Write a c program to stimulate page replacement algorithms

a) **FIFO**
b) **LRU**
c) **Optimal**

```
#include <stdio.h>

#define MAX 100


int inFrame(int frames[], int size, int page) {
    for (int i = 0; i < size; i++)
        if (frames[i] == page)
            return i;
    return -1;
}


int fifo(int pages[], int n, int cap) {
    int frames[MAX], index = 0, count = 0, faults = 0;


    for (int i = 0; i < n; i++) {
        if (inFrame(frames, count, pages[i]) == -1) {
            if (count < cap) {
                frames[count++] = pages[i];
            } else {
                frames[index] = pages[i];
                index = (index + 1) % cap;
            }
```

```
            faults++;
        }
    }
    return faults;
}

int lru(int pages[], int n, int cap) {
    int frames[MAX], recent[MAX], count = 0, faults = 0;

    for (int i = 0; i < n; i++) {
        int idx = inFrame(frames, count, pages[i]);
        if (idx != -1) {
            recent[idx] = i;
        } else {
            if (count < cap) {
                frames[count] = pages[i];
                recent[count++] = i;
            } else {
                int lru = 0;
                for (int j = 1; j < cap; j++)
                    if (recent[j] < recent[lru])
                        lru = j;

                frames[lru] = pages[i];
                recent[lru] = i;
            }
```

```
            faults++;
        }
    }
    return faults;
}

int optimal(int pages[], int n, int cap) {
    int frames[MAX], count = 0, faults = 0;

    for (int i = 0; i < n; i++) {
        if (inFrame(frames, count, pages[i]) != -1)
            continue;

        if (count < cap) {
            frames[count++] = pages[i];
        } else {
            int far = -1, idx = -1;
            for (int j = 0; j < cap; j++) {
                int k;
                for (k = i + 1; k < n; k++)
                    if (frames[j] == pages[k])
                        break;
                if (k > far) {
                    far = k;
                    idx = j;
                }
```

```
            }
            frames[idx] = pages[i];
        }
        faults++;
    }
    return faults;
}

int main() {
    int pages[MAX], n, cap;

    printf("Enter number of pages: ");
    scanf("%d", &n);

    printf("Enter reference string: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &pages[i]);

    printf("Enter number of frames: ");
    scanf("%d", &cap);

    printf("\nPage Faults:\n");
    printf("FIFO   : %d\n", fifo(pages, n, cap));
    printf("LRU    : %d\n", lru(pages, n, cap));
    printf("Optimal: %d\n", optimal(pages, n, cap));
```

```
    return 0;
}
```

OUTPUT: