**Task 3**

**Description**

We had to create a program that assigns an 8-bit counter to each page then shifts the counter to the right depending on the number of frames provided then the page with the smallest counter is replaced, this showed that page faults decreased as the number of frames increased but at some point increasing the frames doesn’t do a lot.

**Code**

#include <stdio.h>  
#include <stdlib.h>  
#include <limits.h>  
#include <string.h>  
  
#define MAX\_PAGES 10000  // Maximum number of pages in the reference sequence  
  
typedef struct {  
    int page\_number;  
    unsigned int age; // Age counter for aging algorithm  
} PageFrame;  
  
void simulate\_aging(int num\_frames, int \*page\_references, int num\_references, FILE \*output) {  
    PageFrame \*frames = (PageFrame \*)malloc(num\_frames \* sizeof(PageFrame));  
    int page\_faults = 0;  
    int count = 0;  
     
    for (int i = 0; i < num\_frames; i++) {  
        frames[i].page\_number = -1; // Empty frame  
        frames[i].age = 0;  
    }  
     
    for (int i = 0; i < num\_references; i++) {  
        int page = page\_references[i];  
        int found = 0;  
         
        // Age all pages  
        for (int j = 0; j < num\_frames; j++) {  
            frames[j].age >>= 1;  
        }  
         
        // Check if page is already in memory  
        for (int j = 0; j < num\_frames; j++) {  
            if (frames[j].page\_number == page) {  
                frames[j].age |= (1 << 31); // Set MSB as reference bit  
                found = 1;  
                break;  
            }  
        }  
         
        if (!found) { // Page fault  
            page\_faults++;  
             
            // Find the least recently used page (smallest age)  
            int min\_index = 0;  
            unsigned int min\_age = UINT\_MAX;  
             
            for (int j = 0; j < num\_frames; j++) {  
                if (frames[j].page\_number == -1) { // Empty frame found  
                    min\_index = j;  
                    break;  
                }  
                if (frames[j].age < min\_age) {  
                    min\_age = frames[j].age;  
                    min\_index = j;  
                }  
            }  
             
            // Replace the least recently used page  
            frames[min\_index].page\_number = page;  
            frames[min\_index].age = (1 << 31); // Set MSB for new page  
        }  
         
        count++;  
        if (count == 1000) {  
            fprintf(output, "%d %d\n", num\_frames, page\_faults);  
            count = 0;  
            page\_faults = 0;  
        }  
    }  
     
    free(frames);  
}  
  
int main(int argc, char \*argv[]) {  
    if (argc != 2) {  
        printf("Usage: %s <input\_file>\n", argv[0]);  
        return 1;  
    }  
     
    FILE \*file = fopen(argv[1], "r");  
    if (!file) {  
        perror("Error opening file");  
        return 1;  
    }  
     
    int page\_references[MAX\_PAGES];  
    int num\_references = 0;  
    while (fscanf(file, "%d", &page\_references[num\_references]) == 1) {  
        num\_references++;  
        if (num\_references >= MAX\_PAGES) break;  
    }  
    fclose(file);  
     
    FILE \*output = fopen("output.dat", "w");  
    if (!output) {  
        perror("Error opening output file");  
        return 1;  
    }  
     
    for (int frames = 1; frames <= 50; frames++) {  
        simulate\_aging(frames, page\_references, num\_references, output);  
    }  
    fclose(output);  
     
    printf("Simulation complete. Data saved to output.dat.\n");  
    return 0;  
}

**Gitlink**

[**https://github.com/FirasAhmed2/Operating-systems-coursework.git**](https://github.com/FirasAhmed2/Operating-systems-coursework.git)