LAB-8

**Write a C program to simulate page replacement algorithms:**

**(a) FIFO**

**(b) LRU**

**(c) Optimal**

CODE:

#include <stdio.h>

int n, f, i, j, k;

int in[100];

int p[50];

int hit = 0;

int pgfaultcnt = 0;

void getData() {

printf("\nEnter length of page reference sequence:");

scanf("%d", &n);

printf("\nEnter the page reference sequence:");

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

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

printf("\nEnter no of frames:");

scanf("%d", &f);

}

void initialize() {

pgfaultcnt = 0;

for (i = 0; i < f; i++)

p[i] = 9999;

}

int isHit(int data) {

hit = 0;

for (j = 0; j < f; j++) {

if (p[j] == data) {

hit = 1;

break;

}

}

return hit;

}

int getHitIndex(int data) {

int hitind;

for (k = 0; k < f; k++) {

if (p[k] == data) {

hitind = k;

break;

}

}

return hitind;

}

void dispPages() {

for (k = 0; k < f; k++) {

if (p[k] != 9999)

printf(" %d", p[k]);

}

}

void dispPgFaultCnt() {

printf("\nTotal no of page faults:%d", pgfaultcnt);

}

void fifo() {

getData();

initialize();

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

printf("\nFor %d :", in[i]);

if (isHit(in[i]) == 0) {

for (k = 0; k < f - 1; k++)

p[k] = p[k + 1];

p[k] = in[i];

pgfaultcnt++;

dispPages();

} else

printf("No page fault");

}

dispPgFaultCnt();

}

void optimal() {

initialize();

int near[50];

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

printf("\nFor %d :", in[i]);

if (isHit(in[i]) == 0) {

for (j = 0; j < f; j++) {

int pg = p[j];

int found = 0;

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

if (pg == in[k]) {

near[j] = k;

found = 1;

break;

} else

found = 0;

}

if (!found)

near[j] = 9999;

}

int max = -9999;

int repindex;

for (j = 0; j < f; j++) {

if (near[j] > max) {

max = near[j];

repindex = j;

}

}

p[repindex] = in[i];

pgfaultcnt++;

dispPages();

} else

printf("No page fault");

}

dispPgFaultCnt();

}

void lru() {

initialize();

int least[50];

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

printf("\nFor %d :", in[i]);

if (isHit(in[i]) == 0) {

for (j = 0; j < f; j++) {

int pg = p[j];

int found = 0;

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

if (pg == in[k]) {

least[j] = k;

found = 1;

break;

} else

found = 0;

}

if (!found)

least[j] = -9999;

}

int min = 9999;

int repindex;

for (j = 0; j < f; j++) {

if (least[j] < min) {

min = least[j];

repindex = j;

}

}

p[repindex] = in[i];

pgfaultcnt++;

dispPages();

} else

printf("No page fault!");

}

dispPgFaultCnt();

}

int main() {

int choice;

while (1) {

printf("\nPage Replacement Algorithms\n1.Enter data\n2.FIFO\n3.Optimal\n4.LRU\n5.Exit\nEnter your choice:");

scanf("%d", &choice);

switch (choice) {

case 1: getData();

break;

case 2: fifo();

break;

case 3: optimal();

break;

case 4: lru();

break;

default: return 0;

break;

}

}

}

OUTPUT:



