Green University of Bangladesh
Department of Computer Science and Engineering(CSE)
Faculty of Sciences and Engineering
Semester: (Spring , Year:2024), B.Sc. in CSE (Day)

LAB REPORT NO #06

Course Title: Operating System Lab

Course Code: CSE - 310 **Section: 213_D5**

Lab Experiment Name: Page Replacement Algorithms .

Student Details

Name	ID
MD Dulal Hossain	213902116

Lab Date	:	**29 - 05 - 2024**
Submission Date	:	**05 - 06 - 2024**
Course Teacher's Name	:	**Md. Solaiman Mia**

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status
Marks: ………………………………… **Signature:.....................**
Comments:... **Date:..............................**

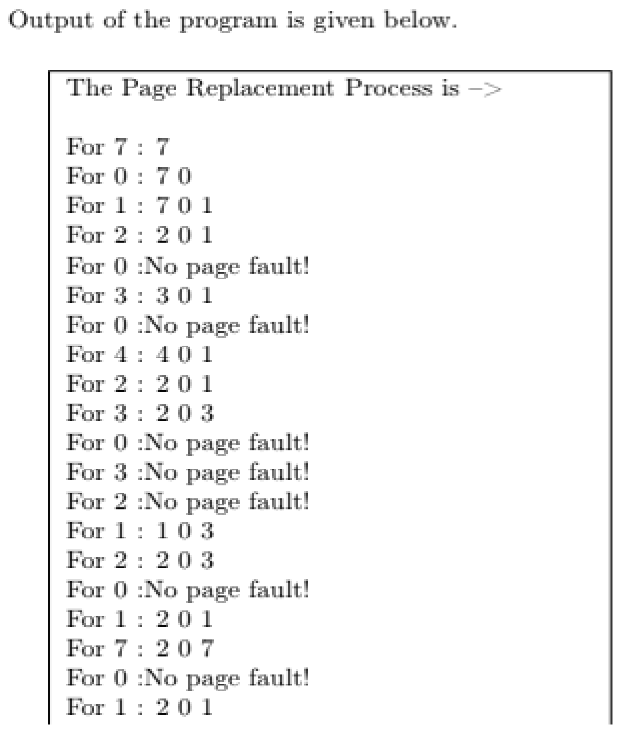
<u>Task :</u>

Title :

Implement LFU (Least Frequently Used) page replacement algorithm.

Input of the program is given below:

```
Enter number of frames: 3
Enter number of pages: 20
Enter reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```



Output of the program is given below.

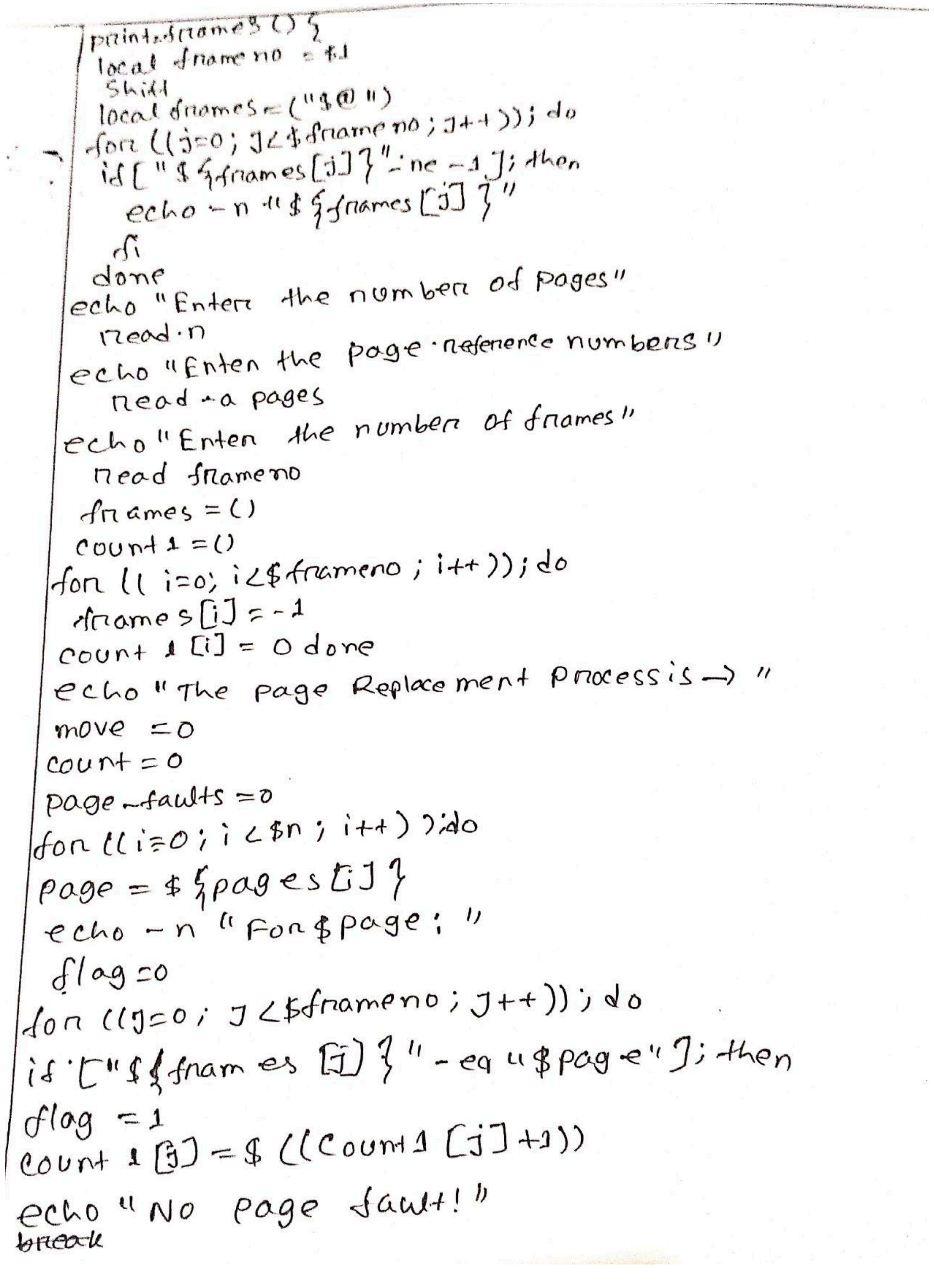
```
The Page Replacement Process is ->

For 7 : 7
For 0 : 7 0
For 1 : 7 0 1
For 2 : 2 0 1
For 0 :No page fault!
For 3 : 3 0 1
For 0 :No page fault!
For 4 : 4 0 1
For 2 : 2 0 1
For 3 : 2 0 3
For 0 :No page fault!
For 3 :No page fault!
For 2 :No page fault!
For 1 : 1 0 3
For 2 : 2 0 3
For 0 :No page fault!
For 1 : 2 0 1
For 7 : 2 0 7
For 0 :No page fault!
For 1 : 2 0 1
```

Algorithms :

1. The script reads the following inputs:
    - n: The number of pages.
    - pages: An array containing the page reference numbers.
    - frame_no: The number of frames available for page storage.
2. It initializes two arrays:
    - frames: An array to store the page numbers currently in the frames. Initially, all frames are set to -1.
    - count1: An array to keep track of how many times each page has been referenced. All counts are initially set to 0.
3. The main loop processes each page reference:
    - For each page, it checks if the page is already in one of the frames.
    - If the page is found in a frame, it increments the reference count for that page and prints “No page fault!”.
    - If the page is not in any frame and there’s space available, it adds the page to an empty frame.
    - If there’s no space available, it replaces the page with the least frequently used page (based on the reference count).
4. The script keeps track of the total number of page faults (page_faults).
5. Finally, it prints the total number of page faults using the LFU (Least Frequently Used) algorithm.

Source Code in Hand Written :

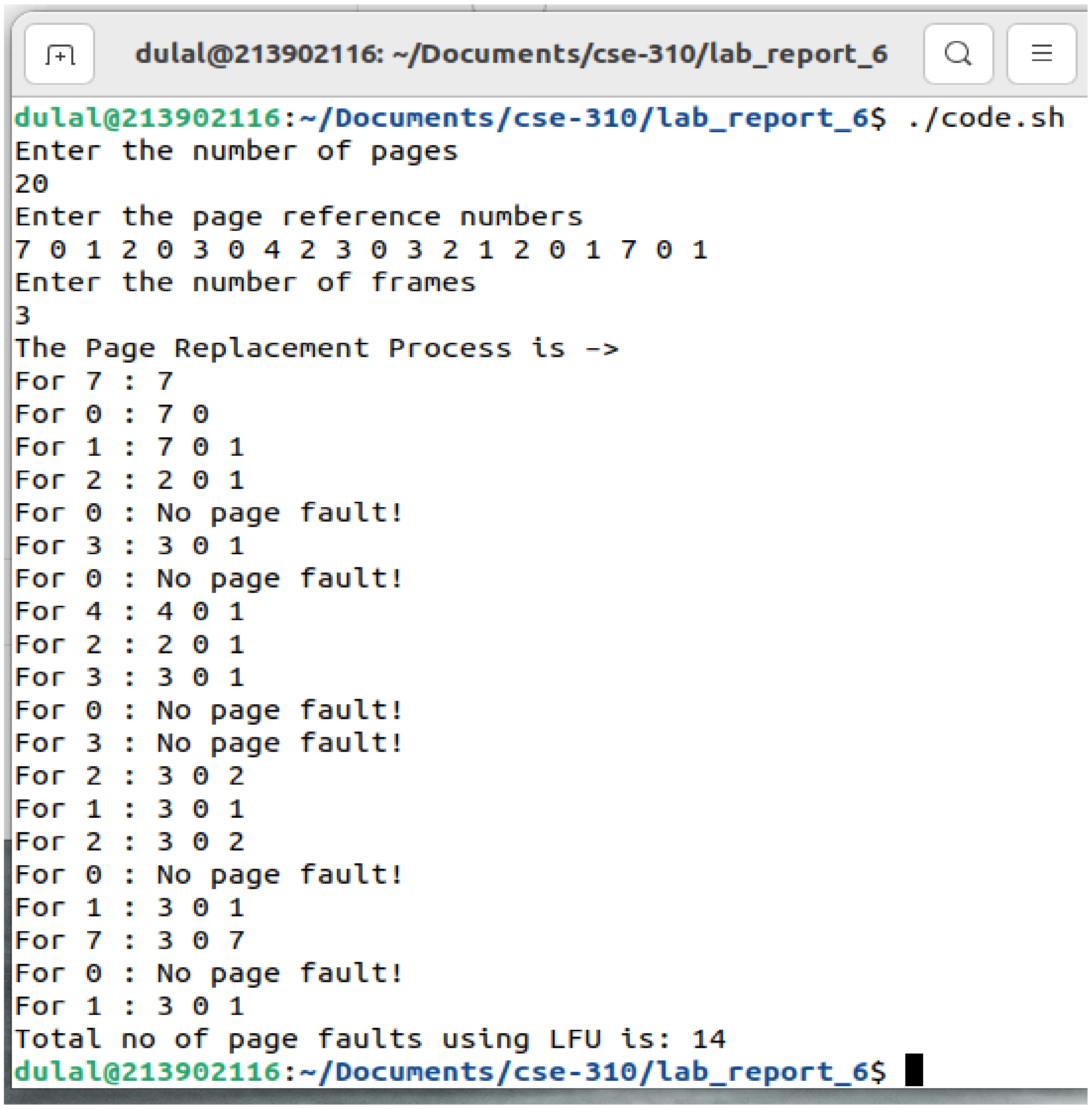


```
printframes () {
 local frameno = $1
  Shift
 local frames=("$@")
for ((j=0; J<$frameno; J++)); do
 if [ "${frames[J]}" -ne -1 ]; then
   echo -n "${frames[J]}"
  fi
 done
echo "Enter the number of pages"
  read n
echo "Enter the page reference numbers"
   read -a pages
echo "Enter the number of frames"
  read frameno
 frames =()
 count1 =()
for (( i=0; i<$frameno ; i++ )); do
 frames[i] = -1
count 1 [i] = 0 done
echo "The page Replacement process is -> "
move =0
count =0
page_faults =0
for ((i=0; i <$n ; i++ ));do
page = ${pages[i]}
 echo -n "For$page: "
 flag =0
for ((J=0; J <$frameno; J++)); do
if ["${fram es [J]}" - eq "$pag e"]; then
flag = 1
count 1 [J] = $ ((Count1 [J] +1))
echo "No page fault!"
break
break
  fi
 done
if [$flag -eq 0] && [$count -lt $frameno]; then
   frames[move] =$page
   count1 [move]=1
   move = $ ((move +1) % frameno))
  count = $ ((count +1))
page-faults =$ ((page-faults+1))
print frames $ frameno "${frames[@]}"
elif [flag- eq 0] ; then
 repindex=0
 leastcount =$ {count1 [0]}
for ((j=1 ; j<$frameno ; j++)); do
 if [ ${count1 [j]}- lt $ leastcount]; then
  repindex = $j
 leastcount = $ {count1[j]}
 fi
done
frames [repindex ]=$page
count1 [repindex]  = 1
page -faults =$ (( page-faults+1))
print_frames $frame no "$ {fames[@]} "
fi
done
echo "Total  no of  page faults using LFU is
              : $ page-faults".
```

Source Code in write :

```
print_frames() {
  local frameno=$1
  shift
  local frames=("$@")
  for ((j=0; j<$frameno; j++)); do
    if [ "${frames[j]}" -ne -1 ]; then
      echo -n "${frames[j]} "
    fi   done
  echo}
echo "Enter the number of pages"
read n
echo "Enter the page reference numbers"
read -a pages
echo "Enter the number of frames"
read frameno
frames=()
count1=()
for ((i=0; i<$frameno; i++)); do
  frames[i]=-1
  count1[i]=0 done
echo "The Page Replacement Process is –>"
move=0
count=0
page_faults=0
for ((i=0; i<$n; i++)); do
  page=${pages[i]}
  echo -n "For $page : "
  flag=0
  for ((j=0; j<$frameno; j++)); do
    if [ "${frames[j]}" -eq "$page" ]; then
      flag=1
      count1[j]=$((count1[j] + 1))
      echo "No page fault!"
      break
    fi  done
  if [ $flag -eq 0 ] && [ $count -lt $frameno ]; then
    frames[move]=$page
    count1[move]=1
    move=$(( (move + 1) % frameno ))
    count=$((count + 1))
    page_faults=$((page_faults + 1))
    print_frames $frameno "${frames[@]}"
  elif [ $flag -eq 0 ]; then
    repindex=0
    leastcount=${count1[0]}
    for ((j=1; j<$frameno; j++)); do
      if [ ${count1[j]} -lt $leastcount ]; then
        repindex=$j
        leastcount=${count1[j]}
      fi  done
    frames[repindex]=$page
    count1[repindex]=1
    page_faults=$((page_faults + 1))
    print_frames $frameno "${frames[@]}"
  Fi done
echo "Total no of page faults using LFU is: $page_faults"
```

Output :



```
dulal@213902116:~/Documents/cse-310/lab_report_6$ ./code.sh
Enter the number of pages
20
Enter the page reference numbers
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames
3
The Page Replacement Process is ->
For 7 : 7
For 0 : 7 0
For 1 : 7 0 1
For 2 : 2 0 1
For 0 : No page fault!
For 3 : 3 0 1
For 0 : No page fault!
For 4 : 4 0 1
For 2 : 2 0 1
For 3 : 3 0 1
For 0 : No page fault!
For 3 : No page fault!
For 2 : 3 0 2
For 1 : 3 0 1
For 2 : 3 0 2
For 0 : No page fault!
For 1 : 3 0 1
For 7 : 3 0 7
For 0 : No page fault!
For 1 : 3 0 1
Total no of page faults using LFU is: 14
dulal@213902116:~/Documents/cse-310/lab_report_6$
```

Figure 5.1 : Output in show Successfully.

Explain Output :

Input Parameters:
n: The number of pages 20.
pages: An array containing the page reference numbers (7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 2, 1, 2, 0, 1, 7, 0, 1).
frame_no: The number of frames available for page storage 3.
Initialization:
The script initializes two arrays:
frames: An array to store the page numbers currently in the frames. Initially, all frames are set to -1.
count1: An array to keep track of how many times each page has been referenced. All counts are initially set to 0.

Page Replacement Process:
The script processes each page reference:
For each page, it checks if the page is already in one of the frames.
If the page is found in a frame, it increments the reference count for that page and prints “No page fault!”.
If the page is not in any frame and there’s space available, it adds the page to an empty frame.
If there’s no space available, it replaces the page with the least frequently used page (based on the reference count).
Output Explanation:
The output shows the process for each page reference:
For example, for page 7, it prints For 7 : 7 (indicating that page 7 is added to frame 7).
When page 0 is referenced again, it prints For 0 : 7 0 (indicating that page 0 is already in frame 7 and 0 is added to frame 0).
Similarly, it processes other pages and prints the updated frames.
The total number of page faults using the LFU algorithm is 14.