Name : Anupam Kunwar
Reg : 19BCE1369
Week – 12

Q1.

AIM: Simulation of the page replacement algorithms and projecting the best approach as a result.

Use the number of page faults as evaluating criteria.

Sample input:

Enter the length of reference string: 20

Enter referred pages:

2,3,0,3,0,3,2,1,2,0,8,0,1,2,0,3,0,4

Sample output:

#number of page faults for Technique 1:-------

#number of page faults for Technique 2:-------

#number of page faults for Technique 3:-------

Technique--- is the best for the given scenario.

Solution :

```
#include <stdio.h>

int FIFO(int referenceString[], int pages)
{
int pageFaults = 0, m, n, s, 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 (referenceString[m] == temp[n])
{
s++;
pageFaults--;
```

```
}
}
pageFaults++;
if ((pageFaults <= frames) && (s == 0))
{
temp[m] = referenceString[m];
}
else if (s == 0)
{
temp[(pageFaults - 1) % frames] = referenceString[m];
}
}
return pageFaults;
}

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;
}

int LRU(int no_of_pages, int pages[])
{
int no_of_frames, frames[10], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;
no_of_frames = no_of_pages;
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;
}
}

return faults;
}

int Optimal(int reference_string[], int pages)
{
int frames[pages], interval[pages];
int total_frames, page_faults = 0;
int m, n, temp, flag, found;
int position, maximum_interval, previous_frame = -1;
total_frames = pages;
for (m = 0; m < total_frames; m++)
{
frames[m] = -1;
}
for (m = 0; m < pages; m++)
{
flag = 0;
for (n = 0; n < total_frames; n++)
{
if (frames[n] == reference_string[m])
```

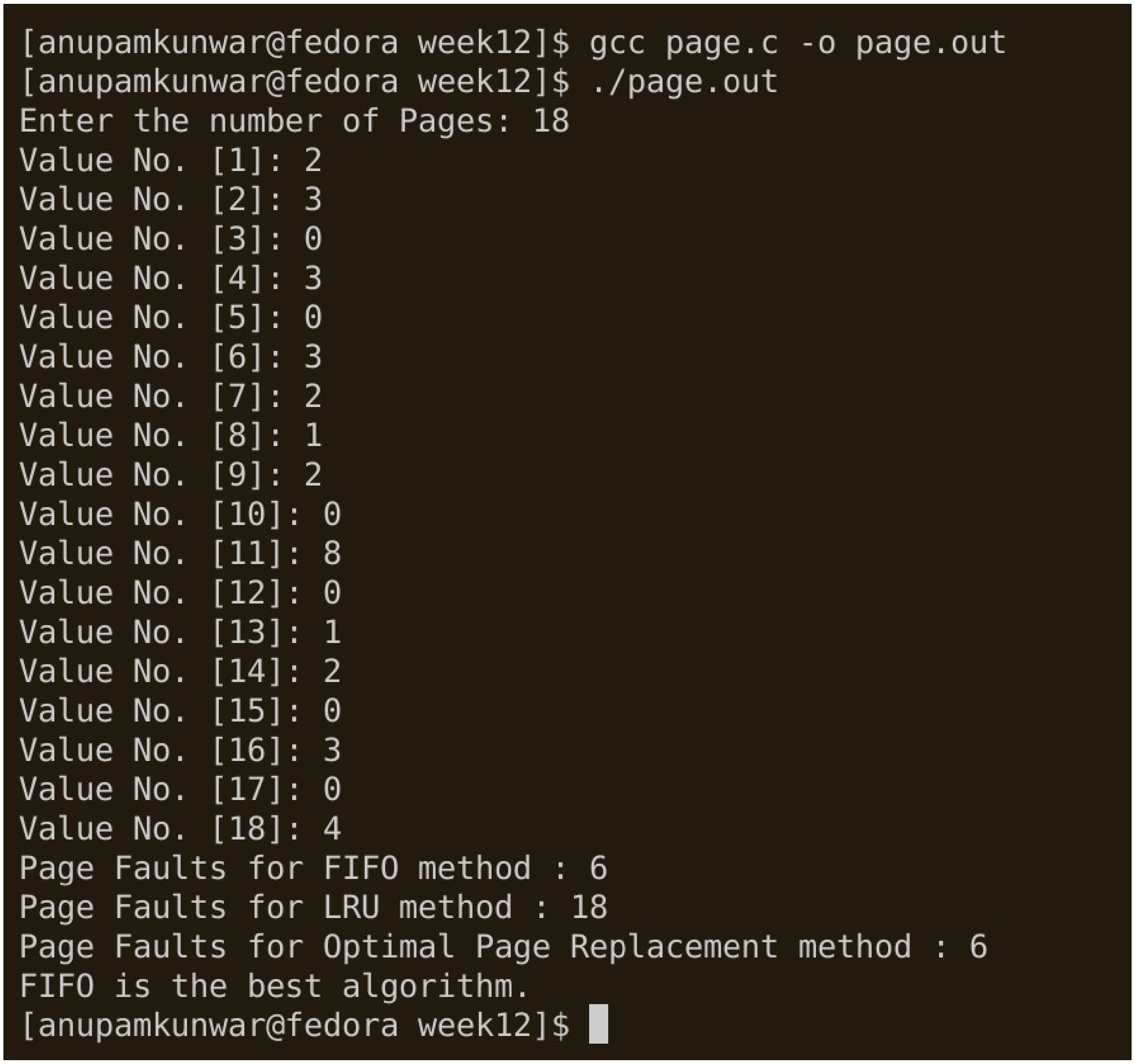
```
{
flag = 1;
break;
}
}
if (flag == 0)
{
if (previous_frame == total_frames - 1)
{
for (n = 0; n < total_frames; n++)
{
for (temp = m + 1; temp < pages; temp++)
{
interval[n] = 0;
if (frames[n] == reference_string[temp])
{
interval[n] = temp - m;
break;
}
}
}
found = 0;
for (n = 0; n < total_frames; n++)
{
if (interval[n] == 0)
{
position = n;
found = 1;
break;
}
}
}
else
{
position = ++previous_frame;
found = 1;
}
if (found == 0)
{
maximum_interval = interval[0];
position = 0;
for (n = 1; n < total_frames; n++)
{
if (maximum_interval < interval[n])
{
maximum_interval = interval[n];
position = n;
}
}
}
frames[position] = reference_string[m];
page_faults++;
}
}
return page_faults;
}

void Sort(int arr[], int n, int arr1[])
{
int i, j, min_idx,temp;
int temp1;
for (i = 0; i < n-1; i++)
{
min_idx = i;
for (j = i+1; j < n; j++)
if (arr[j] < arr[min_idx])
min_idx = j;
temp = arr[min_idx];
temp1 = arr1[min_idx];
arr[min_idx] = arr[i];
arr1[min_idx] = arr1[i];
arr[i] = temp;
arr1[i] = temp1;
}
}

int main()
{
int m, pages;
printf("Enter the number of Pages: ");
scanf("%d", &pages);
int referenceString[pages];
for (m = 0; m < pages; m++)
{
printf("Value No. [%d]: ", m + 1);
scanf("%d", &referenceString[m]);
}
int F, L, O;
F = FIFO(referenceString, pages);
L = LRU(pages, referenceString);
O = Optimal(referenceString, pages);
printf("Page Faults for FIFO method : %d\n", F);
printf("Page Faults for LRU method : %d\n", L);
printf("Page Faults for Optimal Page Replacement method : %d\n", O);
int arr[3] = {F,L,O};
int arr1[3] = {1,2,3};
Sort(arr,3,arr1);
if(arr1[0]==1)
```

```
printf("FIFO is the best algorithm.\n");
else if(arr1[0]==2)
printf("LRU algorithm is the best.\n");
else
printf("Optimal Page Replacement algorithm is best.\n");
return 0;
}
```

Output :



```
[anupamkunwar@fedora week12]$ gcc page.c -o page.out
[anupamkunwar@fedora week12]$ ./page.out
Enter the number of Pages: 18
Value No. [1]: 2
Value No. [2]: 3
Value No. [3]: 0
Value No. [4]: 3
Value No. [5]: 0
Value No. [6]: 3
Value No. [7]: 2
Value No. [8]: 1
Value No. [9]: 2
Value No. [10]: 0
Value No. [11]: 8
Value No. [12]: 0
Value No. [13]: 1
Value No. [14]: 2
Value No. [15]: 0
Value No. [16]: 3
Value No. [17]: 0
Value No. [18]: 4
Page Faults for FIFO method : 6
Page Faults for LRU method : 18
Page Faults for Optimal Page Replacement method : 6
FIFO is the best algorithm.
[anupamkunwar@fedora week12]$ 
```