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

**a) FIFO b) LRU c) Optimal**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX\_FRAMES 10

#define MAX\_PAGES 25

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

    int frame[MAX\_FRAMES], frameCount = 0, pageFaults = 0, frameIndex = 0;

    bool isPagePresent = false;

    for (int i = 0; i < n; i++) {

        isPagePresent = false;

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

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

                isPagePresent = true;

                break;

            }

        }

        if (isPagePresent == false) {

            if (frameCount < capacity) {

                frame[frameCount] = pages[i];

                frameCount++;

            } else {

                frame[frameIndex] = pages[i];

                frameIndex++;

                if (frameIndex >= capacity)

                    frameIndex = 0;

            }

            pageFaults++;

        }

    }

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

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

}

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

    int frame[MAX\_FRAMES], frameCount = 0, pageFaults = 0, counter[MAX\_FRAMES];

    bool isPagePresent = false;

    for (int i = 0; i < n; i++) {

        isPagePresent = false;

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

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

                isPagePresent = true;

                counter[j] = i;

                break;

            }

        }

        if (isPagePresent == false) {

            if (frameCount < capacity) {

                frame[frameCount] = pages[i];

                counter[frameCount] = i;

                frameCount++;

            } else {

                int lru = 0;

                for (int j = 1; j < capacity; j++) {

                    if (counter[j] < counter[lru])

                        lru = j;

                }

                frame[lru] = pages[i];

                counter[lru] = i;

            }

            pageFaults++;

        }

    }

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

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

}

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

    int frame[MAX\_FRAMES], frameCount = 0, pageFaults = 0;

    bool isPagePresent = false;

    for (int i = 0; i < n; i++) {

        isPagePresent = false;

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

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

                isPagePresent = true;

                break;

            }

        }

        if (isPagePresent == false) {

            if (frameCount < capacity) {

                frame[frameCount] = pages[i];

                frameCount++;

            } else {

                int future[MAX\_FRAMES] = {0};

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

                    bool isFound = false;

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

                        if (pages[k] == frame[j]) {

                            future[j] = k;

                            isFound = true;

                            break;

                        }

                    }

                    if (isFound == false)

                        future[j] = n + 1;

                }

                int longest = 0;

                for (int j = 1; j < frameCount; j++) {

                    if (future[j] > future[longest])

                        longest = j;

                }

                frame[longest] = pages[i];

            }

            pageFaults++;

        }

    }

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

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

}

int main() {

    int pages[MAX\_PAGES], n, capacity;

    printf("Page Replacement Algorithms\n");

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

    scanf("%d", &n);

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

    for (int i = 0; i < n; i++) {

        printf("Page %d: ", i + 1);

        scanf("%d", &pages[i]);

    }

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

    scanf("%d", &capacity);

    fifo(pages, n, capacity);

    lru(pages, n, capacity);

    optimal(pages, n, capacity);

    return 0;

}

OUTPUT:

