**NAME : MARIYAM PATEL**

**SRN:202201178**

**ROLL NO:34**

**ASSIGNMENT-8**

Q. Simulate the following page replacement algorithms-  
a. FIFO b. LRU c. OPT

CODE-

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <limits.h>

// Function to check if a page exists in frame

bool isPagePresent(int page, int\* frames, int frameCount) {

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

if (frames[i] == page)

return true;

}

return false;

}

// Function to find the position of the first occurrence of a page in future

int findOptimalPosition(int\* pages, int pageCount, int\* frames, int frameCount, int currentPos) {

int farthest = -1, replaceIndex = 0;

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

int j;

for (j = currentPos + 1; j < pageCount; j++) {

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

if (j > farthest) {

farthest = j;

replaceIndex = i;

}

break;

}

}

if (j == pageCount)

return i;

}

return (farthest == -1) ? 0 : replaceIndex;

}

// FIFO Page Replacement Algorithm

void fifo(int\* pages, int pageCount, int frameCount) {

int\* frames = (int\*)calloc(frameCount, sizeof(int));

int pageFaults = 0;

int currentIndex = 0;

printf("\nFIFO Page Replacement:\n");

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

printf("\nReference to page %d: ", pages[i]);

if (!isPagePresent(pages[i], frames, frameCount)) {

frames[currentIndex] = pages[i];

currentIndex = (currentIndex + 1) % frameCount;

pageFaults++;

printf("Page Fault! Frames: ");

} else {

printf("No Page Fault. Frames: ");

}

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

if (frames[j] != 0)

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

else

printf("- ");

}

}

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

free(frames);

}

// LRU Page Replacement Algorithm

void lru(int\* pages, int pageCount, int frameCount) {

int\* frames = (int\*)calloc(frameCount, sizeof(int));

int\* lastUsed = (int\*)calloc(frameCount, sizeof(int));

int pageFaults = 0;

printf("\nLRU Page Replacement:\n");

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

printf("\nReference to page %d: ", pages[i]);

if (!isPagePresent(pages[i], frames, frameCount)) {

int replaceIndex = 0;

int leastUsed = INT\_MAX;

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

if (frames[j] == 0) {

replaceIndex = j;

break;

}

if (lastUsed[j] < leastUsed) {

leastUsed = lastUsed[j];

replaceIndex = j;

}

}

frames[replaceIndex] = pages[i];

lastUsed[replaceIndex] = i;

pageFaults++;

printf("Page Fault! Frames: ");

} else {

printf("No Page Fault. Frames: ");

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

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

lastUsed[j] = i;

}

}

}

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

if (frames[j] != 0)

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

else

printf("- ");

}

}

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

free(frames);

free(lastUsed);

}

// Optimal Page Replacement Algorithm

void optimal(int\* pages, int pageCount, int frameCount) {

int\* frames = (int\*)calloc(frameCount, sizeof(int));

int pageFaults = 0;

printf("\nOptimal Page Replacement:\n");

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

printf("\nReference to page %d: ", pages[i]);

if (!isPagePresent(pages[i], frames, frameCount)) {

int replaceIndex;

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

if (frames[j] == 0) {

replaceIndex = j;

break;

}

}

if (frames[frameCount - 1] != 0) {

replaceIndex = findOptimalPosition(pages, pageCount, frames, frameCount, i);

}

frames[replaceIndex] = pages[i];

pageFaults++;

printf("Page Fault! Frames: ");

} else {

printf("No Page Fault. Frames: ");

}

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

if (frames[j] != 0)

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

else

printf("- ");

}

}

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

free(frames);

}

int main() {

int pageCount, frameCount;

printf("Enter the number of pages: ");

scanf("%d", &pageCount);

printf("Enter the number of frames: ");

scanf("%d", &frameCount);

int\* pages = (int\*)malloc(pageCount \* sizeof(int));

printf("Enter the page reference string: ");

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

scanf("%d", &pages[i]);

}

fifo(pages, pageCount, frameCount);

lru(pages, pageCount, frameCount);

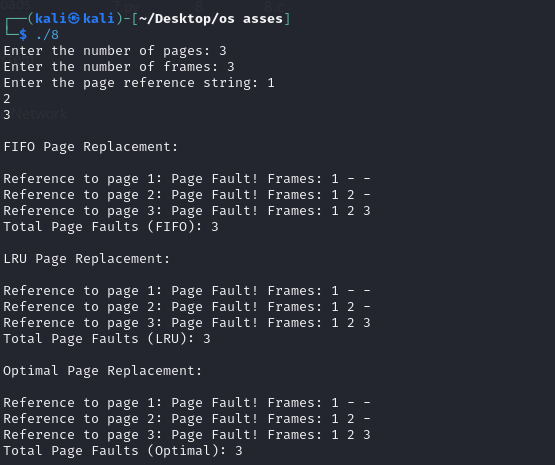
optimal(pages, pageCount, frameCount);

free(pages);

return 0;

}

OUTPUT-

****

**FIFO-**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <stdbool.h>**

**// Function to check if a page exists in frame**

**bool isPagePresent(int pg, int\* fs, int fC) {**

**int i;**

**for ( i = 0; i < fC; i++) {**

**if (fs[i] == pg)**

**return true;**

**}**

**return false;**

**}**

**// FIFO Page Replacement Algorithm**

**void fifo(int\* pgs, int pC, int fC) {**

**int i,j;**

**int\* fs = (int\*)calloc(fC, sizeof(int));**

**int pF = 0;**

**int currentIndex = 0;**

**printf("\nFIFO Page Replacement:\n");**

**for ( i = 0; i < pC; i++) {**

**printf("\nReference to page %d: ", pgs[i]);**

**if (!isPagePresent(pgs[i], fs, fC)) {**

**fs[currentIndex] = pgs[i];**

**currentIndex = (currentIndex + 1) % fC;**

**pF++;**

**printf("Page Fault! Frames: ");**

**} else {**

**printf("No Page Fault. Frames: ");**

**}**

**for ( j = 0; j < fC; j++) {**

**if (fs[j] != 0)**

**printf("%d ", fs[j]);**

**else**

**printf("- ");**

**}**

**}**

**printf("\nTotal Page Faults (FIFO): %d\n", pF);**

**free(fs);**

**}**

**int main() {**

**int i;**

**int pC, fC;**

**printf("Enter the number of pages: ");**

**scanf("%d", &pC);**

**printf("Enter the number of frames: ");**

**scanf("%d", &fC);**

**int\* pgs = (int\*)malloc(pC \* sizeof(int));**

**printf("Enter the page reference string: ");**

**for ( i = 0; i < pC; i++) {**

**scanf("%d", &pgs[i]);**

**}**

**fifo(pgs, pC, fC);**

**free(pgs);**

**return 0;**

**}**

**LRU-**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <stdbool.h>**

**#include <limits.h>**

**// Function to check if a page exists in frame**

**bool isPagePresent(int page, int\* frames, int frameCount) {**

**int i;**

**for ( i = 0; i < frameCount; i++) {**

**if (frames[i] == page)**

**return true;**

**}**

**return false;**

**}**

**// LRU Page Replacement Algorithm**

**void lru(int\* pages, int pageCount, int frameCount) {**

**int i,j;**

**int\* frames = (int\*)calloc(frameCount, sizeof(int));**

**int\* lastUsed = (int\*)calloc(frameCount, sizeof(int));**

**int pageFaults = 0;**

**printf("\nLRU Page Replacement:\n");**

**for ( i = 0; i < pageCount; i++) {**

**printf("\nReference to page %d: ", pages[i]);**

**if (!isPagePresent(pages[i], frames, frameCount)) {**

**int replaceIndex = 0;**

**int leastUsed = INT\_MAX;**

**for ( j = 0; j < frameCount; j++) {**

**if (frames[j] == 0) {**

**replaceIndex = j;**

**break;**

**}**

**if (lastUsed[j] < leastUsed) {**

**leastUsed = lastUsed[j];**

**replaceIndex = j;**

**}**

**}**

**frames[replaceIndex] = pages[i];**

**lastUsed[replaceIndex] = i;**

**pageFaults++;**

**printf("Page Fault! Frames: ");**

**} else {**

**printf("No Page Fault. Frames: ");**

**for ( j = 0; j < frameCount; j++) {**

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

**lastUsed[j] = i;**

**}**

**}**

**}**

**for ( j = 0; j < frameCount; j++) {**

**if (frames[j] != 0)**

**printf("%d ", frames[j]);**

**else**

**printf("- ");**

**}**

**}**

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

**free(frames);**

**free(lastUsed);**

**}**

**int main() {**

**int i;**

**int pageCount, frameCount;**

**printf("Enter the number of pages: ");**

**scanf("%d", &pageCount);**

**printf("Enter the number of frames: ");**

**scanf("%d", &frameCount);**

**int\* pages = (int\*)malloc(pageCount \* sizeof(int));**

**printf("Enter the page reference string: ");**

**for ( i = 0; i < pageCount; i++) {**

**scanf("%d", &pages[i]);**

**}**

**lru(pages, pageCount, frameCount);**

**free(pages);**

**return 0;**

**}**

**OPT-**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <stdbool.h>**

**// Function to check if a page exists in frame**

**bool isPagePresent(int page, int\* frames, int frameCount) {**

**int i;**

**for ( i = 0; i < frameCount; i++) {**

**if (frames[i] == page)**

**return true;**

**}**

**return false;**

**}**

**// Function to find the position of the first occurrence of a page in future**

**int findOptimalPosition(int\* pages, int pageCount, int\* frames, int frameCount, int currentPos) {**

**int i,j;**

**int farthest = -1, replaceIndex = 0;**

**for ( i = 0; i < frameCount; i++) {**

**int j;**

**for (j = currentPos + 1; j < pageCount; j++) {**

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

**if (j > farthest) {**

**farthest = j;**

**replaceIndex = i;**

**}**

**break;**

**}**

**}**

**if (j == pageCount)**

**return i;**

**}**

**return (farthest == -1) ? 0 : replaceIndex;**

**}**

**// Optimal Page Replacement Algorithm**

**void optimal(int\* pages, int pageCount, int frameCount) {**

**int i,j;**

**int\* frames = (int\*)calloc(frameCount, sizeof(int));**

**int pageFaults = 0;**

**printf("\nOptimal Page Replacement:\n");**

**for ( i = 0; i < pageCount; i++) {**

**printf("\nReference to page %d: ", pages[i]);**

**if (!isPagePresent(pages[i], frames, frameCount)) {**

**int replaceIndex;**

**for ( j = 0; j < frameCount; j++) {**

**if (frames[j] == 0) {**

**replaceIndex = j;**

**break;**

**}**

**}**

**if (frames[frameCount - 1] != 0) {**

**replaceIndex = findOptimalPosition(pages, pageCount, frames, frameCount, i);**

**}**

**frames[replaceIndex] = pages[i];**

**pageFaults++;**

**printf("Page Fault! Frames: ");**

**} else {**

**printf("No Page Fault. Frames: ");**

**}**

**for ( j = 0; j < frameCount; j++) {**

**if (frames[j] != 0)**

**printf("%d ", frames[j]);**

**else**

**printf("- ");**

**}**

**}**

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

**free(frames);**

**}**

**int main() {**

**int pageCount, frameCount;**

**int i;**

**printf("Enter the number of pages: ");**

**scanf("%d", &pageCount);**

**printf("Enter the number of frames: ");**

**scanf("%d", &frameCount);**

**int\* pages = (int\*)malloc(pageCount \* sizeof(int));**

**printf("Enter the page reference string: ");**

**for ( i = 0; i < pageCount; i++) {**

**scanf("%d", &pages[i]);**

**}**

**optimal(pages, pageCount, frameCount);**

**free(pages);**

**return 0;**

**}**