

**CSE223: Operating Systems**

**Project (2)**

**Submitted to:**

Dr. Gamal Abdel-Shafy

**Submitted by:**

**Name:** **ID:**

Ayman Hesham Mohamed16P3037

Gina Emil Attia 16P3022

Rowan Hazem Wagieh16P3023

**Brief Description:**

This program implements "page replacement algorithms" which are: First in First Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), Second-chance, enhance Second-chance and optimal algorithms.

In the First, the program generates a random page-reference array where page numbers range from 0 to 99. And then the program applies the random page-reference array to each algorithm and count the number of page faults incurred by each algorithm. And print the answer of those algorithms at the end.

Assuming that the memory is divided into a constant number of frames which is 20 frames.

Briefly, the FIFO algorithm assume that the first that enters the memory is the best one to replace, so after the memory is completely field we go to the first one entered the memory and replace it. For the LRU, the number which hasn’t been used for a long time is the best one to replace so we count how much we used each element and save it and when the memory is completely filled we replace the number with smallest count. But in the LFU, we need to save the index, counter of each number. As for the second chance, every number is associated to a reference bit which is set to 0 or 1, if the bit is 0 so this number is the best one to replace but if it is 1 so we change is to 0 but don’t replace if at this iteration. In the enhanced second chance, every number is associated with 2 bits: “00” or “01” or “10” or “11” but initially the reference bit is set to 1 and the modified bit is set randomly, if the number entered is repeated, no change happens. otherwise we search for "00" because it is the best page to be replaced but if we didn't find it, we search for "01" and changing the reference bit of all numbers we pass by to 0 but if we didn’t find “00” or “01” we modify the reference bit of all numbers in the memory to 0 and search again for “00” or “01” if we couldn’t find “00”. By default, the first iteration there is no "00" but at the second one we would find. Once we found "00" and replace it with the current number, we will continue from this number not from the beginning.

**Assumptions:**

1. Assuming that the memory values is initially set to 100 for all algorithms except for the enhanced second chance.
2. The memory is divided into 20 page frames.
3. Assuming that in the optimal algorithm, after the memory is filled; when a number will occur in an index that is greater than the current one, the index will be updated with it. So, the number that have the least index will be replaced and will not be found in the future.

**Test Cases:**

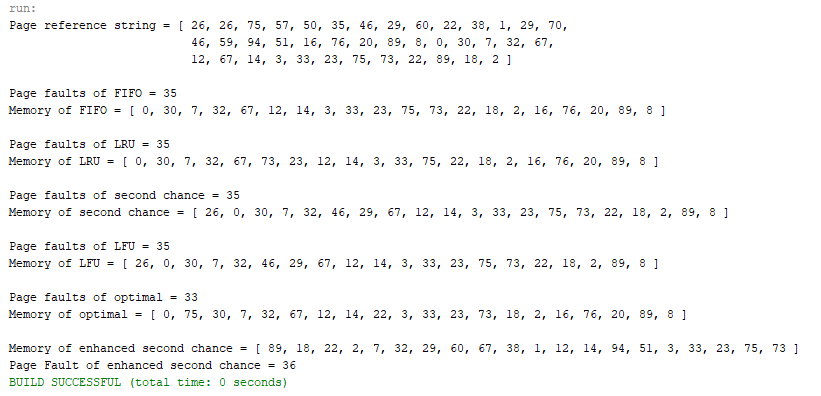
Test case 1 (The size of page reference string is 30)



Comment:

If the user enters 30 as a total size of the pages reference, which is a moderate number comparing to the number of page frames. We can observe the page faults is equal to 25 in all algorithms which is normal according to the size of page reference.

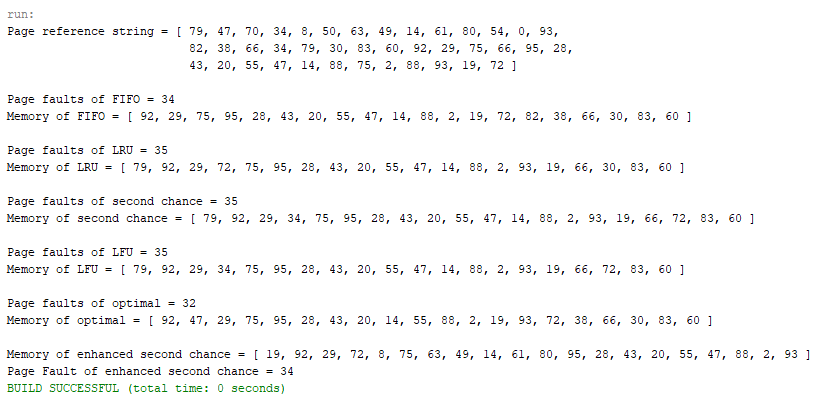
Test case 2 (The size of page reference string is 40)



Comment:

If the user enters 40 as a total size of the pages reference, which is double the size of memory. We can observe the page faults using the optimal algorithm is equal to 33 which is the smallest value comparing to the other algorithms.

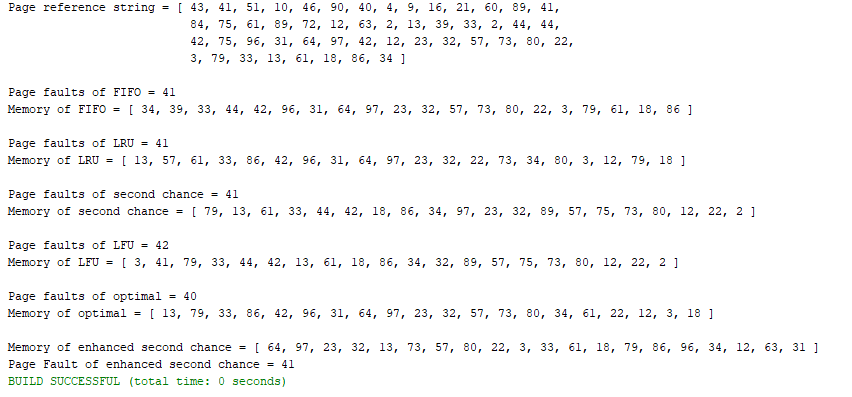
Test case 3 (The size of page reference string is 40)



Comment:

Same size of page reference string as the last test case but, here we can notice the variation of page faults between each algorithm and the other one, but still the optimal algorithm has the smallest page faults.

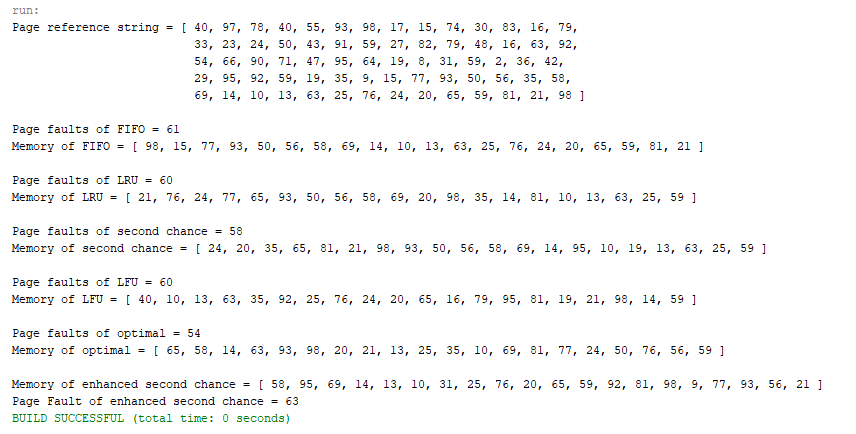
Test case 4 (The size of page reference string is 50)



Comment:

If the user enters 50 as the size of page reference string which is greater than double the size of memory page frames which is 20. We can notice that as long as the size of page references increase the number of page faults increase.

Test case 5 (The size of reference string is 70)



Comment:

If the user enters 80 as a total size of the pages reference, which is relatively a great number. We can observe that the page faults using the optimal algorithm equals 54 which is still the smallest value comparing to the other algorithms.