**LAB-8**

**Page Replacement Algorithm**

1. Write a C program to simulate page replacement algorithms:

(a) FIFO

(b) LRU

(c) Optimal

#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("Enter the page reference sequence: ");

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

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

printf("Enter number of frames: ");

scanf("%d", &f);

}

void initialize() {

pgfaultcnt = 0;

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

p[i] = -1;

}

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) {

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

if (p[k] == data)

return k;

}

return -1;

}

void dispPages() {

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

if (p[k] != -1)

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

}

printf("\n");

}

void dispPgFaultCnt() {

printf("\nTotal number of page faults: %d\n", pgfaultcnt);

}

void fifo() {

initialize();

int index = 0;

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

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

if (!isHit(in[i])) {

p[index] = in[i];

index = (index + 1) % f;

pgfaultcnt++;

dispPages();

} else {

printf("No page fault\n");

}

}

dispPgFaultCnt();

}

void optimal() {

initialize();

int near[50];

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

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

if (!isHit(in[i])) {

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

int pg = p[j];

int found = 0;

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

if (pg == in[k]) {

near[j] = k;

found = 1;

break;

}

}

if (!found)

near[j] = 9999;

}

int max = -1, repindex = -1;

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\n");

}

}

dispPgFaultCnt();

}

void lru() {

initialize();

int least[50];

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

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

if (!isHit(in[i])) {

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;

}

}

if (!found)

least[j] = -9999;

}

int min = 9999, repindex = -1;

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!\n");

}

}

dispPgFaultCnt();

}

int main() {

int choice;

while (1) {

printf("\nPage Replacement Algorithms\n");

printf("1. 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;

case 5: return 0;

default: printf("Invalid choice!\n"); break;

}

}

}

OUTPUT:



