LAB PROGRAM 10

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

INPUT-

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_FRAMES 10
#define MAX_PAGES 100

void printFrames(int frames[], int size) {
    for (int i = 0; i < size; i++) {
        if (frames[i] == -1)
            printf("- ");
        else
            printf("%d ", frames[i]);
    }
    printf("\n");
}

int isPageInFrames(int frames[], int size, int page) {
    for (int i = 0; i < size; i++) {
        if (frames[i] == page) {
            return 1;
        }
    }
    return 0;
}

void fifo(int pages[], int pageCount, int frameCount) {
    int frames[frameCount];
    int pageFaults = 0, index = 0;
    for (int i = 0; i < frameCount; i++) {
        frames[i] = -1; // Initialize frames
    }
    printf("The Page Replacement Process is \n");
    for (int i = 0; i < pageCount; i++) {
        if (!isPageInFrames(frames, frameCount, pages[i])) {
            frames[index] = pages[i];
            index = (index + 1) % frameCount;
            pageFaults++;
            printFrames(frames, frameCount);
            printf("PF No. %d\n", pageFaults);
        } else {
            printFrames(frames, frameCount);
        }
    }
    printf("The number of Page Faults using FIFO are %d\n", pageFaults);
}

void lru(int pages[], int pageCount, int frameCount) {
    int frames[frameCount];
    int time[frameCount];
    int pageFaults = 0;
    for (int i = 0; i < frameCount; i++) {
```

```
        frames[i] = -1;
        time[i] = -1;
    }
    printf("The Page Replacement Process is \n");
    for (int i = 0; i < pageCount; i++) {
        int page = pages[i];
        int pageFound = 0;
        for (int k = 0; k < frameCount; k++) {
            if (frames[k] == page) {
                pageFound = 1;
                time[k] = i;
                break;
            }
        }
        if (!pageFound) {
            int lruIndex = 0;
            for (int j = 1; j < frameCount; j++) {
                if (time[j] < time[lruIndex]) {
                    lruIndex = j;
                }
            }
            frames[lruIndex] = page;
            time[lruIndex] = i;
            pageFaults++;
            printFrames(frames, frameCount);
            printf("PF No. %d\n", pageFaults);
        } else {
            printFrames(frames, frameCount);
        }
    }
    printf("The number of Page Faults using LRU are %d\n", pageFaults);
}

void optimal(int pages[], int pageCount, int frameCount) {
    int frames[frameCount];
    int pageFaults = 0;
    for (int i = 0; i < frameCount; i++) {
        frames[i] = -1;
    }
    printf("The Page Replacement Process is \n");
    for (int i = 0; i < pageCount; i++) {
        int page = pages[i];
        if (!isPageInFrames(frames, frameCount, page)) {
            int farthest = i;
            int replaceIndex = 0;
            for (int j = 0; j < frameCount; j++) {
                int k;
                for (k = i + 1; k < pageCount; k++) {
```

```
                    if (frames[j] == pages[k]) {
                        break;
                    }
                }
                if (k == pageCount) {
                    replaceIndex = j;
                    break;
                } else if (k > farthest) {
                    farthest = k;
                    replaceIndex = j;
                }
            }
            frames[replaceIndex] = page;
            pageFaults++;
            printFrames(frames, frameCount);
            printf("PF No. %d\n", pageFaults);
        } else {
            printFrames(frames, frameCount);
        }
    }
    printf("The number of Page Faults using Optimal are %d\n", pageFaults);
}

int main() {
    int pages[MAX_PAGES];
    int pageCount, frameCount;
    printf("Enter number of pages: ");
    scanf("%d", &pageCount);
    printf("Enter the page reference string:\n");
    for (int i = 0; i < pageCount; i++) {
        scanf("%d", &pages[i]);
    }
    printf("Enter number of frames: ");
    scanf("%d", &frameCount);
    printf("\nFIFO Page Replacement Algorithm:\n");
    fifo(pages, pageCount, frameCount);
    printf("\nLRU Page Replacement Algorithm:\n");
    lru(pages, pageCount, frameCount);
    printf("\nOptimal Page Replacement Algorithm:\n");
    optimal(pages, pageCount, frameCount);
    return 0;
}
```

OUTPUT-

```
Enter number of pages: 20
Enter the page reference string:
0 9 0 1 8 1 8 7 8 7 1 2 8 2 7 8 2 3 8 3
Enter number of frames: 3

FIFO Page Replacement Algorithm:
The Page Replacement Process is
0 - -
PF No. 1
0 9 -
PF No. 2
0 9 -
0 9 1
PF No. 3
8 9 1
PF No. 4
8 9 1
8 9 1
8 7 1
PF No. 5
8 7 1
8 7 1
8 7 1
8 7 2
PF No. 6
8 7 2
8 7 2
8 7 2
8 7 2
8 7 2
3 7 2
PF No. 7
3 8 2
PF No. 8
3 8 2
The number of Page Faults using FIFO are 8

LRU Page Replacement Algorithm:
The Page Replacement Process is
0 - -
PF No. 1
0 9 -
PF No. 2
0 9 -
0 9 1
PF No. 3
0 8 1
PF No. 4
0 8 1
0 8 1
```

```
7 8 1
PF No. 5
7 8 1
7 8 1
7 8 1
7 2 1
PF No. 6
8 2 1
PF No. 7
8 2 1
8 2 7
PF No. 8
8 2 7
8 2 7
8 2 3
PF No. 9
8 2 3
8 2 3
The number of Page Faults using LRU are 9

Optimal Page Replacement Algorithm:
The Page Replacement Process is
0 - -
PF No. 1
0 9 -
PF No. 2
0 9 -
1 9 -
PF No. 3
1 8 -
PF No. 4
1 8 -
1 8 -
1 8 7
PF No. 5
1 8 7
1 8 7
1 8 7
2 8 7
PF No. 6
2 8 7
2 8 7
2 8 7
2 8 7
2 8 7
3 8 7
PF No. 7
3 8 7
3 8 7
The number of Page Faults using Optimal are 7
```