OPERATING SYSTEM LAB (COM -312)

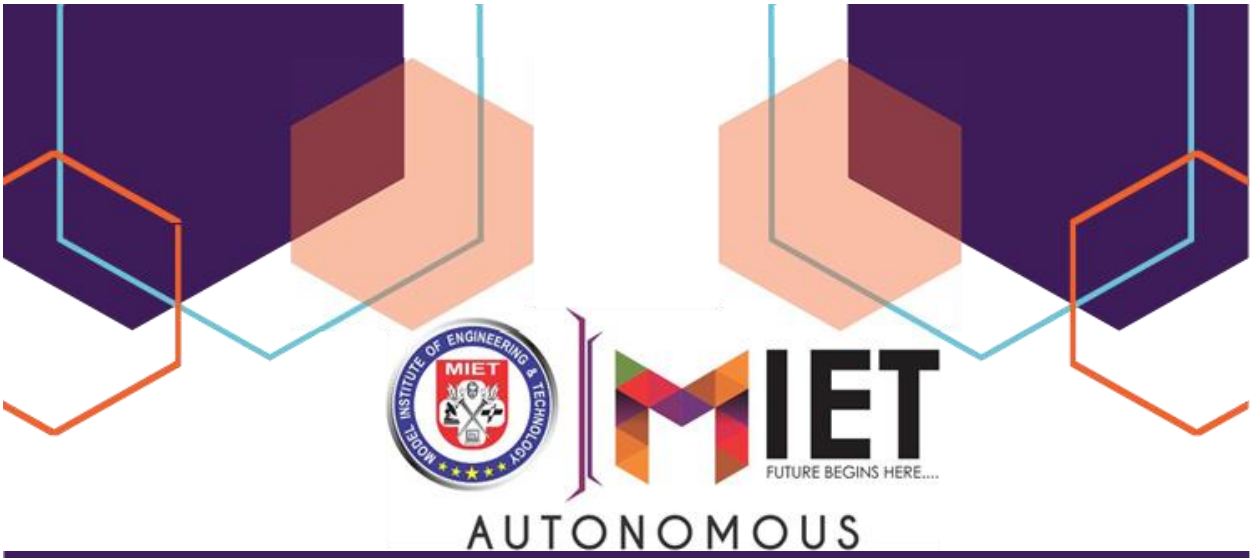
Implementation of the Functionality of FIFO, LRU, and OPTIMAL Page Replacement Algorithms At

CSE, MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY

BACHELOR OF ENGINEERING
In
Computer Science & Engineering

SUBMITTED BY

AYUSH THAKUR & SONALI TANDON
Roll Number: 2021A1R049 & 2021A1R053



SUBMITTED TO

Department of Computer Science & Engineering
Model Institute of Engineering and Technology (Autonomous)
Jammu, India

ACKNOWLEDGEMENT

Under CRIE (Center for Research, Innovation and Entrepreneurship) we both worked under the guidance of mentors of CRIE, who guided us through a way leading to professionalism and practical hands on work experience.It is indeed with a great pleasure and immense sense of gratitude that we acknowledge the help of these individuals. We are highly indebted to our Director Ankur Gupta, "MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY", for the facilities provided to accomplish this main project. We would like to thank our Prof.Ashok Kumar , Head of the Department of Computer Science and Engineering , MIET. For this constructive criticism throughout our project .We feel elated in manifesting our sense of gratitude for our internal project guide. Asst. Saurabh Sharma , Department of Computer Science and Engineering ,MIET.He has been a constant source of inspiration for us and we are very deeply thankful to him.

FACULTY AND MEMBER

Dr. Ankur Gupta - He is the Director at the ModeI Institute of Engineering and Technology, Jammu, India, besides being a Professor in the Department of Computer Science and Engineering. Prior to joining academia, he worked as Technical Team Lead at Hewlett Packard, developing software in the network management and e-Commerce domains. He has 2 patents to his name and 20 patents pending. He obtained his B.E, Hons. He is a senior member of the ACM, senior member IEEE and a life-member of the Computer Society of India. He has received competitive grants over Rs 2 crores from various funding agencies and faculty awards from IBM and EMC.

Prof. Ashok Kumar - He has over 21 years of experience in industry, academics, research and academic administration. He did his Doctorate from IKG Punjab Technical University, Jalandhar; Master of Engineering from Punjab Engineering College, Chandigarh and Bachelor's degree from NIT, Calicut, Kerala. He is Fellow of Institution of Electronics and Telecommunication Engineers (IETE). His research interest areas are Optical Networks, Electronics Product Design, Wireless Sensor Networks and Digital Signal Processing.. He is reviewer of many reputed national and international journals. He has also coordinated many national and international conferences.

Mr. Saurabh Sharma - He is working as Assistant Professor in the Department of Computer Science at Model Institute of Engineering and Technology, Jammu and has over 10 years of experience in academics and research. He has done his M. Tech CSE in Web Mining from Maharishi Markandeshwar University, Mullana, Ambala (HR) in the year 2011.His research interest areas are Web Mining, Data Mining, Machine Learning, ANN, Pattern Classification and Object Identification. He has, to his credit, 25 research papers published in journals of national and international repute and he has guided 11 M. Tech Dissertations.

Abstract

As the technology has advanced and the program has become more and more complex , the RAM size is not increasing a lot. So, typically when the multi scenario tasks come into picture. There are a lot of apps which are running in the background and when you open up more complex applications, especially Graphic intensive applications they take more RAM and more processing, so this makes the entire ram full and thus memory management comes into picture. Memory management is the process of controlling and coordinating a computer's main memory. Paging is one such methodology in which memory is efficiently managed. In computer operating systems, paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. In this game, the operating system. Retrieve data from secondary storage in same size blocks called a page. It is actually a part of virtual memory implementations in modern operating systems. It helps to prevent external fragmentation. The Memory Management Unit usually divides the virtual address space into pages and therefore is called the Paged Memory Management Unit.Major page replacement algorithms are attempted to be summarized in this study. We examine conventional algorithms like FIFO, LRU, and optimal replacement.

Index Terms: Page replacement Algorithms, page fault, page hit, page miss, Hit ratio, Paging, Main memory

CONTENTS

1. Introduction

In computer operating systems, Paging is one of the memory management schemes by which a computer can store and retrieve data. Forms can reach storage for use in main memory. Whenever a paging memory management scheme is implemented in any kind of operating system, who decides which memory pages to page out. Sometimes called swap out or write to disk and when a page of memory needs to be allocated. Thus, this algorithm basically dictates how and when a page is supposed to be removed from physical memory and shifted to the virtual memory and vice versa. So, whenever a new page is referenced and not present in physical memory (RAM) page fault occurs and the operating system replaces one of the existing pages with a newly needed page. So, actually there are 2 scenarios, the first one is whether the memory is full and you have to replace a page with another page or the memories are not full, so find an empty frame and add that page from virtual memory into that frame. Thus, a page fault happens when the page is not found in the physical memory then a page fault interrupt is generated. An interrupt is a high priority signal to the op. This R attempts to summarize major page replacement algorithms which states that it requires a necessary page from virtual memory into the physical memory.

2. Objective

Simulator should accept the number of physical frames, list of page accesses, and the page replacement algorithm and output the number of faults and whether each access was a fault or not. Implementation of FIFO,LRU,and OPTIMAL Page Replacement Algorithms. There must be an input file (*.txt) in the directory. The first line in the file should contain the number of physical frames. Each subsequent line represents one page access, and contains exactly one integer, which represents the page number being accessed. All values must be non-negative and fit in the int data type of the system.

3. Terminology

Page - Paging is that process where a page from virtual memory is swapped into the main memory which is of fixed size. Thus it is a fixed-length contiguous block of virtual memory.

Logical Address -Basically logical address is generated when a program is running.by the CPU logical address or virtual address do not exist physically, therefore. It is called a virtual address. This address is used as a reference to assess the physical memory location by CPU.

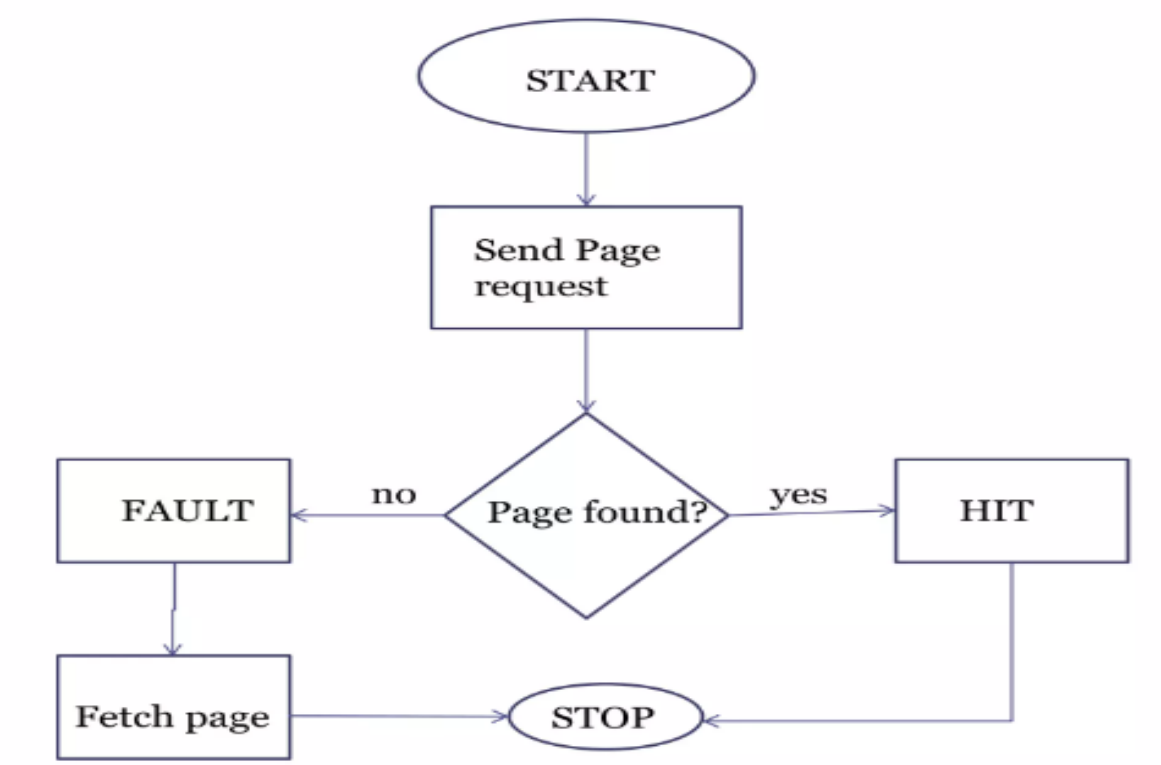
Physical Address- Actually it defines the physical location of the required data in a memory. The user never deals with physical address but can access it by corresponding logical address. Thus logical addresses must be mapped to the physical address by memory management unit before they are used.

Page fault - Page fault takes control like an error. A page fault will occur if a program tries to access memory that isn't physically present in main memory, or if it's not even present there. The

fault instructs the operating system to relocate all data from secondary memory, such as the hard drive, to the system's primary memory by tracing it through virtual memory management.

Page Hit-Page hits occur when a page is requested to be loaded into memory but is already present in the memory.

4. Page Replacement Algorithm



(Fig. 1)

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).There are two main aspects of virtual memory, Frame allocation and Page Replacement. It is very important to have the optimal frame allocation and page replacement algorithm. Frame allocation is all about how many frames are to be allocated to the process while the page replacement is all about determining the page number which needs to be replaced in order to make space for the requested page.

A. First in First Out (FIFO)

FIFO algorithm is the simplest of all the page replacement algorithms. According to this algorithm, when a page fault occurs, if there are one or more free frames in a memory the newly called page will be loaded into it or any of them. However, in cases where there are no free frames, the page with the longest memory residence time will be selected to exist. This means that pages enter memory before all other pages will exit memory before them. The idea behind this policy is that the first landing party has many opportunities to use it and the second should be given an opportunity. This page algorithm has many problems. First, if a page is used repeatedly, eventually it will be recognized as a new or existing page and can be selected from memory even when it is most needed. In this case the selection becomes inactive because the resulting page should be loaded into memory almost immediately. In this, we maintain a queue of all the pages that are in the memory currently. The oldest page in the memory is at the front-end of the queue and the most recent page is at the back or rear-end of the queue.

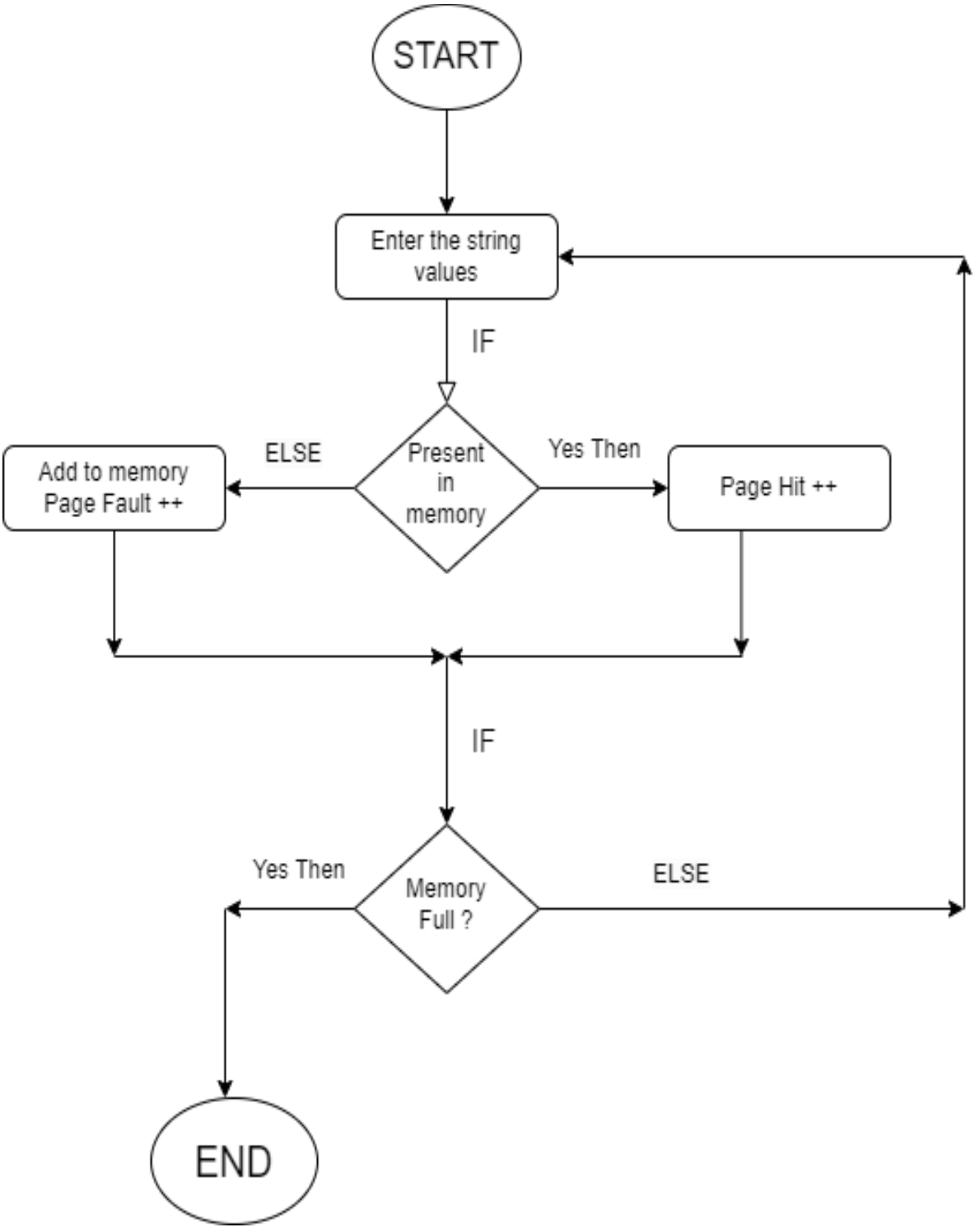
B. Least Recently Used (LRU)

Least Recently Used algorithm is a page replacement technique used for memory management. Usually, page references originate from the way they are aggregated into a single field. This algorithm believes in this principle and chooses the compromised area as one that should not be visited for a long time. This technique is difficult to implement as it requires careful registration of the trees. One way to use this is to timestamp each reference page. According to this method, that page is replaced with the one which is least recently used. Thus, in memory any page that has been unused for a longer period of time than the others are replaced.

C. Optimal Page Replacement

This algorithm chooses the one whose reference is farthest. But it still has excellent page fault behavior because it can communicate more memory references in less time. The optimal page replacement algorithm replaces the most recently used page with the lowest page error rate for a given page reference stream. The problem here is knowing the future perfectly.

5. Flowchart



(Fig. 2)

6. Pseudo Code

A. FIFO

#declare variable: values, pages, frames, pageFaults, pageHits
#initialize pageHits=0 and pageFaults=0
Enter the number of pages.
Enter the reference string values.
Enter the number of frames.
Start traversing the string values into the frame.
Check if string value is already present in the memory.
If yes (pageHit)
Else (pageFault)
Print pageHits and pageFaults

B. LRU

Iterate through the referenced pages.
If the current page is already present in pages:
Remove the current page from pages. Append the current page to the end of pages.
Increment page hits.
Else:
Increment page faults.
If pages contain less pages than its capacity s:
Append current page into pages.
Else:
Remove the first page from pages.
Append the current page at the end of pages.
Return the number of page hits and page faults.

C. Optimal Page Replacement

Iterate through the referenced pages.
If the current page is already present in pages:
Remove the future page from upcoming pages.
Append the future page to the end of pages.
Increment page hits.
Else:
Increment page faults.
If pages contain less pages than its capacity s:
Append current page into pages.

Else:

Remove the last page from pages.

Append the current page at the end of pages.

Return the number of page hits and page faults.

7. Working procedure for Page Replacement Algorithm.

Page replacement algorithms help to decide which page must be swapped out from the main memory to create a room for the incoming page. The page to be replaced is chosen using page replacement algorithms. Three page replacement methods are explored in this study.

The system is using 3 page frames for storing the pages/reference strings in the main memory. Assuming that all the page frames are initially empty while processing the page reference string given.

1. FIFO (First In First Out)

Reference String

3	2	1	3	4	1	6	2

Total number of Reference string = 8

Page Frame

3	3	3	3	4	4	4	4
	2	2	2	2	2	6	6
	1	1	1	1	1	1	2

Total Page Frames = 3

Number Of Faults and Page Hits

F	**F**	**F**	**H**	**F**	**H**	**F**	**F**

In the above reference string, with 3 page frames:

Total number of page faults = 6

Total number of page hits = Total no. of references - Total no. of page faults

= 8 - 6 => 2

Hit Ratio = Total no. of page hits / Total no. of references

= 2 / 8 => 0.25

2. LRU

Reference String

3	2	1	3	4	1	6	2

Total number of Reference string = 8

Page Frame

3	3	3	3	3	3	6	6
	2	2	2	4	4	4	2
	1	1	1	1	1	1	1

Total Page Frames = 3

Number Of Faults and Page Hits

F	**F**	**F**	**H**	**F**	**H**	**F**	**F**

In the above reference string, with 3 page frames:

Total number of page faults = 6

Total number of page hits = Total no. of references - Total no. of page faults

= 8 - 6 => 2

Hit Ratio = 2 / 8 => 0.25

3. Optimal Page Replacement

Reference String

3	2	1	3	4	1	6	2

Total number of Reference string = 8

Page Frame

3	3	3	3	4	4	4	4

	2	2	2	2	2	2	2
	1	1	1	1	1	6	6

Total Page Frames = 3

Number Of Faults and Page Hits

F	**F**	**F**	**H**	**F**	**H**	**F**	**H**

In the above reference string, with 3 page frames:

Total number of page faults = 5

Total number of page hits = Total no. of references - Total no. of page faults

= 8 - 5 => 3

Hit Ratio = Total number of page hits / Total number of references

= 3 / 8 => 0.375

9. Implementation

a) Coding

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <limits.h>
//FIND LRU
int findLRU(int time[], int n){
int i, minimum = time[0], pos = 0;
for(i = 1; i < n; ++i){
if(time[i] < minimum){
minimum = time[i];
```

```
pos = i;
} }
return pos;
}
//FIND LRU
//FUNCTIONS
void homescreen();
void back();
void fifo();
void lru();
void opr();
//FUNCTIONS
//code
int main()
{
   system("color 03");
   homescreen();
}
void homescreen()
{
   int ch;
   printf("Select an option for page replacement\n\n");
   printf("1.FIFO (First In First Out)\n2.LRU (Least Recently Used)\n3.Optimal Page
Replacement\n4.Exit");
   printf("\n\nEnter your choice: ");
   ch=toupper(getch());
   if(ch=='1')
    {
       system("cls");
       printf("**************************************************\n");
```

```
        printf("You have selected FIFO page replacement algorithm\n");
        printf("*************************************************\n");
        fifo();
    }
    else if(ch=='2')
    {
        system("cls");
printf("*************************************************\n");
        printf("You have selected LRU page replacement algorithm\n");
        printf("*************************************************\n");
      lru();
    }
    else if(ch=='3')
    {
        system("cls");
        printf("*************************************************\n");
        printf("You have selected Optimal page replacement algorithm\n");
        printf("*************************************************\n");
        opr();
    }
    else if(ch=='4')
    {
        system("cls");
        printf("\n\n\t\t\t\t\t\tThanks for using the program\n\n\n");
        getch();
        exit(1);
    }
    else
    {
        printf("(Invalid Choice)");
```

```
        printf("\n\nPress any key to continue");
        getch();
        system("cls");
        homescreen();
    }
}
void back()
{
    int ch;
    system("cls");
    printf("Thanks for using the program\n");
    printf("To use the program again press R\n");
    printf("To exit the program press E\n");
    ch=toupper(getch());
    if(ch=='R' || ch=='r')
    {
        system("cls");
        homescreen();
    }
    else if(ch=='E' || ch=='e')
    exit(1);
    else
    {
        printf("\nInvalid Option..");
        printf("\n\nPress any key to continue");
        getch();
        back();
    }
}
//FIFO PAGE REPLACEMENT
```

```
void fifo()
{
//declaring variables..
int values[50], pageFaults = 0, m, n, s, pages, frames,pageHits=0;
printf("\nEnter the number of Pages: ");
scanf("%d", &pages); //taking input for total number of pages
printf("\nEnter the string values:\n");
//for loop start
for( m = 0; m < pages; m++)
{
printf("Value No. [%d]: ", m + 1);
scanf("%d", &values[m]);//taking input for the values
}
//for loop end
printf("\nEnter total number of frames: ");
scanf("%d", &frames); //taking input for frames

int temp[frames];
for(m = 0; m < frames; m++)
{
temp[m] = -1;
}
for(m = 0; m < pages; m++)
{
s = 0;
for(n = 0; n < frames; n++)
{
if(values[m] == temp[n])
{
s++;
```

```
pageFaults--;
}
}
pageFaults++;
if((pageFaults <= frames) && (s == 0))
{
temp[m] = values[m];
}
else if(s == 0)
{
temp[(pageFaults - 1) % frames] = values[m];
}
printf("\n");
for(n = 0; n < frames; n++)
{
printf("%d\t", temp[n]);
}
}
pageHits=pages-pageFaults;
printf("\n\n");
printf("\nTotal Page Faults: %d\n", pageFaults);
printf("Total Page Hits: %d\n",pageHits);
printf("\n\nPress any key to continue");
getch();
back();
}
//FIFO PAGE REPLACEMENT

//LRU PAGE REPLACEMENT
void lru()
```

```
{
int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1,
flag2, i, j, pos, faults = 0, hits=0;
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("\nEnter reference string\n");
for(i = 0; i < no_of_pages; ++i)
{
printf("Value No. [%d]: ", i + 1);
scanf("%d", &pages[i]);
}
printf("\nEnter number of frames: ");
scanf("%d", &no_of_frames);
for(i = 0; i < no_of_frames; ++i){
frames[i] = -1;
}
for(i = 0; i < no_of_pages; ++i){
flag1 = flag2 = 0;
for(j = 0; j < no_of_frames; ++j){
if(frames[j] == pages[i]){
counter++;
time[j] = counter;
flag1 = flag2 = 1;
break;
}
}
if(flag1 == 0){
for(j = 0; j < no_of_frames; ++j){
if(frames[j] == -1){
counter++;
```

```
faults++;
frames[j] = pages[i];
time[j] = counter;
flag2 = 1;
break;
}
}
}
if(flag2 == 0){
pos = findLRU(time, no_of_frames);
counter++;
faults++;
frames[pos] = pages[i];
time[pos] = counter;
}
printf("\n");
for(j = 0; j < no_of_frames; ++j){
printf("%d\t", frames[j]);
}
}
hits = no_of_pages - faults;
printf("\n\nTotal Page Faults = %d", faults);
printf("\nTotal Page Hits = %d", hits);
printf("\n\nPress any key to continue");
getch();
back();
}
//LRU PAGE REPLACEMENT

//OPTIMAL PAGE REPLACEMENT
```

```
void opr()
{
int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j,
k, pos, max, faults = 0, hits=0;
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("\nEnter page reference string\n");
for( i = 0; i < no_of_pages; i++)
{
printf("Value No. [%d]: ", i + 1);
scanf("%d", &pages[i]);//taking input for the values
}
printf("\nEnter number of frames: ");
scanf("%d", &no_of_frames);
for(i = 0; i < no_of_frames; ++i){
frames[i] = -1;
}
for(i = 0; i < no_of_pages; ++i){
flag1 = flag2 = 0;
for(j = 0; j < no_of_frames; ++j){
if(frames[j] == pages[i])
{
flag1 = flag2 = 1;
break;
}
}
if(flag1 == 0){
for(j = 0; j < no_of_frames; ++j){
if(frames[j] == -1){
faults++;
```

```
frames[j] = pages[i];
flag2 = 1;
break;
}
}
}
if(flag2 == 0){
flag3 =0;
for(j = 0; j < no_of_frames; ++j){
temp[j] = -1;
for(k = i + 1; k < no_of_pages; ++k){
if(frames[j] == pages[k]){
temp[j] = k;
break;
}
}
}
for(j = 0; j < no_of_frames; ++j){
if(temp[j] == -1){
pos = j;
flag3 = 1;
break;
}
}

if(flag3 ==0){
max = temp[0];
pos = 0;
for(j = 1; j < no_of_frames; ++j){
if(temp[j] > max){
```

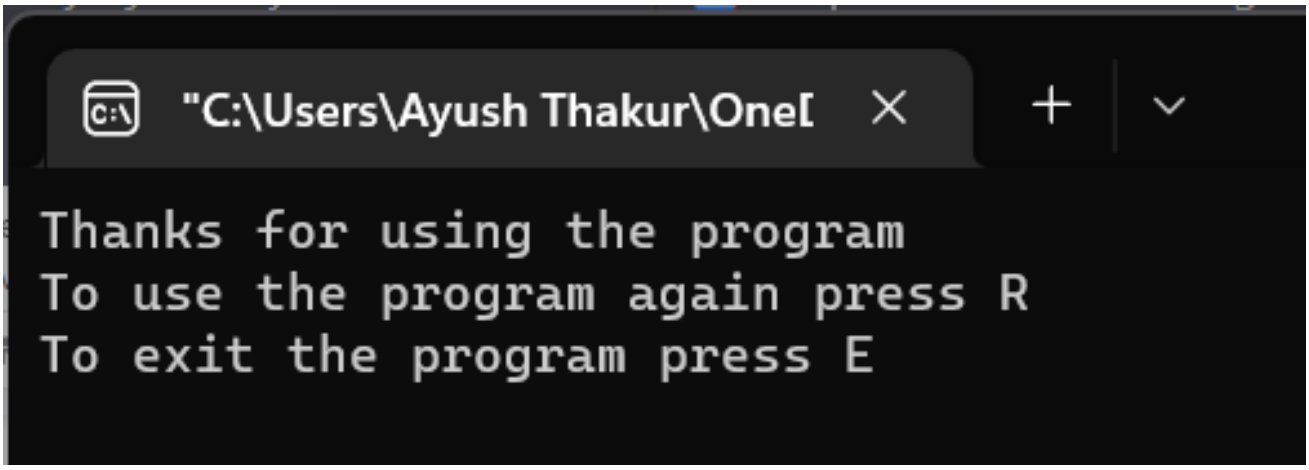
```
max = temp[j];
pos = j;

}
}
}

frames[pos] = pages[i];
faults++;
}
printf("\n");
for(j = 0; j < no_of_frames; ++j){
printf("%d\t", frames[j]);

}
}

hits = no_of_pages - faults;
printf("\n\nTotal Page Faults = %d", faults);
printf("\nTotal Page Hits = %d", hits);
printf("\n\nPress any key to continue");
getch();
back();

}

//OPTIMAL PAGE REPLACEMENT
```

b) Output

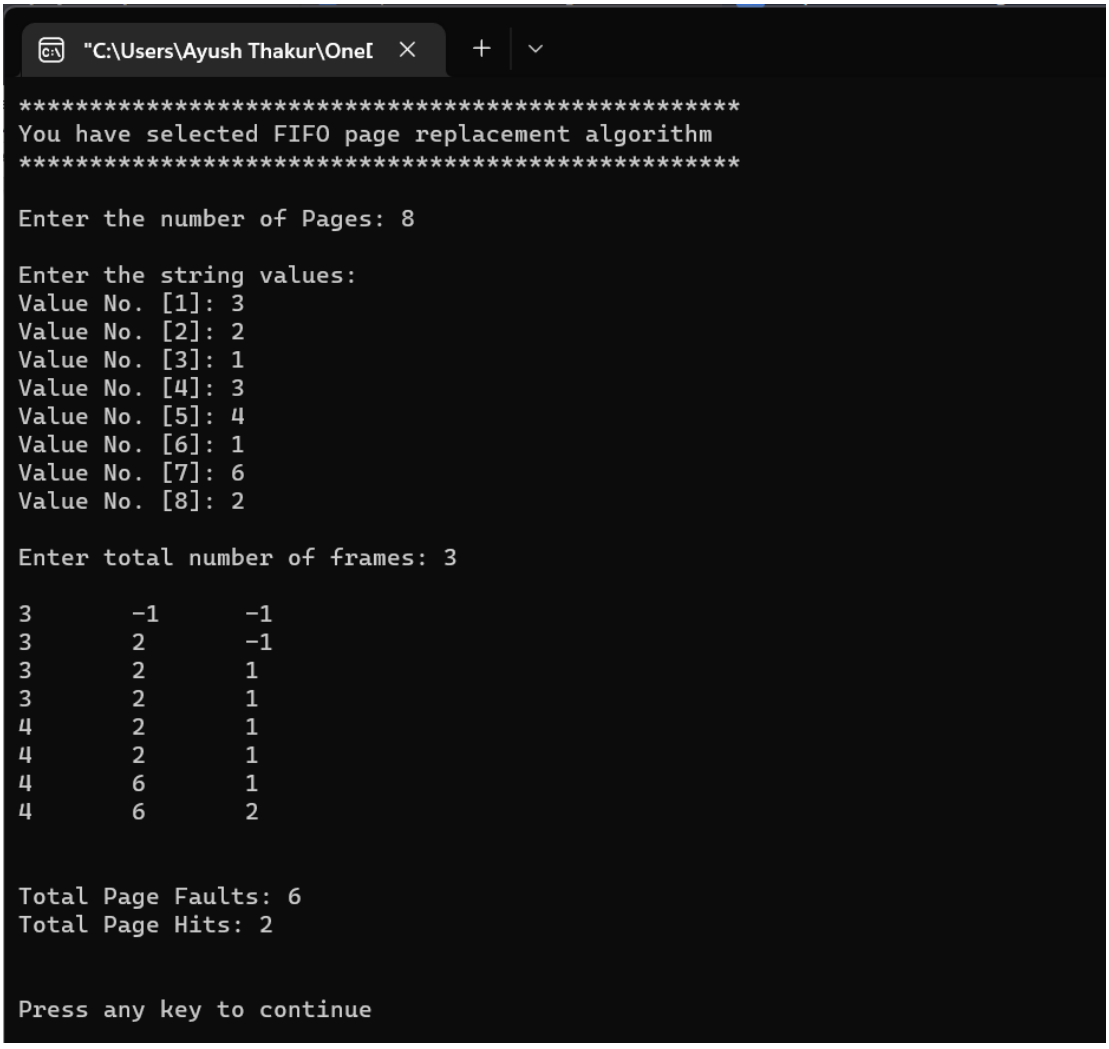
void homeScreen();

```
Select an option for page replacement

1.FIFO (First In First Out)
2.LRU (Least Recently Used)
3.Optimal Page Replacement
4.Exit

Enter your choice:
```

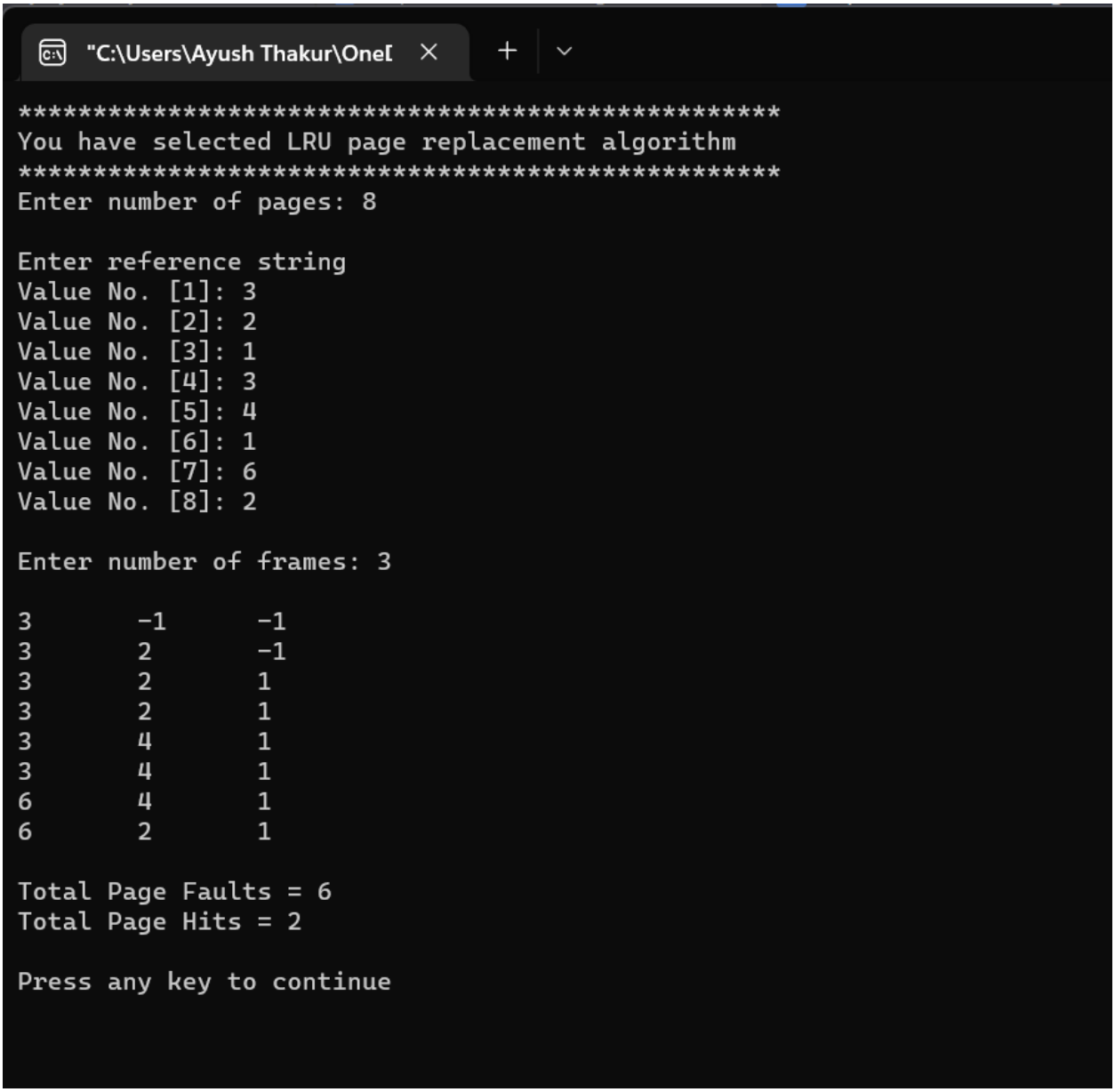
void back();



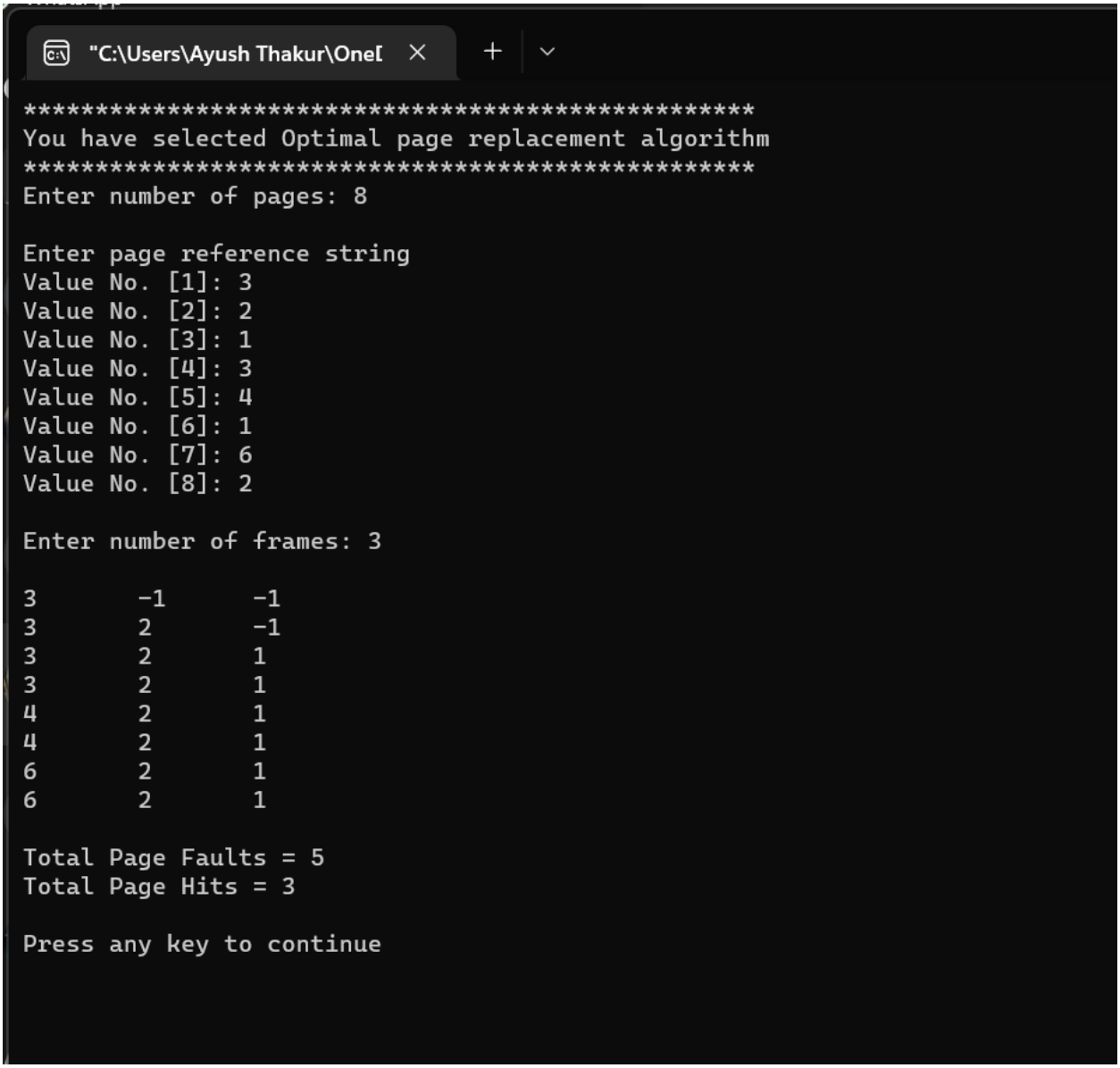
void fifo();


"C:\Users\Ayush Thakur\OneI
****
You have selected FIFO page replacement algorithm
****
Enter the number of Pages: 8
Enter the string values:
Value No. [1]: 3
Value No. [2]: 2
Value No. [3]: 1
Value No. [4]: 3
Value No. [5]: 4
Value No. [6]: 1
Value No. [7]: 6
Value No. [8]: 2
Enter total number of frames: 3
3 -1 -1
3 2 -1
3 2 1
3 2 1
4 2 1
4 2 1
4 6 1
4 6 2
Total Page Faults: 6
Total Page Hits: 2
Press any key to continue

void lru();


"C:\Users\Ayush Thakur\One[
****
You have selected LRU page replacement algorithm
****
Enter number of pages: 8
Enter reference string
Value No. [1]: 3
Value No. [2]: 2
Value No. [3]: 1
Value No. [4]: 3
Value No. [5]: 4
Value No. [6]: 1
Value No. [7]: 6
Value No. [8]: 2
Enter number of frames: 3
3 -1 -1
3 2 -1
3 2 1
3 2 1
3 4 1
3 4 1
6 4 1
6 2 1
Total Page Faults = 6
Total Page Hits = 2
Press any key to continue

void opr();


"C:\Users\Ayush Thakur\One[
****
You have selected Optimal page replacement algorithm
****
Enter number of pages: 8
Enter page reference string
Value No. [1]: 3
Value No. [2]: 2
Value No. [3]: 1
Value No. [4]: 3
Value No. [5]: 4
Value No. [6]: 1
Value No. [7]: 6
Value No. [8]: 2
Enter number of frames: 3
3 -1 -1
3 2 -1
3 2 1
3 2 1
4 2 1
4 2 1
6 2 1
6 2 1
Total Page Faults = 5
Total Page Hits = 3
Press any key to continue

References

1. https://www.geeksforgeeks.org/page-replacement-algorithms-in-operating-systems/
2. https://www.javatpoint.com/os-page-replacement-algorithms
3. https://www.scaler.com/topics/operating-system/page-replacement-algorithm/
4. https://en.wikipedia.org/wiki/Page_replacement_algorithm
5. https://afteracademy.com/blog/what-are-the-page-replacement-algorithms/
6. https://www.academia.edu/29087781/A_STUDY_OF_PAGE_REPLACEMENT_ALGORITHMS
7. https://www.researchgate.net/publication/326331469_Study_of_Page_Replacement_Algorithms_and_their_analysis_with_C

Linkedin

https://www.linkedin.com/in/ayushthakur-in/
https://www.linkedin.com/in/sonali-tandon-a2842721

Github: Ayush Thakur , Sonali Tandon