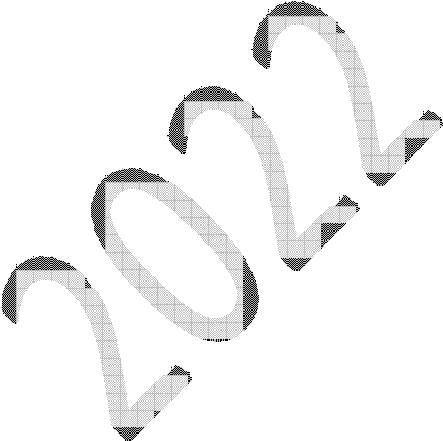
# Table of Contents

**USCSP301-USCS303: Operating System(OS) Practical-08**

Practical-08: Page Replacement Algorithm FIFO Practical Date: 30th Aug, 2021



Practical Aim: Page Replacement Algorithm FIFO Algorithm 3

Solved Example 5

[Question 6](#_TOC_250001)

Implementation 6

[Input 9](#_TOC_250000)

Output 9

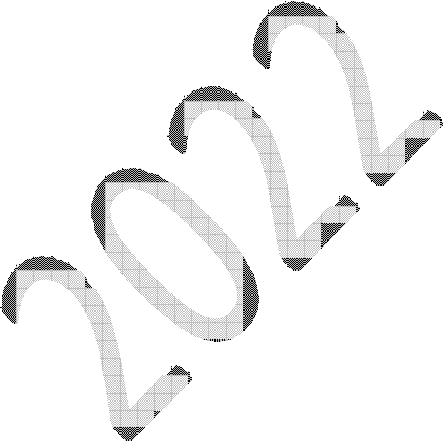
Sample Output 9

# Page Replacement Algorithm: FIFO

## Content:

* + In FIFO page replacement algorithm, the oldest page, which has spent the longest time in memory is chosen and replaced.

## Process:



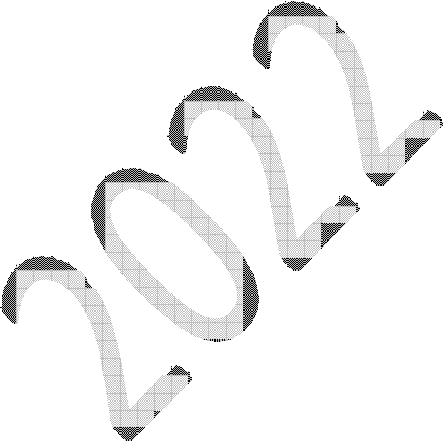
* + Implement FIFO Algorithm and find out page hits and page faults.

## Prior Knowledge:

* + Page Replacement Algorithm.

**Page Replacement Algorithm**

* In operating systems that use paging for memory management, **page replacement algorithm** are needed to decide which page needed to be replaced when new page comes in.
* Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page.



* Different page replacement algorithms suggest different ways to decide which page to replace.
* The target for all algorithms is to reduce number of page faults.
* **Page Fault** – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded inn physical memory.

**Step 1:** First of all, find the location of the desired page on the disk.

**Step 2:** Find a free Frame:

**Step 2.1:** If there is a free Frame, then use it.

**Step 2.2:** If there is no free frame then make use of the page replacement algorithm in order to select the victim frame.

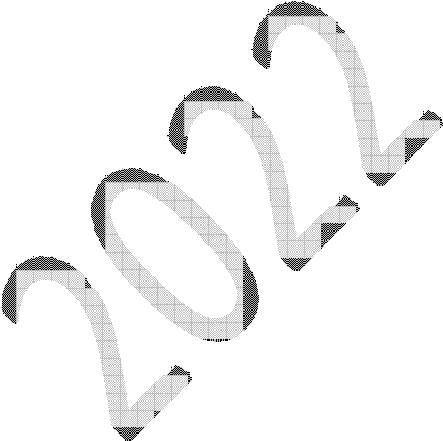
**Step 2.3:** Then after that write the victim frame to the disk and then make the changes in the page table and frame table accordingly.

**Step 3:** After that read the desired page into the newly freed frame and then change the page and frame tables.

**Step 4:** Restart the process.

# First In First Out (FIFO)

* It is very simple way of Page replacement and is referred to as **First In First Out (FIFO).**
* This algorithm mainly replaces the oldest page that has been present in the main memory for the longest time.
* This algorithm is implemented by keeping the track of all the pages in the queue.

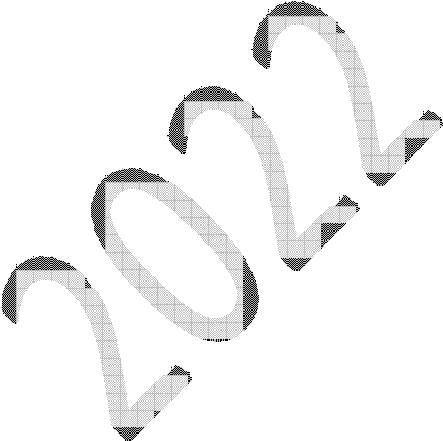


* As new pages are requested and are swapped in, they are added to the tail of a queue and the page which is at the head becomes the victim.
* This is not an effective way of page replacement but it can be used for small systems.

**EXAMPLE**

# FIFO Page Replacement Example

* Apply the FIFO replacement algorithm for the following page-reference strings: 0,2,1,6,4,0,1,0,3,1,2,1.
* Indicate the number of page faults for FIFO algorithm assuming demand paging with four frames.



* Find the number of hits, number of faults and hit ratio.

**Page Reference String:** 0,2,1,6,4,0,1,0,3,1,2,1

**Demand Paging or Number of Frames:** 4

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 2 |
| -1 | 2 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 |
| -1 | -1 | -1 | 6 | 6 | 6 | 6 | 6 | 6 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2 | 1 | 6 | 4 | 0 | 1 | 0 | 3 | 1 | 2 | 1 |

**Number of Hits:** count of no replacements = 3

**Number of Faults:** count of replacements = 9

**Hit Ratio:** Number of Hits / Len(Ref String) = 3/12 = 0.25

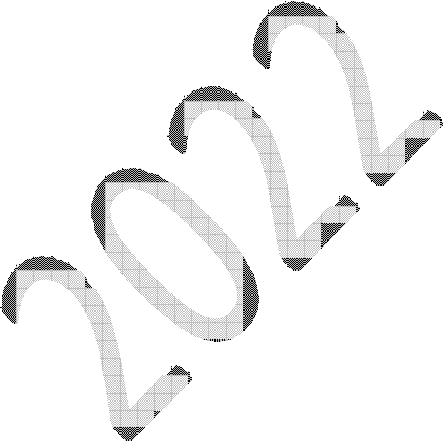
### Question:

Write a Java Program that implements the FIFO page-replacement algorithm.

### Source Code:

//NAME: Sumit Telawane

//BATCH: B1



//PRN: 2020016400825777

//DATE: 30th Aug, 2021

//PRAC-08: PAGE REPLACEMENT ALGORITHM

import java.io.\*; import java.util.\*;

public class P8\_PR\_FIFO\_ST

{

public static void main(String[] args) throws IOException

{

Scanner scan = new Scanner(System.in);

int frames, pointer = 0, hit =0, fault = 0, ref\_len; int buffer[];

int reference[];

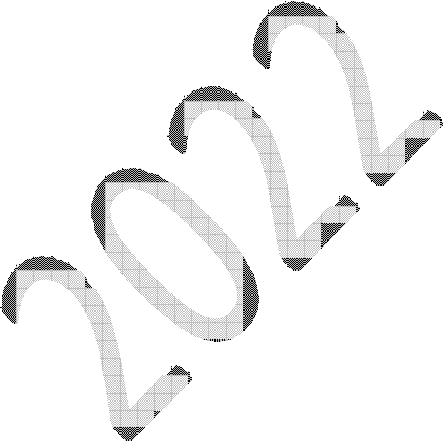
int mem\_layout[][];

System.out.print("Please enter the number of Frames: "); frames = scan.nextInt();

System.out.print("Please enter the length of the Reference string: "); ref\_len = scan.nextInt();

reference = new int[ref\_len]; mem\_layout = new int[ref\_len][frames]; buffer = new int[frames];

for(int j = 0; j<frames; j++) buffer[j] = -1;



System.out.println("Please enter the reference string: "); for(int i=0; i<ref\_len; i++)

{

reference[i] = scan.nextInt();

}

System.out.println(); for(int i=0; i< ref\_len; i++)

{

int search =-1;

for(int j=0; j<frames; j++)

{

if(buffer[j] ==reference[i])

{

search = j; hit++; break;

}

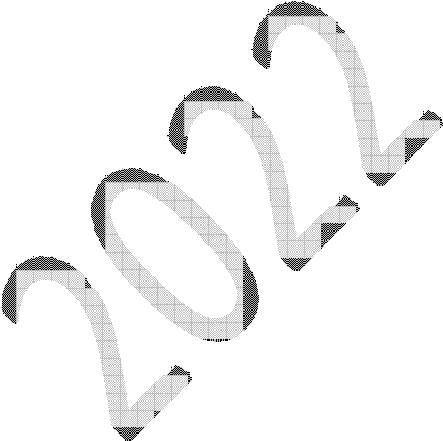
}

if (search==-1)

{

buffer[pointer]= reference[i]; fault++;

pointer++; if(pointer==frames) pointer = 0;



}

for(int j=0; j<frames; j++) mem\_layout[i][j]=buffer[j];

}

for(int i=0; i<frames; i++)

{

for(int j =0; j<ref\_len; j++) System.out.printf("%3d" , mem\_layout[j][i]); System.out.println();

}

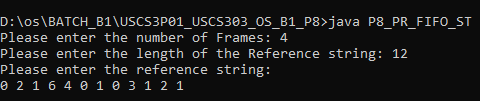
System.out.println("The number of Hits: "+hit); System.out.println("Hit Ratio: " +(float)((float)hit/ref\_len)); System.out.println("The number of Faults: "+fault);

}

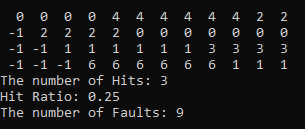
}



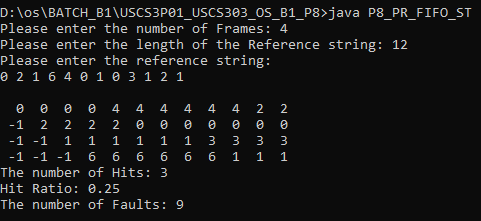
### Input:



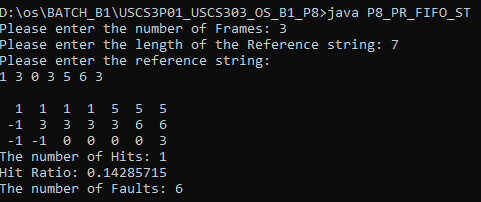
**Output:**



**Sample Output – 01**



**Sample Output – 02**



**Sample Output – 03:**

