\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_OPERATING\_\_SYSTE\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*

NAME: NAVEEN SAI

ROLL NO: 33

SECTION:K17KV

GROUP: 1

\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PROJECT\_\_ \_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*

QUESTION :

Write a program that implements the least frequently used page replacement algorithm. First, generate

a random page-reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithm so that

the number of page frames can vary from 1 to 7. Assume that demand paging is used .

INTRODUCTION TO THE PAGE REPLACEMENTS:

 A program needs its code and information live in memory before it is executed. Yet, once in a while the memory estimate can't fit all the code and information of running projects. The arrangement is to cut both memory and projects into equivalent measured pages. An OS can undoubtedly swap in the required page of a program from plate to memory. Paging happens when a page blame happens. A page blame happens when a required page isn't inhabitant in memory and should be swapped in, conceivably overwriting another page in memory. A page substitution calculation chooses which memory page must be paged out (swap out, write to circle) to account for the mentioned page. To plan a decent page substitution calculation, we would prefer not to every now and again swap a similar memory page in and out. So to assess a page substitution calculation, we can run it on a specific string of memory references and decide the quantity of page blames that happen, the less the better.

Understanding Page Replacement Algorithms :

The page substitution calculations help a working framework in choosing the memory pages that should be swapped out, kept in touch with the plate when a page of memory should be apportioned in the framework.

The LRU Page Replacement strategy is a checking calculation. It monitors the page utilization in a given timeframe. The LRU calculation offers The LRU calculation offers ideal execution however is expensive in its usage. The LRU page substitution strategy is adjusted for usage, and its successors are The LRU page swap procedure is altered for usage, and its successors are LRU – K and ARC calculations.

DESCRIPITION: Page number: the page in the disk which will be demanded to swap in the memory.

Page frame: the page in the computer memory page-reference string: a sequence of the pages on the disk that are demanded to swap into the memory.

\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_C\*O\*D\*E\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*

#include<stdio.h>

int main()

{

int frames[10], temp[10], pages[10];

int total\_pages, m, n, position, k, l, total\_frames;

int a = 0, b = 0, page\_fault = 0;

printf("\nEnter Total Number of Frames:\t");

scanf("%d", &total\_frames);

for(m = 0; m < total\_frames; m++)

{

frames[m] = -1;

}

printf("Enter Total Number of Pages:\t");

scanf("%d", &total\_pages);

printf("Enter Values for Reference String:\n");

for(m = 0; m < total\_pages; m++)

{

printf("Value No.[%d]:\t", m + 1);

scanf("%d", &pages[m]);

}

for(n = 0; n < total\_pages; n++)

{

a = 0, b = 0;

for(m = 0; m < total\_frames; m++)

{

if(frames[m] == pages[n])

{

a = 1;

b = 1;

break;

}

}

if(a == 0)

{

for(m = 0; m < total\_frames; m++)

{

if(frames[m] == -1)

{

frames[m] = pages[n];

b = 1;

break;

}

}

}

if(b == 0)

{

for(m = 0; m < total\_frames; m++)

{

temp[m] = 0;

}

for(k = n - 1, l = 1; l <= total\_frames - 1; l++, k--)

{

for(m = 0; m < total\_frames; m++)

{

if(frames[m] == pages[k])

{

temp[m] = 1;

}

}

}

for(m = 0; m < total\_frames; m++)

{

if(temp[m] == 0)

position = m;

}

frames[position] = pages[n];

page\_fault++;

}

printf("\n");

for(m = 0; m < total\_frames; m++)

{

printf("%d\t", frames[m]);

}

}

printf("\nTotal Number of Page Faults:\t%d\n", page\_fault);

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_OUTPUT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ENTER TOTAL NUMBER OF FRAMES :4

ENTER THE TOTAL NUMBER OF PAGES :5

ENTER VALUES NUMBER OF PAGES :

VALUE NO 1 :5

VALUE NO 2 :3

VALUE NO 3 :1

VALUE NO 4 :2

VALUE NO 5 :4

5 -1 -1 -1

5 3 -1 -1

5 3 1 -1

5 3 1 2

4 3 1 2

\*TOTAL NUMBER OF PAGES FAULTS :1

The test pilot will create an irregular page-reference string and consider both the two calculation works, every one of which will print out its number of page deficiencies. The page outlines are spoken to by a progressively dispensed whole number exhibit; the variable lementCount will follow which is the nextpage edge to be supplanted .

In every calculation, it will mimic the paging procedure to take in the page-reference string and after that choose which page in the page casings will be paged out each time.

\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_END\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*