

33. Construct a C program to simulate the optimal paging technique of memory Management.

```
#include <stdio.h>

int findOptimal(int pages[], int frame[], int n, int index, int frameCount) {
    int farthest = index, result = -1;
    int i, j;
    for (i = 0; i < frameCount; i++) {
        int found = 0;
        for (j = index; j < n; j++) {
            if (frame[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    result = i;
                }
                found = 1;
                break;
            }
        }
        if (found == 0) // Not used in future
            return i;
    }
    if (result == -1)
        return 0;
    else
        return result;
}

int main() {
    int i, j, n, frameCount;
    int pages[30], frame[10];
    int pageFaults = 0;
```

```
int flag, pos;

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

scanf("%d", &n);

printf("Enter the page reference string:\n");

for (i = 0; i < n; i++)

scanf("%d", &pages[i]);

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

scanf("%d", &frameCount);

for (i = 0; i < frameCount; i++)

frame[i] = -1;

printf("\nPage Reference | Frame Content | Page Fault\n");

printf("-----\n");

for (i = 0; i < n; i++) {

flag = 0;

// Check if page is already in frame

for (j = 0; j < frameCount; j++) {

if (frame[j] == pages[i]) {

flag = 1;

break;

}

}

// Page fault

if (flag == 0) {

// Empty frame

int emptyFound = 0;

for (j = 0; j < frameCount; j++) {

if (frame[j] == -1) {

frame[j] = pages[i];

emptyFound = 1;

break;

}

}

}
```

```

    }

    // No empty frame => replace optimal page
    if (!emptyFound) {
        pos = findOptimal(pages, frame, n, i + 1, frameCount);
        frame[pos] = pages[i];
    }

    pageFaults++;

    printf("%10d | ", pages[i]);
    for (j = 0; j < frameCount; j++) {
        if (frame[j] != -1)
            printf("%d ", frame[j]);
        else
            printf("- ");
    }
    printf("|\n");
} else {
    printf("%10d | ", pages[i]);
    for (j = 0; j < frameCount; j++) {
        if (frame[j] != -1)
            printf("%d ", frame[j]);
        else
            printf("- ");
    }
    printf("|\n");
}

printf("\nTotal Page Faults = %d\n", pageFaults);
printf("Page Fault Ratio = %.2f\n", (float) pageFaults / n);

return 0;
}

```

```
}
```

OUTPUT:

```
#include <stdio.h>
int findOptimal(int pages[], int frame[], int n, int index, int frameCount)
{
    int farthest = index, result = -1;
    int i, j;
    for (i = 0; i < frameCount; i++) {
        int found = 0;
        for (j = index; j < n; j++) {
            if (frame[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    result = i;
                }
                found = 1;
                break;
            }
        }
        if (found == 0) // Not used in future
            return i;
    }
    if (result == -1)
        return 0;
    else
        return result;
}
```

```
Enter the number of pages: 4
Enter the page reference string:
1 2 3 2
Enter the number of frames: 5

Page Reference | Frame Content | Page Fault
-----
1 | 1 - - - | Yes
2 | 1 2 - - | Yes
3 | 1 2 3 - - | Yes
2 | 1 2 3 - - | No

Total Page Faults = 3
Page Fault Ratio = 0.75

-----
Process exited after 11.36 seconds with return
value 0
Press any key to continue . . .
```