:

Cycle	0	1	2	3	4	5
Page Requested	-	0	1	2	3	4
OS Page 0	-	0	0	0	0	4*
OS Page 1	-	-	1	1	1	1
OS Page 2	-	-	-	2	2	2
OS Page 3	-	-	-	-	3	3

Both LRU and FIFO page replacement algorithm have 1 page replacement. Hence, there will be no performance improvement, and Dr. X cannot advertise this application.

Second Test case

For the second test case, there are only two pages which can be allocated and the size of each page is 128 bytes. The addresses 0, 255, 127, 256, 60, 1024 and 120 correspond to the 0th, 1st, 0th, 2nd, 0th, 8th and 0th page. (Note that the floor(127/128), floor(60/128), and floor(120/128) are all equal to 0. Similarly, the floor(255/128) = 1).

The page replacements are shown with asterisks (*) in the following tables.

FIFO Page Replacement:

Cycle	0	1	2	3	4	5	6	7
New Data	-	0	1	0	2	0	8	0
OS Page 0	-	0	0	0	2*	2	8*	8
OS Page 1	-	-	1	1	1	0*	0	0

LRU Page Replacement:

Cycle	0	1	2	3	4	5	6	7
New Data	-	0	1	0	2	0	8	0
OS Page 0	-	0	0	0	0	0	0	0
OS Page 1	-	-	1	1	2*	2	8*	8

In case of FIFO page replacement, there are 3 page replacements. In case of LRU page replacement, there are 2 page replacements. Hence, there will be improved performance in this case, and Dr. X can advertise this application.

Max Score: 49pts dynamic

Submissions: 1037

Max Score: 49

Difficulty: Hard

More

Current Buffer (saved locally, editable)

Java 8

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) {
7         /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
8     }
9 }
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Scoring | Environment | FAQ | Terms Of Service | Privacy Policy