Assignment 6

```
#include <stdio.h>

#include <stdlib.h>


#define MAX_FRAMES 10

#define MAX_PAGES 30


// Function prototypes

void fifo(int pages[], int n, int frames);

void lru(int pages[], int n, int frames);

void optimal(int pages[], int n, int frames);


// Main function

int main() {

    int pages[MAX_PAGES], n, frames;

    int choice;


    while (1) {

        printf("\n--- Page Replacement Algorithms ---\n");

        printf("1. FIFO\n");

        printf("2. LRU\n");

        printf("3. Optimal\n");

        printf("4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        if (choice == 4) {

            printf("Exiting...\n");

            break;

        }

        printf("Enter the number of frames: ");

        scanf("%d", &frames);

        printf("Enter the number of pages: ");

        scanf("%d", &n);
        printf("Enter the page reference string: ");
        for (int i = 0; i < n; i++) {
            scanf("%d", &pages[i]);
        }
        switch (choice) {
            case 1:
                fifo(pages, n, frames);
                break;
            case 2:
                lru(pages, n, frames);
                break;
            case 3:
                optimal(pages, n, frames);
                break;
            default:
                printf("Invalid choice, please try again.\n");
        }
    }
    return 0;
}
// FIFO Page Replacement Algorithm
void fifo(int pages[], int n, int frames) {
    int frame[MAX_FRAMES] = {-1};
    int front = 0, pageFaults = 0;
    printf("\nFIFO Page Replacement\n");
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < frames; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                break;
            }
        }
        if (!found) {
            frame[front] = pages[i];
            front = (front + 1) % frames;
            pageFaults++;
        }
        // Print frames
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frames; j++) {
            if (frame[j] != -1) {
                printf("%d ", frame[j]);
            } else {
                printf("- ");
            }
        }
        printf("\n");
    }
    printf("Total page faults: %d\n", pageFaults);
}
// LRU Page Replacement Algorithm
void lru(int pages[], int n, int frames) {
    int frame[MAX_FRAMES] = {-1}, age[MAX_FRAMES] = {0};
    int pageFaults = 0;
    printf("\nLRU Page Replacement\n");
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < frames; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                age[j] = i; // Update age
                break;
            }
        }
        if (!found) {
            int min_age = 0;
            for (int j = 1; j < frames; j++) {
```

```c
                if (age[j] < age[min_age]) {
                    min_age = j;
                }
            }
            frame[min_age] = pages[i];
            age[min_age] = i;
            pageFaults++;
        }
        // Print frames
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frames; j++) {
            if (frame[j] != -1) {
                printf("%d ", frame[j]);
            } else {
                printf("- ");
            }
        }
        printf("\n");
    }
    printf("Total page faults: %d\n", pageFaults);
}
// Optimal Page Replacement Algorithm
void optimal(int pages[], int n, int frames) {
    int frame[MAX_FRAMES] = {-1};
    int pageFaults = 0;
    printf("\nOptimal Page Replacement\n");
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < frames; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                break;
            }
        }
        if (!found) {
            int farthest = i + 1, replaceIndex = -1;
            for (int j = 0; j < frames; j++) {
                int k;
                for (k = i + 1; k < n; k++) {
                    if (frame[j] == pages[k]) {
                        if (k > farthest) {
                            farthest = k;
                            replaceIndex = j;
                        }
                        break;
                    }
                }
                if (k == n) {  // Page not found later
                    replaceIndex = j;
                    break;
                }
            }
            if (replaceIndex == -1) replaceIndex = 0;
            frame[replaceIndex] = pages[i];
            pageFaults++;
        }
        // Print frames
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frames; j++) {
            if (frame[j] != -1) {
                printf("%d ", frame[j]);
            } else {
                printf("- ");
            }
        }
        printf("\n");
    }
    printf("Total page faults: %d\n", pageFaults); }
```

Output:

```
--- Page Replacement Algorithms ---

1. FIFO

2. LRU

3. Optimal

4. Exit

Enter your choice: 1

Enter the number of frames: 3

Enter the number of pages: 8

Enter the page reference string: 3

6

6

1

7

8

5

4


FIFO Page Replacement

Page 3: 3 0 0

Page 6: 3 6 0

Page 6: 3 6 0

Page 1: 3 6 1

Page 7: 7 6 1

Page 8: 7 8 1

Page 5: 7 8 5

Page 4: 4 8 5

Total page faults: 7


--- Page Replacement Algorithms ---

1. FIFO

2. LRU

3. Optimal

4. Exit
```

```
Enter your choice: 2

Enter the number of frames: 3

Enter the number of pages: 5

Enter the page reference string: 7

4

2

8

9


LRU Page Replacement

Page 7: 7 0 0

Page 4: 4 0 0

Page 2: 4 2 0

Page 8: 4 2 8

Page 9: 9 2 8

Total page faults: 5


--- Page Replacement Algorithms ---

1. FIFO

2. LRU

3. Optimal

4. Exit

Enter your choice: 3

Enter the number of frames: 3

Enter the number of pages: 1

Enter the page reference string: 4


Optimal Page Replacement

Page 4: 4 0 0

Total page faults: 1
```