EXP 9

#include <stdio.h> #include <stdlib.h>

#define FRAME\_SIZE 3 // Number of page frames

#define PAGE\_REFERENCES 12 // Number of page references void printFrames(int frames[], int frameCount) {

for (int i = 0; i < frameCount; i++) { if (frames[i] != -1)

printf("%d ", frames[i]);

}

printf("\n");

}

// FIFO Page Replacement void fifo(int pages[], int n) {

int frames[FRAME\_SIZE];

for (int i = 0; i < FRAME\_SIZE; i++) frames[i] = -1; // Initialize frames int index = 0, pageFaults = 0;

printf("FIFO Page Replacement:\n"); for (int i = 0; i < n; i++) {

int j;

int pageFound = 0;

for (j = 0; j < FRAME\_SIZE; j++) {

if (frames[j] == pages[i]) {

pageFound = 1; // Page is already in frames break;

} }

if (!pageFound) {

frames[index] = pages[i]; // Replace page in FIFO manner index = (index + 1) % FRAME\_SIZE; // Circular increment pageFaults++;

}

printFrames(frames, FRAME\_SIZE);

}

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

}

// LRU Page Replacement void lru(int pages[], int n) {

int frames[FRAME\_SIZE];

int lastUsed[FRAME\_SIZE] = {0};

for (int i = 0; i < FRAME\_SIZE; i++) frames[i] = -1; // Initialize frames int pageFaults = 0;

printf("LRU Page Replacement:\n"); for (int i = 0; i < n; i++) {

int j;

int pageFound = 0;

for (j = 0; j < FRAME\_SIZE; j++) {

if (frames[j] == pages[i]) {

pageFound = 1; // Page is already in frames lastUsed[j] = i; // Update last used time break;

}

}

if (!pageFound) {

// Find the least recently used page to replace int lruIndex = 0;

for (j = 1; j < FRAME\_SIZE; j++) {

if (lastUsed[j] < lastUsed[lruIndex]) { lruIndex = j;

}

}

frames[lruIndex] = pages[i]; // Replace LRU page lastUsed[lruIndex] = i; // Update last used time pageFaults++;

}

printFrames(frames, FRAME\_SIZE);

}

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

}

// Optimal Page Replacement void optimal(int pages[], int n) {

int frames[FRAME\_SIZE];

for (int i = 0; i < FRAME\_SIZE; i++) frames[i] = -1; // Initialize frames int pageFaults = 0;

printf("Optimal Page Replacement:\n"); for (int i = 0; i < n; i++) {

int j;

int pageFound = 0;

for (j = 0; j < FRAME\_SIZE; j++) {

if (frames[j] == pages[i]) {

pageFound = 1; // Page is already in frames break;

}

}

if (!pageFound) {

// Find the optimal page to replace

int farthestIndex = -1, replaceIndex = -1; for (j = 0; j < FRAME\_SIZE; j++) {

int k;

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

if (frames[j] == pages[k]) { if (farthestIndex < k) { farthestIndex = k; replaceIndex = j;

}

break;

}

}

if (k == n) { // If page not found in future replaceIndex = j;

break;

}

}

frames[replaceIndex] = pages[i]; // Replace optimal page pageFaults++;

}

printFrames(frames, FRAME\_SIZE);

}

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

}

int main() {

int pages[PAGE\_REFERENCES] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3}; // Sample page reference string

fifo(pages, PAGE\_REFERENCES); lru(pages, PAGE\_REFERENCES); optimal(pages, PAGE\_REFERENCES);

return 0;

}

