USCSP301 : USCS303 - Operating System (OS)

Contents

[USCS3P01:USCS303 – Operating System (OS) Practical\_08 2](#_Toc81347669)

[Practical Date: 31st August 2021 2](#_Toc81347670)

[Practical Aim: Page Replacement Algorithm : FIFO 2](#_Toc81347671)

[Page Replacement Algorithm 2](#_Toc81347672)

[Solved Example: 4](#_Toc81347673)

[Do it yourself: 01 5](#_Toc81347674)

[Do it yourself: 02 5](#_Toc81347675)

[Question: 5](#_Toc81347676)

[Implementation: 5](#_Toc81347677)

[Input: 7](#_Toc81347678)

[Output: 7](#_Toc81347679)

[Sample Output01: 8](#_Toc81347680)

[Input: 8](#_Toc81347681)

[Output: 8](#_Toc81347682)

[Sample Output 02: 9](#_Toc81347683)

[Input: 9](#_Toc81347684)

[Output: 9](#_Toc81347685)

[Sample Output 03: 10](#_Toc81347686)

### **USCS3P01:USCS303 – Operating System (OS) Practical\_08**

### **Practical Date: 31st August 2021**

### **Practical Aim: Page Replacement Algorithm : FIFO**

**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 is 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 pages.
* 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 in 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 a 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 victim.
* This is not an effective way of page replacement but it can be used for small systems.

### **Solved Example:**

* Apply the FIFO replacement algorithms 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 you 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 |
| W | W | W | W | W | W | R | R | W | W | W | R |

Number of Hits: count of number replacements = 3

Number of faults: count of replacements = 9 [R]

Hit Ratio : Number of hits / len( Ref string)= 3/12 = 0.25 [W]

### **Do it yourself: 01**

* Consider the following example 3 frames with 1 ,3 ,0 ,3 ,5 ,6 ,3 page-reference strings.
* Find the number of hits, number of faults and hit ratio using page using FIFO Page Replacement Algorithm.

### **Do it yourself: 02**

* Consider the following example 3 frames with 7 ,0 ,1 ,2 ,0 ,3 ,0 ,4 ,2 ,3 ,0 ,3 ,2 ,1 ,2 ,0 ,1 ,7 ,0 ,1 page-reference strings.
* Find the number of hits, number of faults and hit ratio using page using LRU Page Replacement Algorithm

### **Question:**

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

### **Implementation:**

// Name: Gaurang Sanyasi

// Batch: B2

// PRN: 2020016400785461

// Date: 31st August, 2021

// Prac-08: Page Replacement Algorithm FIFO

import java.io.\*;

import java.util.\*;

public class P8\_PR\_FIFO\_GS

{

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 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);

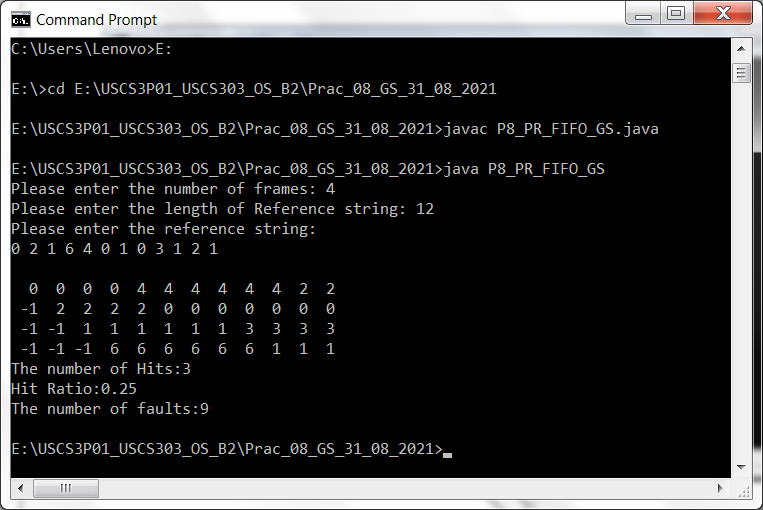
System.out.println("Hit Ratio:"+(float)((float)hit/ref\_len));

System.out.println("The number of faults:"+fault);

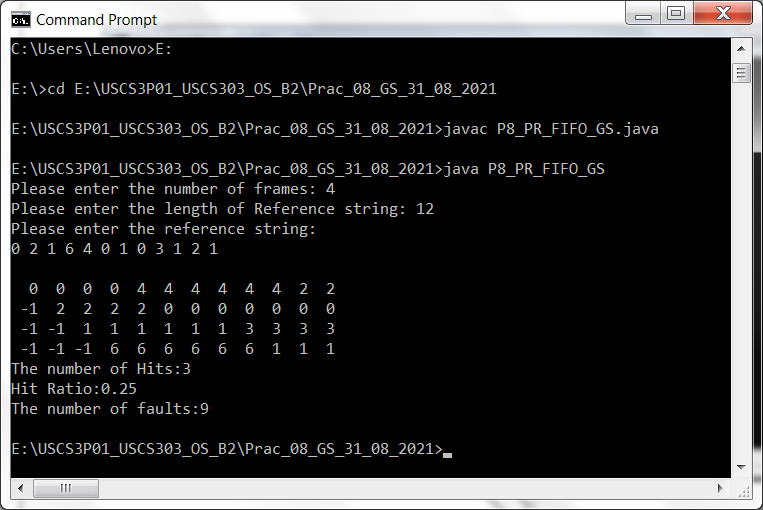
}

}

### **Input:**

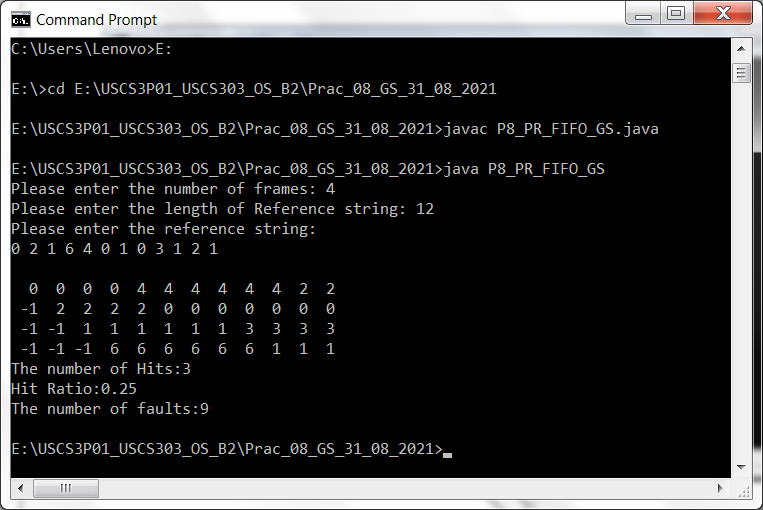


### **Output:**

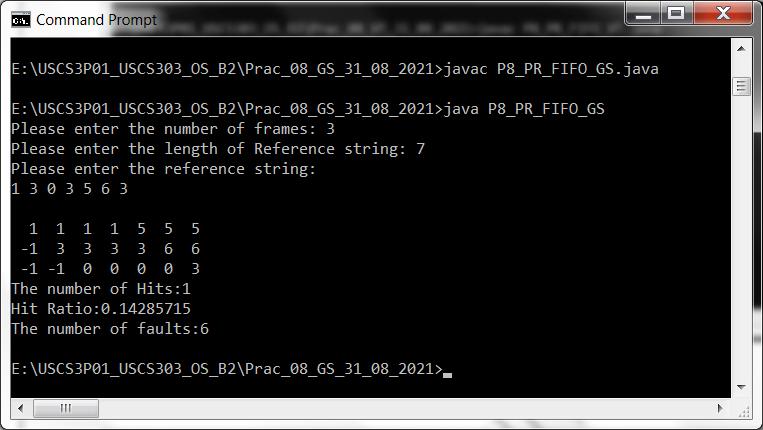
****

### 

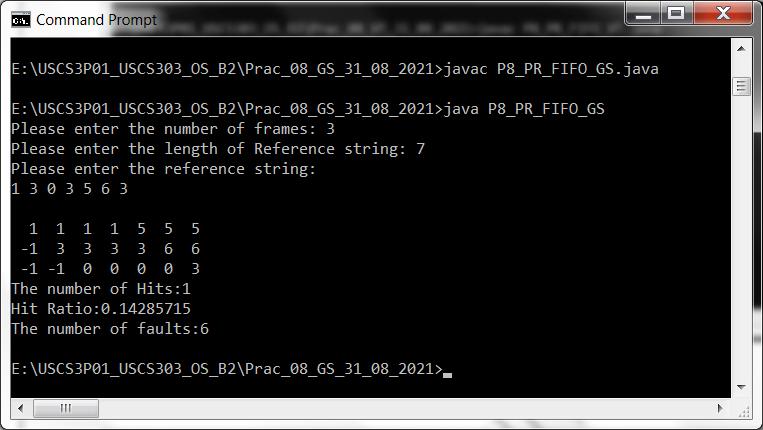
### **Sample Output01:**



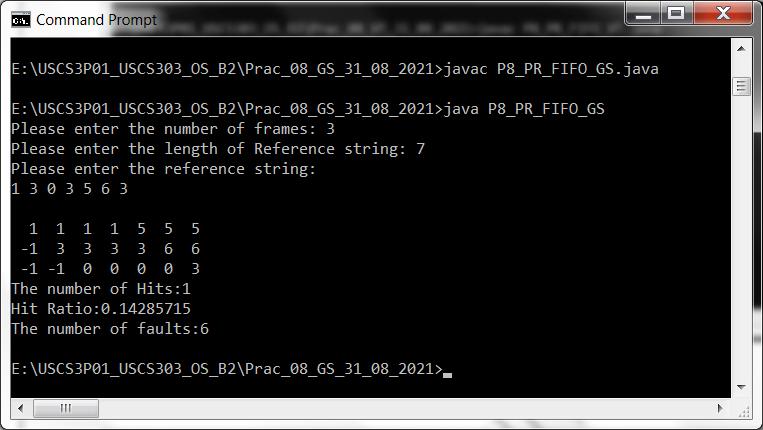
### **Input:**



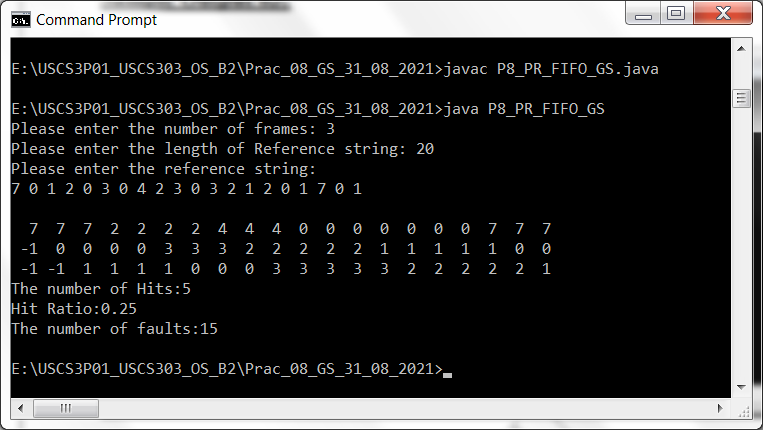
### **Output:**

****

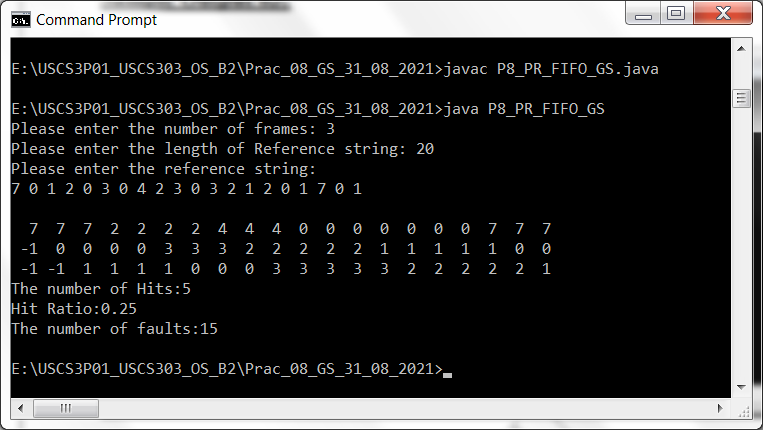
### **Sample Output 02:**

****

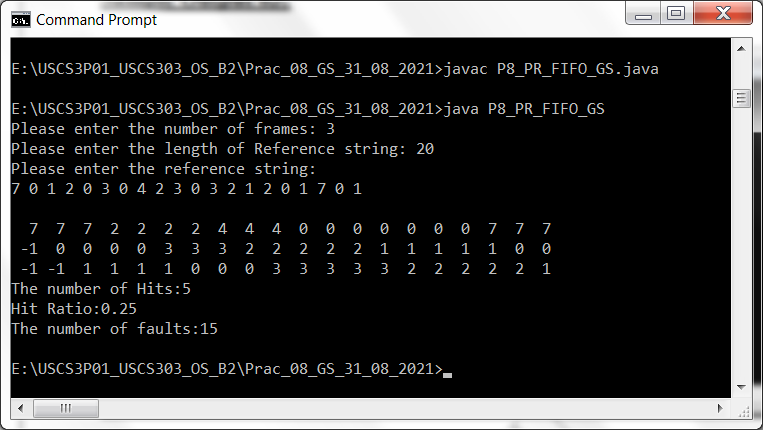
### **Input:**

****

### **Output:**

****

### **Sample Output 03:**

****