Program-7

Write a C program to simulate page replacement algorithms. a) FIFO b) LRU c) Optimal

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_PAGES 10
#define MAX_FRAMES 3

// Function to simulate FIFO page replacement algorithm
void FIFO(int pages[], int n, int frames) {
   int frame[frames];
   for (int i = 0; i < frames; i++) {
      frame[i] = -1; // Initialize all frames as empty
   }

   int pageFaults = 0, pointer = 0;

   for (int i = 0; i < n; i++) {
      int page = pages[i];
      int found = 0;

      // Check if the page is already in the frame
      for (int j = 0; j < frames; j++) {
         if (frame[j] == page) {
            found = 1;
            break;
         }
      }

      // If page is not found in frame, replace the page using FIFO
      if (!found) {
         frame[pointer] = page;
         pointer = (pointer + 1) % frames; // Move to the next frame
         pageFaults++;
      }

      printf("Frames: ");
      for (int k = 0; k < frames; k++) {
         if (frame[k] != -1)
            printf("%d ", frame[k]);
```

```
        }
        printf("\n");
    }

    printf("Total page faults using FIFO: %d\n", pageFaults);
}

// Function to simulate LRU page replacement algorithm
void LRU(int pages[], int n, int frames) {
    int frame[frames];
    int lastUsed[frames];
    for (int i = 0; i < frames; i++) {
        frame[i] = -1;  // Initialize all frames as empty
        lastUsed[i] = -1; // Initialize last used time
    }

    int pageFaults = 0;

    for (int i = 0; i < n; i++) {
        int page = pages[i];
        int found = 0;
        int leastRecent = 0;

        // Check if the page is already in the frame
        for (int j = 0; j < frames; j++) {
            if (frame[j] == page) {
                found = 1;
                lastUsed[j] = i;  // Update the last used time
                break;
            }
        }

        // If page is not found in frame, replace the page using LRU
        if (!found) {
            // Find the least recently used page
            for (int j = 1; j < frames; j++) {
                if (lastUsed[j] < lastUsed[leastRecent]) {
                    leastRecent = j;
                }
            }
```

```
            frame[leastRecent] = page;
            lastUsed[leastRecent] = i;
            pageFaults++;
        }

        printf("Frames: ");
        for (int k = 0; k < frames; k++) {
            if (frame[k] != -1)
                printf("%d ", frame[k]);
        }
        printf("\n");
    }

    printf("Total page faults using LRU: %d\n", pageFaults);
}

// Function to simulate Optimal page replacement algorithm
void Optimal(int pages[], int n, int frames) {
    int frame[frames];
    for (int i = 0; i < frames; i++) {
        frame[i] = -1;  // Initialize all frames as empty
    }

    int pageFaults = 0;

    for (int i = 0; i < n; i++) {
        int page = pages[i];
        int found = 0;

        // Check if the page is already in the frame
        for (int j = 0; j < frames; j++) {
            if (frame[j] == page) {
                found = 1;
                break;
            }
        }

        // If page is not found in frame, replace the page using Optimal
        if (!found) {
            int farthest = -1;
            int replaceIndex = -1;
```

```
            // Find the page that will not be used for the longest time
            for (int j = 0; j < frames; j++) {
                int k;
                for (k = i + 1; k < n; k++) {
                    if (frame[j] == pages[k]) {
                        break;
                    }
                }

                if (k == n) {
                    replaceIndex = j;
                    break;
                }

                if (k > farthest) {
                    farthest = k;
                    replaceIndex = j;
                }
            }

            frame[replaceIndex] = page;
            pageFaults++;
        }

        printf("Frames: ");
        for (int k = 0; k < frames; k++) {
            if (frame[k] != -1)
                printf("%d ", frame[k]);
        }
        printf("\n");
    }

    printf("Total page faults using Optimal: %d\n", pageFaults);
}

int main() {
    int pages[MAX_PAGES] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3}; // Test page reference string
    int n = 10; // Number of pages in the reference string
    int frames = 3; // Number of frames in the page table
```

```
    printf("FIFO Page Replacement:\n");
    FIFO(pages, n, frames);

    printf("\nLRU Page Replacement:\n");
    LRU(pages, n, frames);

    printf("\nOptimal Page Replacement:\n");
    Optimal(pages, n, frames);

    return 0;
}
```

Output: