Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2023), BSc. in CSE (Day)

LAB REPORT - 04

Course Title: Operating Systems Lab

Course Code: CSE-310 **Section:** PC-201 DB

Student Details

	Name	Students Id
1.	Md. Romzan Alom	201902144

Lab Date: 26-05-2023

Submission Date: 09-06-2023

Course Teacher's Name: Jarin Tasnim Tonvi

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

Lab Report Status

Marks: ……………………………… **Signature: …………………….**

Comments: …………………………. **Date: …………………………..**

1. TITLE OF THE LAB EXPERIMENT

Implement LRU page replacement algorithm using C programming language.

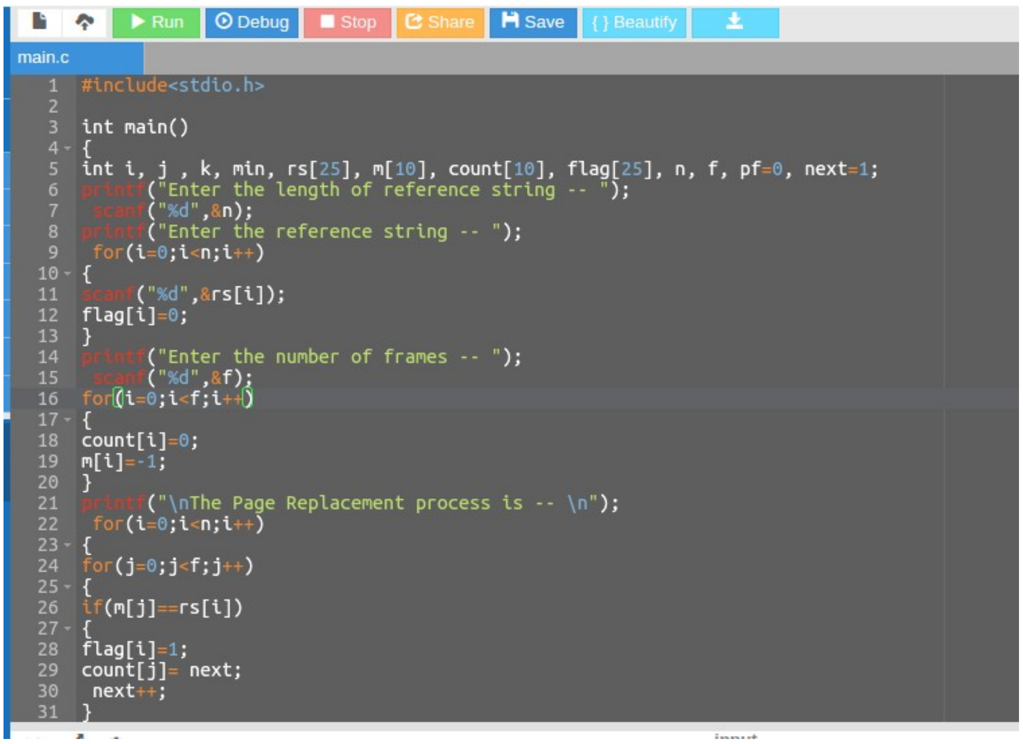
2. OBJECTIVES / AIM

- To gain a deeper understanding of the LRU page replacement algorithm and its implementation.
- To learn how to simulate system memory using arrays and queues in C programming language.
- To evaluate the performance of the LRU page replacement algorithm on a trace file of page references.
- To compare the performance of the LRU page replacement algorithm with other page replacement algorithms.
- To gain practical experience in implementing algorithms and evaluating their performance.

3. PROBLEM STATEMENT

The LRU (Least Recently Used) page replacement algorithm was implemented by creating an array and queue to simulate the main memory of the system. When a page is accessed, it is checked if it is already in memory and added to the queue. To evaluate the performance of the algorithm, a trace file was used to simulate the algorithm and record the number of page faults and execution time.

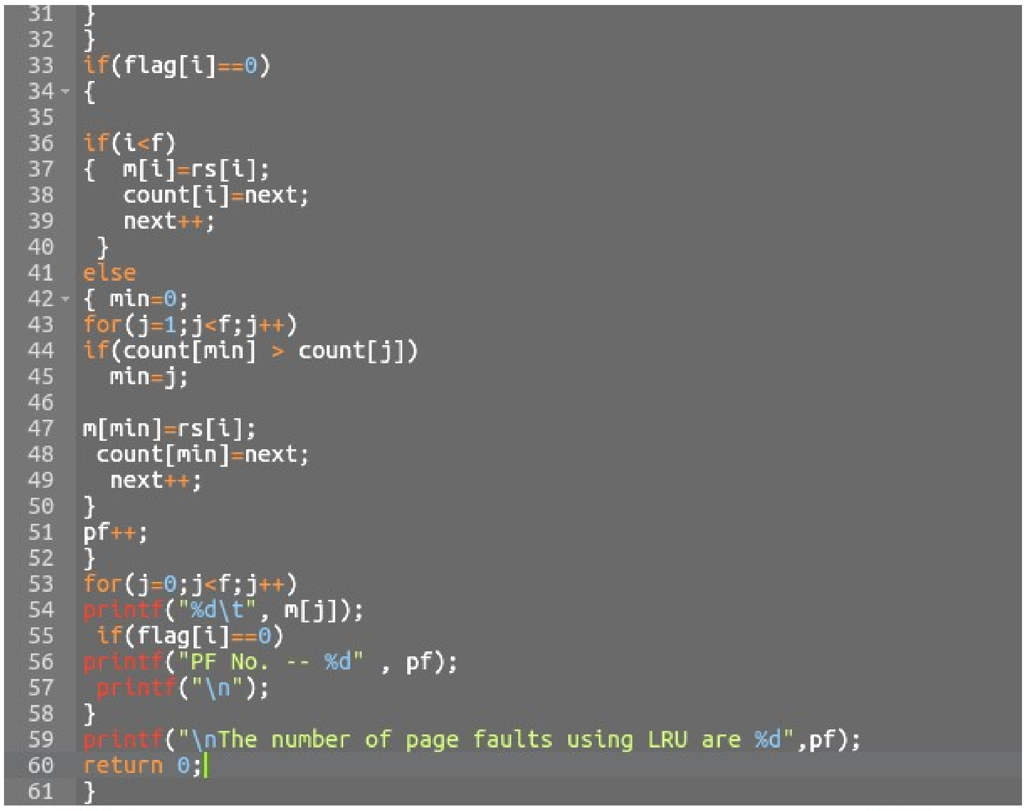
4. IMPLEMENTATION



```c
#include<stdio.h>

int main()
{
int i, j , k, min, rs[25], m[10], count[10], flag[25], n, f, pf=0, next=1;
printf("Enter the length of reference string -- ");
 scanf("%d",&n);
printf("Enter the reference string -- ");
 for(i=0;i<n;i++)
{
scanf("%d",&rs[i]);
flag[i]=0;
}
printf("Enter the number of frames -- ");
 scanf("%d",&f);
for(i=0;i<f;i++)
{
count[i]=0;
m[i]=-1;
}
printf("\nThe Page Replacement process is -- \n");
 for(i=0;i<n;i++)
{
for(j=0;j<f;j++)
{
if(m[j]==rs[i])
{
flag[i]=1;
count[j]= next;
 next++;
}
```

Figure_01: Code of LRU part-1



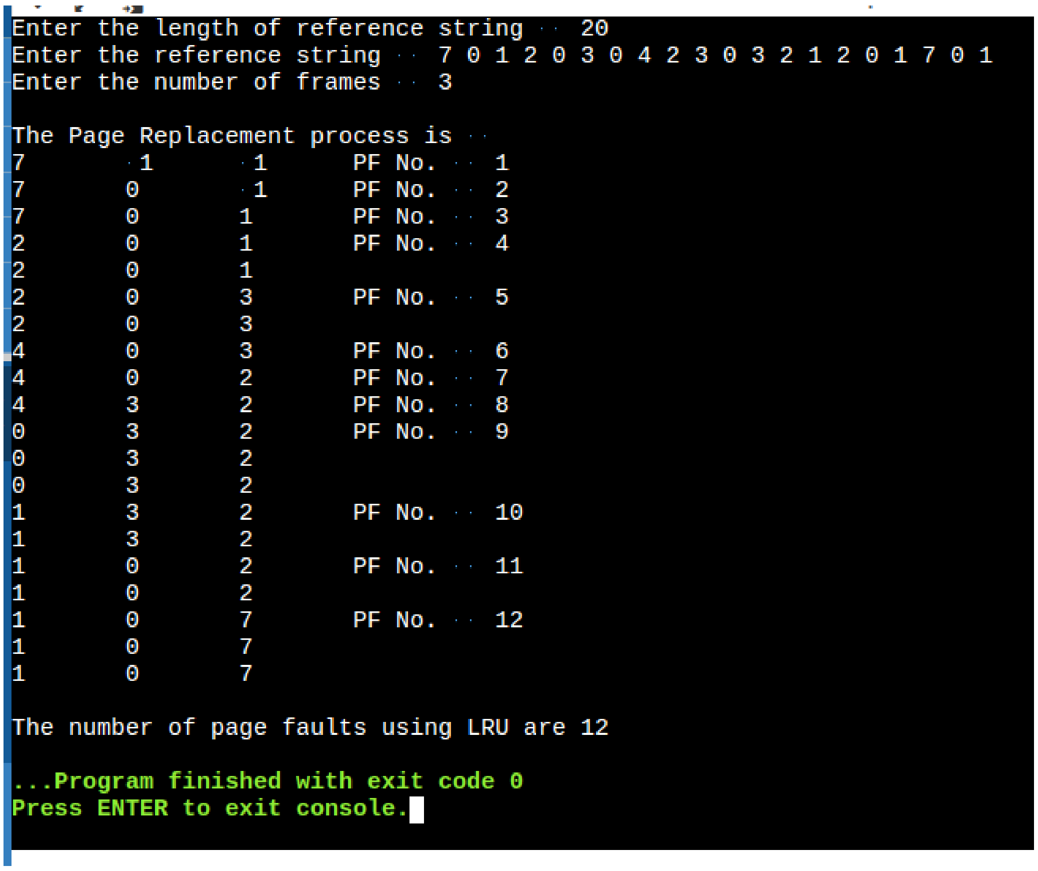
```c
 }
}
if(flag[i]==0)
{

if(i<f)
{  m[i]=rs[i];
   count[i]=next;
   next++;
 }
else
{ min=0;
for(j=1;j<f;j++)
if(count[min] > count[j])
  min=j;

m[min]=rs[i];
 count[min]=next;
  next++;
}
pf++;
}
for(j=0;j<f;j++)
printf("%d\t", m[j]);
 if(flag[i]==0)
printf("PF No. -- %d" , pf);
 printf("\n");
}
printf("\nThe number of page faults using LRU are %d",pf);
return 0;
}
```

Figure_02: Code of LRU part-2

5. TEST RESULT



```
Enter the length of reference string -- 20
Enter the reference string -- 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 --
7        1        1        PF No. -- 1
7        0        1        PF No. -- 2
7        0        1        PF No. -- 3
2        0        1        PF No. -- 4
2        0        1
2        0        3        PF No. -- 5
2        0        3
4        0        3        PF No. -- 6
4        0        2        PF No. -- 7
4        3        2        PF No. -- 8
0        3        2        PF No. -- 9
0        3        2
0        3        2
1        3        2        PF No. -- 10
1        3        2
1        0        2        PF No. -- 11
1        0        2
1        0        7        PF No. -- 12
1        0        7
1        0        7

The number of page faults using LRU are 12

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure_03: Input & output LRU

According to this figure here we see user can put the string length, string and frame number as input then using LRU to find the page fault as output. From this figure we can see 12 page faults are occurred.

5. ANALYSIS AND DISCUSSION

This experiment mainly based on C program. It may have some compiler error. Operating systems' memory is effectively managed using the LRU page replacement method. It makes sure that the pages that have been utilized most recently are preserved in memory, which lowers the amount of page faults and boosts system performance. To track the most recent pages that have been accessed, however, the method needs additional cost, which might affect system speed. Furthermore, the technique could not work effectively when there are several pages with close access times since it might eliminate a page that is still being used.

6. SUMMARY

In this lab, we used the C programming language to create the LRU page replacement method and assess its performance using a trace file. The algorithm worked rapidly and caused few page errors. Overall, the LRU page replacement technique, which can be implemented using the C programming language, is a good memory management strategy for operating systems.