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

Contents

[**USCS3P01:USCS303 – Operating System (OS) Practical\_09** 2](#_Toc81385926)

[**Practical Date: 31st August 2021** 2](#_Toc81385927)

[**Practical Aim: Page Replacement Algorithm : LRU** 2](#_Toc81385928)

[**Page Replacement Algorithm** 2](#_Toc81385929)

[**Solved Example:** 3](#_Toc81385930)

[**Do it Yourself 1 :** 4](#_Toc81385931)

[**Do it Yourself 2 :** 4](#_Toc81385932)

[**Question:** 4](#_Toc81385933)

[**Implementation** 4](#_Toc81385934)

[**Input:** 7](#_Toc81385935)

[**Output:** 7](#_Toc81385936)

[**Sample Output01:** 8](#_Toc81385937)

[**Input2:** 8](#_Toc81385938)

[**Output2:** 9](#_Toc81385939)

[**Sample Output02:** 9](#_Toc81385940)

[**Input:** 10](#_Toc81385941)

[**Output:** 11](#_Toc81385942)

[**SampleOutput3:** 12](#_Toc81385943)

### **USCS3P01:USCS303 – Operating System (OS) Practical\_09**

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

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

**Page Replacement Algorithm : LRU**

* **Content**: In page replacement algorithm the page that has not been used for the longest Period of the time is chosen and replaced.
* **Process**: Implement LRU algorithm and find out page hits and page faults.
* **Prior Knowledge**: Page Replacement Algorithm.

### **Page Replacement Algorithm**

* In demand paging memory management technique, if a page demanded for execution is not present in main memory, then a page fault occurs.
* To load the page in demand into main memory, a free page frame is search in main memory and allocated.
* If no page frame is free, Memory Manager has to free a frame by swapping its content to secondary storage and thus make room for the required page.
* To swap pages many screens of strategies are used.

**Least Recently Used (LRU)**

* The Least Recently used (LRU) algorithm replaces the page that has not been used for the longest period of time.
* It is based on the observation that pages that have not been used for the long time with probably remain unused for the longest time and are to be replaced.

### **Solved Example:**

* Apply the LRU replacement algorithms for the following page-reference strings :7,0 , 1 , 2 ,0 , 3 ,0 , 4 ,2 ,3 ,0 ,3 ,2 .
* Indicate the number of page faults for LRU you algorithm assuming demand paging with four frames.
* Find the number of hits, number of faults and hit ratio.

Page-Reference String :7,0 , 1 , 2 ,0 , 3 ,0 , 4 ,2 ,3 ,0 ,3 ,2

Demand Paging or Number of Frames: 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 |  | 7 |  | 7 |  | 7 |  | 7 |  | 3 |  | 3 |  | 3 |  | 3 |  | 3 |  | 3 |  | 3 |  | 3 |
| -1 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |
| -1 |  | -1 |  | 1 |  | 1 |  | 1 |  | 1 |  | 1 |  | 4 |  | 4 |  | 4 |  | 4 |  | 4 |  | 4 |
| -1 |  | -1 |  | -1 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |  | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 |
| W | W | W | W | R | W | R | W | R | R | R | R | R |

Number of Hits: count of number replacements = 7

Number of faults: count of replacements = 6 [R]

Hit Ratio: Number of hits / len( Ref string)= 7/13 = 0.53846157 [W]

### **Do it Yourself 1 :**

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 LRU Page Replacement Algorithm.

### **Do it Yourself 2 :**

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 LRU page-replacement algorithm.

### **Implementation**

// Name: Gaurang Sanyasi

// Batch: B2

// PRN: 2020016400785461

// Date: 31st August, 2021

// Prac-08: Page Replacement Algorithm LRU

import java.io.\*;

import java.util.\*;

public class P9\_PR\_LRU\_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;

Boolean isFull = false;

int buffer[];

ArrayList<Integer> stack = new ArrayList<Integer>();

int reference[];

int mem\_layout[][];

System.out.print("Please enter the number of frames:");

frames = scan.nextInt();

System.out.print("Please enter the length ofReference 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++)

{

if(stack.contains(reference[i]))

{

stack.remove(stack.indexOf(reference[i]));

}

stack.add(reference[i]);

int search = -1;

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

{

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

{

search = j;

hit++;

break;

}

}

if(search==-1)

{

if(isFull)

{

int min\_loc = ref\_len;

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

{

if(stack.contains(buffer[j]))

{

int temp=stack.indexOf(buffer[j]);

if(temp<min\_loc)

{

min\_loc=temp;

pointer=j;

}

}

}

}

buffer[pointer]=reference[i];

fault++;

pointer++;

if(pointer==frames)

{

pointer=0;

isFull=true;

}

}

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);

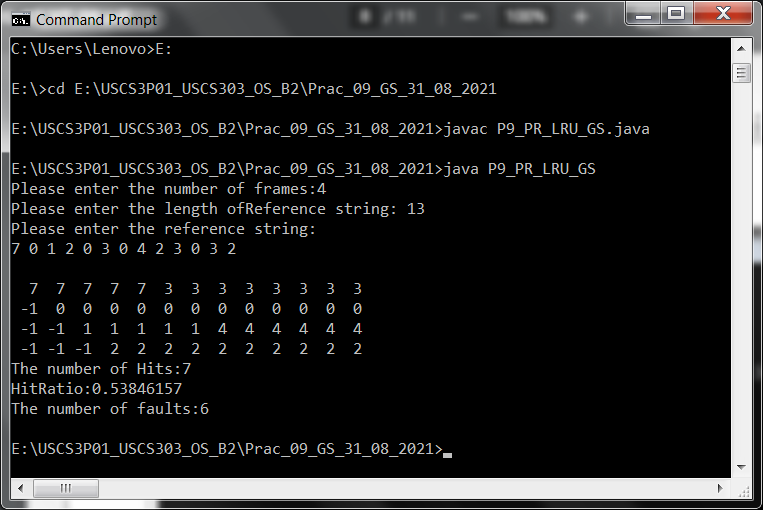
System.out.println("HitRatio:"+(float)((float)hit/ref\_len));

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

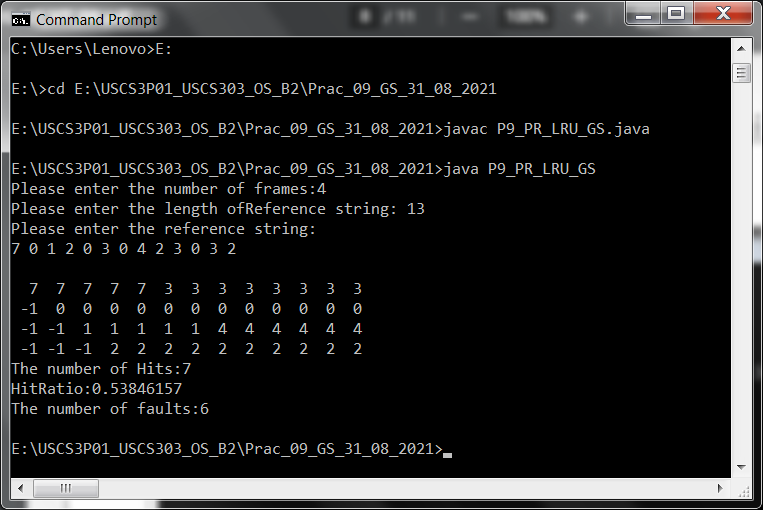
}

}

### **Input:**

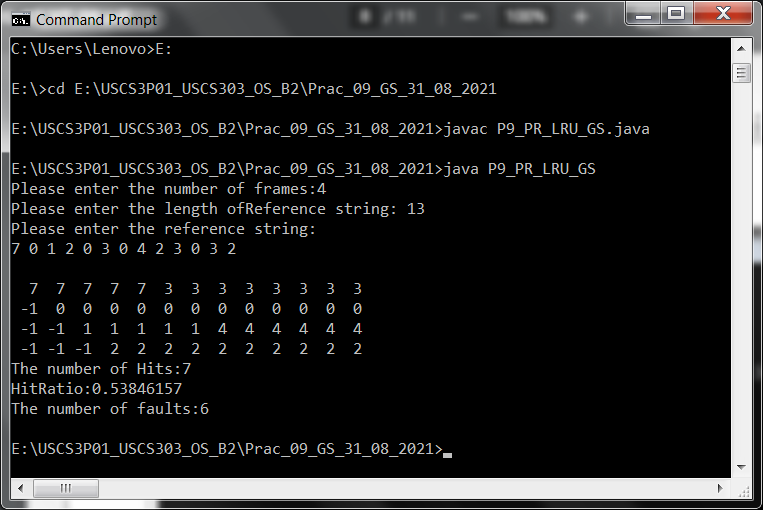
****

### **Output:**

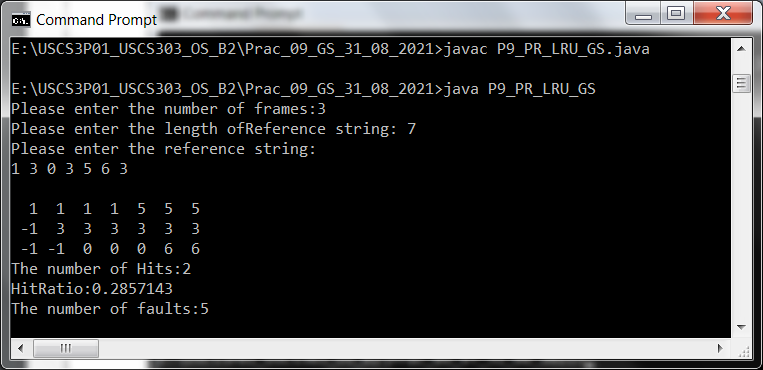
****

### 

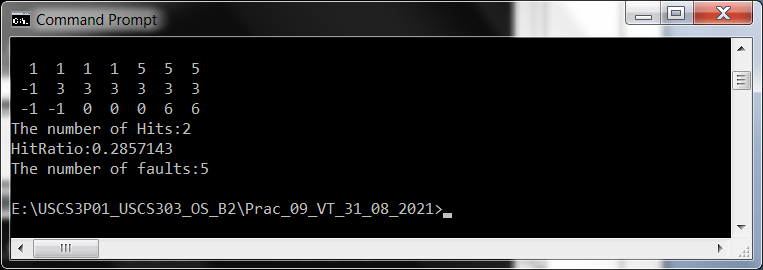
### **Sample Output01:**

****

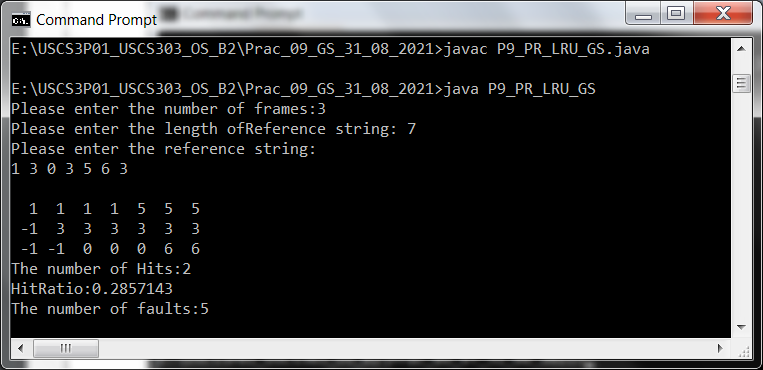
### **Input2:**

****

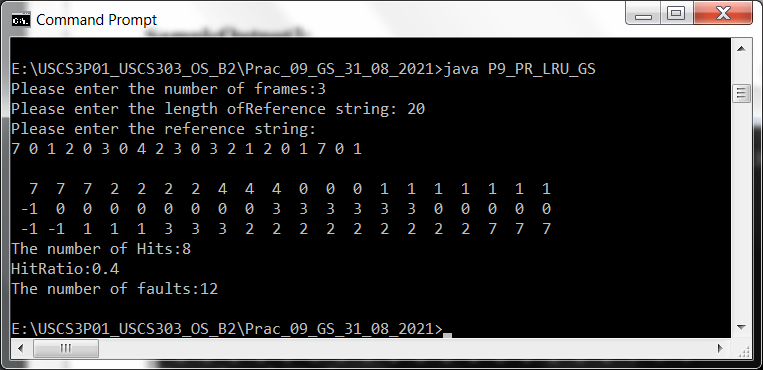
### **Output2:**

****

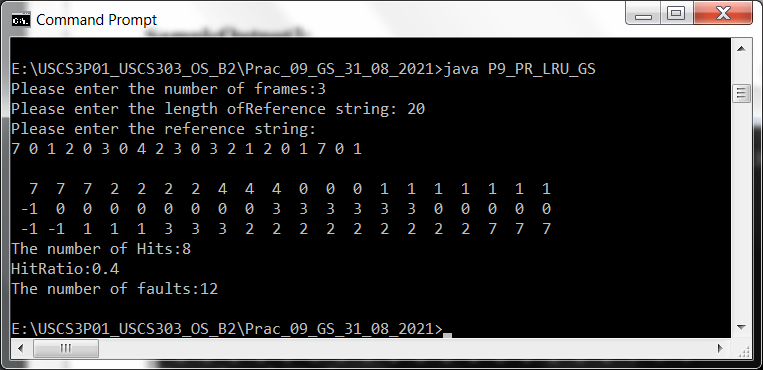
### **Sample Output02:**

****

### **Input:**

****

### **Output:**

****

### 

### **SampleOutput3:**

### 