Index

S.No	Title	Date Allotted	Date Checked	Sign/ Remarks
1	Linux Terminal Commands	3/01/23		
2	Linux Shell Commands	10/01/23		
3	Exercise (1)	17/01/23		
4	Exercise (2)	17/02/23		
5	Exercise (3)	24/01/23		
6	Exercise (4)	31/01/23		
7	Process Scheduling(FCFS)	7/02/23		
8	Process Scheduling(SJF)	14/02/23		

S.No	Title	Date Allotted	Date Checked	Sign/ Remarks
9	Process Scheduling(SRTF)	14/02/23		
10	Process Scheduling(Non-preemptive Priority)	21/02/23		
11	Process Scheduling(Preemptive Priority)	21/02/23		
12	Process Scheduling(Round Robin)	28/02/23		
13	Resource Allocation and Safe States(Banker's Algorithm)	13/03/23		
14	Page Re-Allocation (FIFO)	04/04/23		
15	Page Re-Allocation (LRU)	04/04/23		
16	Page Re-Allocation (Optimal)	04/04/23		

Page Re-Allocation(FIFO)

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

#define MAX_PAGES 3
#define MAX_FRAMES 4

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

int findOldestPageIndex(int pages[], int n, int frames[], int size) {
    int oldestIndex = 0;
    int oldestPage = pages[0];
    for (int i = 1; i < n; i++) {
        if (!isPagePresent(pages[i], frames, size) && pages[i] != oldestPage) {
            return i;
        }
    }
    return oldestIndex;
}

void pageReAllocationFIFO(int pages[], int n, int frames[], int size) {
    int pageFaults = 0;
    int frameIndex = 0;

    printf("Page Re-Allocation (FIFO) Algorithm:\n");

    for (int i = 0; i < n; i++) {
        printf("Accessing Page %d\n", pages[i]);
        if (!isPagePresent(pages[i], frames, size)) {
            printf("Page Fault: Page %d not found in frames.\n", pages[i]);
            pageFaults++;

            if (frameIndex == size) {
                int oldestIndex = findOldestPageIndex(pages, n, frames, size);
                printf("Replacing Page %d with Page %d\n", frames[oldestIndex], pages[i]);
                frames[oldestIndex] = pages[i];
            }
            else {
                printf("Adding Page %d to Frame %d\n", pages[i], frameIndex);
                frames[frameIndex++] = pages[i];
            }
        }
        else {
            printf("Page %d found in frames.\n", pages[i]);
        }
```

```
            printf("Current Frames: ");
            for (int j = 0; j < frameIndex; j++) {
                printf("%d ", frames[j]);
            }
            printf("\n");
        }

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

int main() {
    int pages[] = {1, 2, 3, 4, 5, 1, 2, 3, 4, 5};
    int n = sizeof(pages) / sizeof(pages[0]);

    int frames[MAX_FRAMES];

    for (int i = 0; i < MAX_FRAMES; i++) {
        frames[i] = -1;
    }

    pageReAllocationFIFO(pages, n, frames, MAX_FRAMES);

    return 0;
}
```

Output:

```
Accessing Page 5
Page Fault: Page 5 not found in frames.
Replacing Page 1 with Page 5
Current Frames: 5 2 3 4
Total Page Faults: 6
```

Page Re-Allocation(LRU)

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

#define MAX_PAGES 3
#define MAX_FRAMES 4

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

int findLeastRecentlyUsedPageIndex(int pages[], int n, int frames[], int size) {
    int lruIndex = 0;
    int lruPage = pages[0];
    for (int i = 1; i < n; i++) {
        if (!isPagePresent(pages[i], frames, size) && pages[i] != lruPage) {
            return i;
        }
    }
    return lruIndex;
}

void pageReAllocationLRU(int pages[], int n, int frames[], int size) {
    int pageFaults = 0;
    int frameIndex = 0;

    printf("Page Re-Allocation (LRU) Algorithm:\n");

    for (int i = 0; i < n; i++) {
        printf("Accessing Page %d\n", pages[i]);
        if (!isPagePresent(pages[i], frames, size)) {
            printf("Page Fault: Page %d not found in frames.\n", pages[i]);
            pageFaults++;

            if (frameIndex == size) {
                int lruIndex = findLeastRecentlyUsedPageIndex(pages, n, frames, size);
                printf("Replacing Page %d with Page %d\n", frames[lruIndex], pages[i]);
                frames[lruIndex] = pages[i];
            }
            else {
                printf("Adding Page %d to Frame %d\n", pages[i], frameIndex);
                frames[frameIndex++] = pages[i];
            }
        }
        else {
            printf("Page %d found in frames.\n", pages[i]);
        }
```

```
        printf("Current Frames: ");
        for (int j = 0; j < frameIndex; j++) {
            printf("%d ", frames[j]);
        }
        printf("\n");
    }

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

int main() {
    int pages[] = {1, 2, 3, 4, 5, 1, 2, 3, 4, 5};
    int n = sizeof(pages) / sizeof(pages[0]);

    int frames[MAX_FRAMES];

    for (int i = 0; i < MAX_FRAMES; i++) {
        frames[i] = -1;
    }

    pageReAllocationLRU(pages, n, frames, MAX_FRAMES);

    return 0;
}
```

Output:

```
Accessing Page 5
Page Fault: Page 5 not found in frames.
Replacing Page 1 with Page 5
Current Frames: 5 2 3 4
Total Page Faults: 6
```

Page Re-Allocation(Optimal)

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

#define MAX_PAGES 3
#define MAX_FRAMES 4

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

int findOptimalPageIndex(int pages[], int n, int frames[], int size, int currentIndex) {
    int farthestIndex = -1;
    int farthestPage = -1;
    for (int i = 0; i < size; i++) {
        int j;
        for (j = currentIndex + 1; j < n; j++) {
            if (pages[j] == frames[i]) {
                if (j > farthestIndex) {
                    farthestIndex = j;
                    farthestPage = pages[j];
                }
                break;
            }
        }
        if (j == n) {
            return i;
        }
    }
    if (farthestIndex == -1) {
        return 0;
    }
    else {
        return farthestIndex;
    }
}

void pageReAllocationOptimal(int pages[], int n, int frames[], int size) {
    int pageFaults = 0;
    int frameIndex = 0;

    printf("Page Re-Allocation (Optimal) Algorithm:\n");

    for (int i = 0; i < n; i++) {
        printf("Accessing Page %d\n", pages[i]);
        if (!isPagePresent(pages[i], frames, size)) {
            printf("Page Fault: Page %d not found in frames.\n", pages[i]);
            pageFaults++;
```

```
            if (frameIndex == size) {
                int optimalIndex = findOptimalPageIndex(pages, n, frames, size, i);
                printf("Replacing Page %d with Page %d\n", frames[optimalIndex], pages[i]);
                frames[optimalIndex] = pages[i];
            }
            else {
                printf("Adding Page %d to Frame %d\n", pages[i], frameIndex);
                frames[frameIndex++] = pages[i];
            }
        }
        else {
            printf("Page %d found in frames.\n", pages[i]);
        }

        printf("Current Frames: ");
        for (int j = 0; j < frameIndex; j++) {
            printf("%d ", frames[j]);
        }
        printf("\n");
    }

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

int main() {
    int pages[] = {1, 2, 3, 4, 5, 1, 2, 3, 4, 5};
    int n = sizeof(pages) / sizeof(pages[0]);

    int frames[MAX_FRAMES];

    for (int i = 0; i < MAX_FRAMES; i++) {
        frames[i] = -1;
    }

    pageReAllocationOptimal(pages, n, frames, MAX_FRAMES);

    return 0;
}
```

Output:

```
Accessing Page 5
Page Fault: Page 5 not found in frames.
Replacing Page 1 with Page 5
Current Frames: 5 2 3 4
Total Page Faults: 6
```