```
#include <stdio.h>

int findOptimal(int pages[], int frame[], int n, int index, int f) {
    int i, j, farthest = index, pos = -1, found;

    for (i = 0; i < f; i++) {
        found = 0;
        for (j = index; j < n; j++) {
            if (frame[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    pos = i;
                }
                found = 1;
                break;
            }
        }
        // If a page is not found in future, return it immediately
        if (found == 0)
            return i;
    }

    // If all pages are found, return the farthest one
    if (pos == -1)
        return 0;
    else
        return pos;
}

int main() {
    int pages[50], frame[10];
```

```
int n, f, i, j, pos, page_faults = 0, flag1, flag2;

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

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

printf("Enter the number of frames (minimum 3): ");
scanf("%d", &f);

if (f < 3) {
   printf("Error: Minimum frame size must be 3!\n");
   return 0;
}

for (i = 0; i < f; i++)
   frame[i] = -1;

printf("\nPage Replacement Process (Optimal):\n");

for (i = 0; i < n; i++) {
   flag1 = flag2 = 0;

   // Check if the page is already present in frame
   for (j = 0; j < f; j++) {
      if (frame[j] == pages[i]) {
         flag1 = flag2 = 1;
         break;
      }
```

```
    }

    // If page not found in frame and empty space available
    if (flag1 == 0) {
        for (j = 0; j < f; j++) {
            if (frame[j] == -1) {
                frame[j] = pages[i];
                page_faults++;
                flag2 = 1;
                break;
            }
        }
    }

    // If page not found and no empty frame → replace optimally
    if (flag2 == 0) {
        pos = findOptimal(pages, frame, n, i + 1, f);
        frame[pos] = pages[i];
        page_faults++;
    }

    printf("Page %d -> Frames: ", pages[i]);
    for (j = 0; j < f; j++) {
        if (frame[j] != -1)
            printf("%d ", frame[j]);
        else
            printf("- ");
    }
    printf("\n");
}
```

```
    printf("\nTotal Page Faults = %d\n", page_faults);
    printf("Page Fault Rate = %.2f%%\n", ((float)page_faults / n) * 100);

    return 0;
}
```

OUTPUT

```
Enter the number of pages: 12
Enter the page reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter the number of frames (minimum 3): 3
Page Replacement Process (Optimal):
Page 1 -> Frames: 1 - -
Page 2 -> Frames: 1 2 -
Page 3 -> Frames: 1 2 3
Page 4 -> Frames: 4 2 3
Page 1 -> Frames: 4 1 3
Page 2 -> Frames: 4 1 2
Page 5 -> Frames: 5 1 2
Page 1 -> Frames: 5 1 2
Page 2 -> Frames: 5 1 2
Page 3 -> Frames: 3 1 2
Page 4 -> Frames: 3 4 2
Page 5 -> Frames: 3 4 5

Total Page Faults = 9
Page Fault Rate = 75.00%
```