Page Replacement Algorithm 33185 OS

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

#define MAX 20

bool isPageInFrame(int frames[], int n, int page)
{
        for (int i = 0; i < n; i++)
        {
        if (frames[i] == page)
        return true;
        }
        return false;
}

int FCFS(int pages[], int numPages, int frames[], int numFrames)
{
        int pageFaults = 0, pointer = 0;

        for (int i = 0; i < numPages; i++) {
        if (!isPageInFrame(frames, numFrames, pages[i]))
        {
        frames[pointer] = pages[i];
        pointer = (pointer + 1) % numFrames;
        pageFaults++;
        }
        }

        return pageFaults;
}

int LRU(int pages[], int numPages, int frames[], int numFrames)
{
        int pageFaults = 0, leastRecentlyUsed[MAX], time = 0;

        // Initialize leastRecentlyUsed
        for (int i = 0; i < numFrames; i++) {
        leastRecentlyUsed[i] = -1; // Not used
        }

        for (int i = 0; i < numPages; i++)
        {
        time++;
        if (!isPageInFrame(frames, numFrames, pages[i]))
        {
        int lru = 0;
        for (int j = 1; j < numFrames; j++)
        {
                if (leastRecentlyUsed[j] < leastRecentlyUsed[lru])
                lru = j;
        }

        frames[lru] = pages[i];
        leastRecentlyUsed[lru] = time; // Update LRU time
        pageFaults++;
        }
        else
        {
        for (int j = 0; j < numFrames; j++)
        {
                if (frames[j] == pages[i])
                {
                leastRecentlyUsed[j] = time; // Update LRU time
                }
        }
        }
        }

        return pageFaults;
}

int Optimal(int pages[], int numPages, int frames[], int numFrames)
{
        int pageFaults = 0;

        for (int i = 0; i < numPages; i++)
        {
        if (!isPageInFrame(frames, numFrames, pages[i]))
        {
        int farthest = -1, replace = 0;

        for (int j = 0; j < numFrames; j++)
        {
                int k;
                for (k = i + 1; k < numPages; k++)
                {
```

```
                if (frames[j] == pages[k]) break;
                }

                if (k > farthest || k == numPages)
                {
                farthest = k;
                replace = j;
                }
        }

        frames[replace] = pages[i];
        pageFaults++;
        }
        }

        return pageFaults;
}

int main() {
        int pages[MAX], frames[MAX];
        int numPages, numFrames;

        printf("Enter number of pages: ");
        scanf("%d", &numPages);
        printf("Enter page reference sequence: ");
        for (int i = 0; i < numPages; i++)
        {
        scanf("%d", &pages[i]);
        }

        printf("Enter number of frames (at least 3): ");
        scanf("%d", &numFrames);

        if (numFrames < 3) {
        printf("Number of frames should be at least 3.\n");
        return 1;
        }

        for (int i = 0; i < numFrames; i++) frames[i] = -1;
        printf("FCFS Page Faults: %d\n", FCFS(pages, numPages, frames, numFrames));

        for (int i = 0; i < numFrames; i++) frames[i] = -1;
        printf("LRU Page Faults: %d\n", LRU(pages, numPages, frames, numFrames));
```

```
    for (int i = 0; i < numFrames; i++) frames[i] = -1;
    printf("Optimal Page Faults: %d\n", Optimal(pages, numPages, frames, numFrames));

    return 0;
}
```

OUTPUT:

Testcase 1

```
Enter number of pages: 9
Enter page reference sequence: 1 3 0 3 5 6 3 2 1
Enter number of frames (at least 3): 3
FCFS Page Faults: 8
LRU Page Faults: 7
Optimal Page Faults: 6
```

Testcase 2

```
Enter number of pages: 12
Enter page reference sequence: 7 0 1 2 0 3 0 4 2 3 0 3
Enter number of frames (at least 3): 3
FCFS Page Faults: 10
LRU Page Faults: 9
Optimal Page Faults: 7
```

TestCase 3

```
Enter number of pages: 8
Enter page reference sequence: 2 3 2 1 4 5 7 0
Enter number of frames (at least 3): 4
FCFS Page Faults: 7
LRU Page Faults: 7
Optimal Page Faults: 7
```