**FIFO**

#include<stdio.h>

#include<limits.h>

// This function is just use to check if the frame is containg our current page already.

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

{

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

{

if(incomingPage == queue[i])

{

return 1;

}

}

return 0;

}

// This function is used to print thr frames.

void printFrames(int queue[], int occupied)

{

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

{

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

}

}

int main()

{

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

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

int frames = 4;

int queue[n];

int distance[n];

int occupied = 0;

int pageFaults = 0;

int hit = 0;

int first\_in\_index = 0;

printf("Page:\tPage 1\t\tPage 2\t\tPage 3\t\tPage 4\n");

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

{

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

// Condition 1: When the page is already present in the frame, so dont do anything, just print the frame as it is.

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

{

hit++;

printFrames(queue, occupied);

}

// Condition 2: When the frame is not completely filled, so fill it directly and increment the pageFaults.

else if(occupied < frames)

{

queue[occupied] = incomingStream[i];

occupied++;

pageFaults++;

printFrames(queue, occupied);

}

// Condition 3: When the frame is full, and our page is not present in the frame

else

{

queue[first\_in\_index] = incomingStream[i];

// Update the first in index

first\_in\_index = (first\_in\_index+1) % frames;

printFrames(queue, occupied);

pageFaults++;

}

printf("\n");

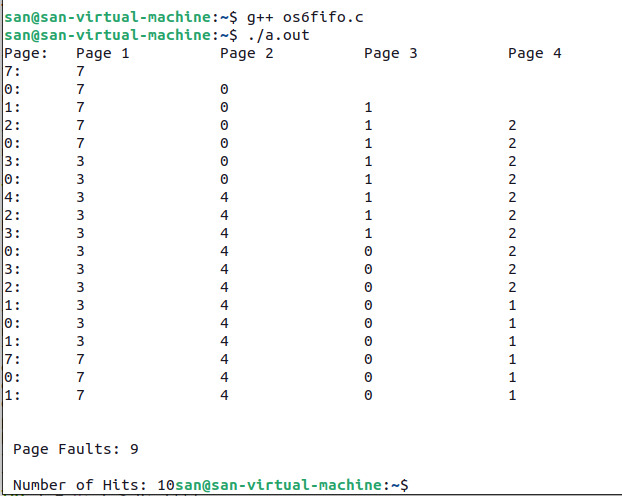
}

printf("\n\n Page Faults: %d", pageFaults);

printf("\n\n Number of Hits: %d", hit);

return 0;

}



**Optimal:**

#include <stdio.h>

#include <limits.h>

// This function checks if the current page is already in the frame.

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

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

if (incomingPage == queue[i]) {

return 1;

}

}

return 0;

}

// This function prints the frames.

void printFrames(int queue[], int occupied) {

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

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

}

}

int main() {

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

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

int frames = 4;

int queue[frames];

int distance[frames];

int occupied = 0;

int pageFaults = 0;

int hit = 0;

printf("Page:\tPage 1\t\tPage 2\t\tPage 3\t\tPage 4\n");

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

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

// Condition 1: Page is already present in the frame.

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

hit++;

printFrames(queue, occupied);

}

// Condition 2: The frame is not completely filled.

else if (occupied < frames) {

queue[occupied] = incomingStream[i];

occupied++;

pageFaults++;

printFrames(queue, occupied);

}

// Condition 3: The frame is full, and the page is not present in the frame.

else {

int max = INT\_MIN;

int index = -1;

// Calculate the future distance for each page in the frame.

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

distance[j] = INT\_MAX;

for (int k = i + 1; k < n; k++) {

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

distance[j] = k - i;

break;

}

}

// Update the index of the page with the maximum distance.

if (distance[j] > max) {

max = distance[j];

index = j;

}

}

// Replace the page at the index with the maximum future distance.

queue[index] = incomingStream[i];

printFrames(queue, occupied);

pageFaults++;

}

printf("\n");

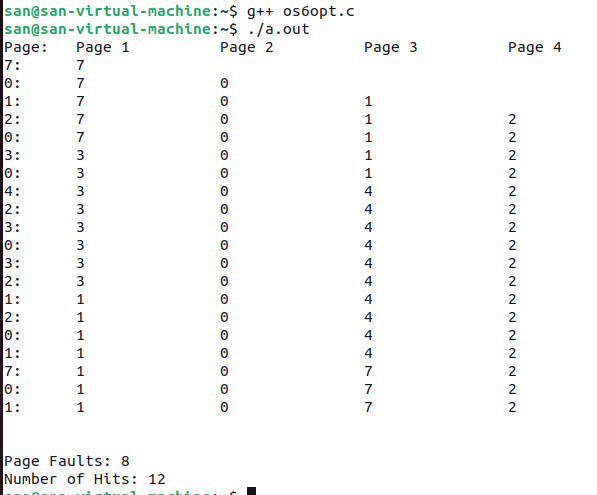
}

printf("\n\nPage Faults: %d", pageFaults);

printf("\nNumber of Hits: %d\n", hit);

return 0;

}



**LRU:**

#include<stdio.h>

#include<limits.h>

// This function is just use to check if the frame is containg our current page already.

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

{

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

{

if(incomingPage == queue[i])

{

return 1;

}

}

return 0;

}

void printFrames(int queue[], int occupied)// This function is used to print thr frames.

{

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

{

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

}

}

int main()

{

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

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

int frames = 4;

int queue[n];

int distance[n];

int occupied = 0;

int pageFaults = 0;

int hit = 0;

printf("Page:\tPage 1\t\tPage 2\t\tPage 3\t\tPage 4\n");

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

{

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

// Condition 1: When the page is already present in the frame, so dont do anything, just print the frame as it is.

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

{

hit++;

printFrames(queue, occupied);

}

// Condition 2: When the frame is not completely filled, so fill it directly and increment the pageFaults.

else if(occupied < frames)

{

queue[occupied] = incomingStream[i];

occupied++;

pageFaults++;

printFrames(queue, occupied);

}

// Condition 3: When the frame is full, and our page is not present in the frame

// So, find the max distance of each page present in the frame, then the

// page having most distance, we will replace it with our current page.

else

{

int max = INT\_MIN;

int index;

// This is to count distance value for every page in frame.

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

{

distance[j] = 0;

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

{

++distance[j];

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

break;

}

// If current page in frame distance is greater than previous, then update it.

if(distance[j] > max)

{

max = distance[j];

index = j;

}

}

// Finally, replace our page with the page in frame having most greater distance, whose index is store in the index variable and also increment the page faults.

queue[index] = incomingStream[i];

printFrames(queue, occupied);

pageFaults++;

}

printf("\n");

}

printf("\n\n Page Faults: %d", pageFaults);

printf("\n\n Number of Hits: %d", hit);

return 0;

}

