Write a C program to simulate page replacement algorithms

a) FIFO
b) LRU
c) Optimal

Code:

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

#define MAX 50

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

int search(int key, int frames[], int frameCount) {
    for (int i = 0; i < frameCount; i++) {
        if (frames[i] == key)
            return 1;
    }
    return 0;
}

int findLRU(int time[], int frameCount) {
    int min = time[0], pos = 0;
    for (int i = 1; i < frameCount; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

int findOptimal(int pages[], int frames[], int index, int len, int frameCount) {
    int pos = -1, farthest = index;
    for (int i = 0; i < frameCount; i++) {
```

```
        int j;
        for (j = index; j < len; j++) {
            if (frames[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    pos = i;
                }
                break;
            }
        }
        if (j == len) return i;
    }
    return (pos == -1) ? 0 : pos;
}

void fifo(int pages[], int len, int frameCount) {
    int frames[MAX], front = 0, faults = 0;

    for (int i = 0; i < frameCount; i++)
        frames[i] = -1;

    printf("\nFIFO Page Replacement Process:\n");
    for (int i = 0; i < len; i++) {
        if (!search(pages[i], frames, frameCount)) {
            frames[front] = pages[i];
            front = (front + 1) % frameCount;
            faults++;
            printf("PF No. %d: ", faults);
            printFrames(frames, frameCount);
        }
    }
    printf("FIFO Page Faults: %d\n", faults);
}

void lru(int pages[], int len, int frameCount) {
    int frames[MAX], time[MAX], count = 0, faults = 0;

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

    printf("\nLRU Page Replacement Process:\n");
    for (int i = 0; i < len; i++) {
```

```
        if (!search(pages[i], frames, frameCount)) {
            int pos = (count < frameCount) ? count : findLRU(time, frameCount);
            frames[pos] = pages[i];
            time[pos] = i;
            faults++;
            printf("PF No. %d: ", faults);
            printFrames(frames, frameCount);
            if (count < frameCount) count++;
        } else {
            for (int j = 0; j < frameCount; j++) {
                if (frames[j] == pages[i]) {
                    time[j] = i;
                    break;
                }
            }
        }
    }
    printf("LRU Page Faults: %d\n", faults);
}

void optimal(int pages[], int len, int frameCount) {
    int frames[MAX], faults = 0, count = 0;

    for (int i = 0; i < frameCount; i++)
        frames[i] = -1;

    printf("\nOptimal Page Replacement Process:\n");
    for (int i = 0; i < len; i++) {
        if (!search(pages[i], frames, frameCount)) {
            int pos = (count < frameCount) ? count : findOptimal(pages, frames, i + 1, len,
frameCount);
            frames[pos] = pages[i];
            faults++;
            printf("PF No. %d: ", faults);
            printFrames(frames, frameCount);
            if (count < frameCount) count++;
        }
    }
    printf("Optimal Page Faults: %d\n", faults);
}

int main() {
    int frames, len, pages[MAX];
```

```
    printf("Enter the number of Frames: ");
    scanf("%d", &frames);
    printf("Enter the length of reference string: ");
    scanf("%d", &len);
    printf("Enter the reference string: ");
    for (int i = 0; i < len; i++) {
        scanf("%d", &pages[i]);
    }

    fifo(pages, len, frames);
    lru(pages, len, frames);
    optimal(pages, len, frames);

    return 0;
}
```

Output:

```
Enter the number of Frames: 3
Enter the length of reference string: 22
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 0 3 2 1 2 0 1 7 0 1

FIFO Page Replacement Process:
PF No. 1: 7 _ _
PF No. 2: 7 0 _
PF No. 3: 7 0 1
PF No. 4: 2 0 1
PF No. 5: 2 3 1
PF No. 6: 2 3 0
PF No. 7: 4 3 0
PF No. 8: 4 2 0
PF No. 9: 4 2 3
PF No. 10: 0 2 3
PF No. 11: 0 1 3
PF No. 12: 0 1 2
PF No. 13: 7 1 2
PF No. 14: 7 0 2
PF No. 15: 7 0 1
FIFO Page Faults: 15

LRU Page Replacement Process:
PF No. 1: 7 _ _
PF No. 2: 7 0 _
PF No. 3: 7 0 1
PF No. 4: 2 0 1
PF No. 5: 2 0 3
PF No. 6: 4 0 3
PF No. 7: 4 0 2
PF No. 8: 4 3 2
PF No. 9: 0 3 2
PF No. 10: 1 3 2
PF No. 11: 1 0 2
PF No. 12: 1 0 7
LRU Page Faults: 12

Optimal Page Replacement Process:
PF No. 1: 7 _ _
PF No. 2: 7 0 _
PF No. 3: 7 0 1
PF No. 4: 2 0 1
PF No. 5: 2 0 3
PF No. 6: 2 4 3
PF No. 7: 2 0 3
PF No. 8: 2 0 1
PF No. 9: 7 0 1
Optimal Page Faults: 9

Process returned 0 (0x0)   execution time : 47.714 s
Press any key to continue.
```