LAB - 9

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

Source Code:

```
#include <stdio.h>

#define NUM_FRAMES 3
#define NUM_PAGES 10

void printFrames(int frames[]) {
   for (int i = 0; i < NUM_FRAMES; i++)
      printf("%2d ", frames[i]);
   printf("\n");
}

int findIndex(int arr[], int n, int element) {
   for (int i = 0; i < n; i++)
      if (arr[i] == element)
         return i;
   return -1;
}

void fifo(int pages[]) {
   int frames[NUM_FRAMES] = {0};
   int frameIndex = 0, pageFaults = 0;

   for (int i = 0; i < NUM_PAGES; i++) {
      int page = pages[i];
      if (findIndex(frames, NUM_FRAMES, page) == -1) {
         frames[frameIndex] = page;
         frameIndex = (frameIndex + 1) % NUM_FRAMES;
         pageFaults++;
      }
      printf("Page %2d -> ", page);
      printFrames(frames);
   }
```

```
    printf("FIFO Page Faults: %d\n", pageFaults);
}

void lru(int pages[]) {
    int frames[NUM_FRAMES] = {0};
    int pageFaults = 0;

    for (int i = 0; i < NUM_PAGES; i++) {
        int page = pages[i];
        int index = findIndex(frames, NUM_FRAMES, page);
        if (index == -1) {
            for (int j = 0; j < NUM_FRAMES; j++)
                if (frames[j] == 0 || findIndex(pages, i, frames[j]) == -1) {
                    frames[j] = page;
                    break;
                }
            pageFaults++;
        }
        printf("Page %2d -> ", page);
        printFrames(frames);
    }

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

void optimal(int pages[]) {
    int frames[NUM_FRAMES] = {0};
    int pageFaults = 0;

    for (int i = 0; i < NUM_PAGES; i++) {
        int page = pages[i];
        int index = findIndex(frames, NUM_FRAMES, page);
        if (index == -1) {
            int optimalIndex = -1;
            for (int j = 0; j < NUM_FRAMES; j++) {
                int pageIndex = findIndex(pages, NUM_PAGES, frames[j]);
                if (pageIndex == -1) {
                    optimalIndex = j;
                    break;
                }
```

```
            if (optimalIndex == -1 || pageIndex > findIndex(pages,
NUM_PAGES, frames[optimalIndex]))
                optimalIndex = j;
        }
        frames[optimalIndex] = page;
        pageFaults++;
    }
    printf("Page %2d -> ", page);
    printFrames(frames);
  }

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

int main() {
  int pages[NUM_PAGES] = {2, 3, 2, 1, 5, 2, 4, 5, 3, 2};

  printf("Page Reference Sequence: ");
  for (int i = 0; i < NUM_PAGES; i++)
    printf("%2d ", pages[i]);
  printf("\n");

  int ch;
  printf("Enter choice (1: FIFO, 2: LRU, 3: Optimal, 4: Exit): ");
  scanf("%d", &ch);

  switch (ch) {
    case 1:
      fifo(pages);
      break;
    case 2:
      lru(pages);
      break;
    case 3:
      optimal(pages);
      break;
    case 4:
      exit(0);
      break;
    default:
```

```
        printf("Invalid choice\n");
        break;
    }

    return 0;
}
```

OUTPUT:

FIFO:

```
Page Reference Sequence:  2  3  2  1  5  2  4  5  3  2
enter 1 for FIFO
 enter 2 for LRU
 Enter 3 for OPTIMAL
1
Page  2 ->  2  0  0
Page  3 ->  2  3  0
Page  2 ->  2  3  0
Page  1 ->  2  3  1
Page  5 ->  5  3  1
Page  2 ->  5  2  1
Page  4 ->  5  2  4
Page  5 ->  5  2  4
Page  3 ->  3  2  4
Page  2 ->  3  2  4
FIFO Page Faults: 7

Process returned 0 (0x0)   execution time : 2.953 s
Press any key to continue.
```

LRU:

```
Page Reference Sequence:  2  3  2  1  5  2  4  5  3  2
Enter choice (1: FIFO, 2: LRU, 3: Optimal, 4: Exit): 2
Page  2 ->  2  0  0
Page  3 ->  2  3  0
Page  2 ->  2  3  0
Page  1 ->  2  3  1
Page  5 ->  2  3  1
Page  2 ->  2  3  1
Page  4 ->  2  3  1
Page  5 ->  2  3  1
Page  3 ->  2  3  1
Page  2 ->  2  3  1
LRU Page Faults: 6

Process returned 0 (0x0)   execution time : 10.565 s
Press any key to continue.
```

OPTIMAL:

```
Page Reference Sequence:  2  3  2  1  5  2  4  5  3  2
Enter choice (1: FIFO, 2: LRU, 3: Optimal, 4: Exit): 3
Page  2 ->  2  0  0
Page  3 ->  2  3  0
Page  2 ->  2  3  0
Page  1 ->  2  3  1
Page  5 ->  2  3  5
Page  2 ->  2  3  5
Page  4 ->  2  3  4
Page  5 ->  2  3  5
Page  3 ->  2  3  5
Page  2 ->  2  3  5
Optimal Page Faults: 6

Process returned 0 (0x0)   execution time : 2.812 s
Press any key to continue.
```