# Practical – 10: To implement Page Replacement algorithm for Least Recently Used.

#include<stdio.h>

#include<limits.h>

// BE20F05F062 Akash Shridharan

int checkHit(int incomingPage, int queue[], int occupied){

for(int i = 0; i < occupied; i++){

if(incomingPage == queue[i])

return 1;

}

return 0;

}

void printFrame(int queue[], int occupied)

{

for(int i = 0; i < occupied; i++)

printf("%d\t\t\t",queue[i]);

}

int main()

{

int incomingStream[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1};

int n = sizeof(incomingStream)/sizeof(incomingStream[0]);

int frames = 3;

int queue[n];

int distance[n];

int occupied = 0;

int pagefault = 0;

printf("Page\t Frame1 \t Frame2 \t Frame3\n");

for(int i = 0;i < n; i++)

{

printf("%d: \t\t",incomingStream[i]);

if(checkHit(incomingStream[i], queue, occupied)){

printFrame(queue, occupied);

}

else if(occupied < frames){

queue[occupied] = incomingStream[i];

pagefault++;

occupied++;

printFrame(queue, occupied);

}

else{

int max = INT\_MIN;

int index;

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

{

distance[j] = 0;

for(int k = i - 1; k >= 0; k--)

{

++distance[j];

if(queue[j] == incomingStream[k])

break;

}

if(distance[j] > max){

max = distance[j];

index = j;

}

}

queue[index] = incomingStream[i];

printFrame(queue, occupied);

pagefault++;

}

printf("\n");

}

printf("Page Fault: %d",pagefault);

return 0;

}

# OUTPUT:

Page Frame1 Frame2 Frame3

7: 7

0: 7 0

1: 7 0 1

2: 2 0 1

0: 2 0 1

3: 2 0 3

0: 2 0 3

4: 4 0 3

2: 4 0 2

3: 4 3 2

0: 0 3 2

3: 0 3 2

2: 0 3 2

1: 1 3 2

Page Fault: 10