

**EXP NO:** 11c) **OPTIMAL PAGE REPLACEMENT**

**DATE:29/4/25**

**PROGRAM:**

#include <stdio.h>

#define MAX 10

int findOptimalPage(int frames[], int n, int pages[], int currentIndex, int capacity) {

int farthest = currentIndex;

int optimalPage = -1;

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

int j;

for (j = currentIndex + 1; j < n; j++) {

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

if (j > farthest) {

farthest = j;

optimalPage = i;

}

break;

}

}

if (j == n) {

return i; // If the page is not found, return this frame

}

}

return optimalPage;

}

void optimalPageReplacement(int pages[], int n, int capacity) {

int frames[capacity], pageFaults = 0;

// Initialize frames to -1 (empty)

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

frames[i] = -1;

}

printf("Page Reference String: ");

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

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

}

printf("\n");

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

int found = 0;

// Check if the page is already in one of the frames

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

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

found = 1;

break;

}

}

// If page is not found, replace the optimal page

if (!found) {

int optimalPage = findOptimalPage(frames, n, pages, i, capacity);

frames[optimalPage] = pages[i];

pageFaults++;

// Display the frames after each page reference

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

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

}

printf("\n");

}

}

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

}

int main() {

int pages[MAX], n, capacity;

printf("Enter number of frames: ");

scanf("%d", &capacity);

printf("Enter number of pages: ");

scanf("%d", &n);

printf("Enter reference string: ");

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

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

}

optimalPageReplacement(pages, n, capacity);

return 0;

}

**OUTPUT:**

Enter number of frames: 3

Enter number of pages: 6

Enter reference string: 2

3

4

3

4

8

Page Reference String: 2 3 4 3 4 8

2 -1 -1

3 -1 -1

3 4 -1

8 4 -1

Total Page Faults = 4