Pagerep.c

```
#include<stdio.h>
int n, nf, in[100], p[50], hit=0, pgfaultcnt=0;

void getData() {
   printf("\nEnter length of page reference sequence: ");
   scanf("%d", &n);
   printf("Enter page reference sequence: ");
   for(int i = 0; i < n; i++) scanf("%d", &in[i]);
   printf("Enter no of frames: ");
   scanf("%d", &nf);
}

void initialize() {
   pgfaultcnt = 0;
   for(int i = 0; i < nf; i++) p[i] = 9999;
}

int isHit(int data) {
   for(int j = 0; j < nf; j++) if(p[j] == data) return 1;
   return 0;
}

void dispPages() {
   for(int k = 0; k < nf; k++) if(p[k] != 9999) printf(" %d", p[k]);
}

void dispPgFaultCnt() {
   printf("\nTotal page faults: %d", pgfaultcnt);
}

void fifo() {
   initialize();
   for(int i = 0; i < n; i++) {
      if(!isHit(in[i])) {
         for(int k = 0; k < nf - 1; k++) p[k] = p[k + 1];
         p[nf - 1] = in[i];
         pgfaultcnt++;
         dispPages();
      } else {
         printf("No page fault");
      }
   }
   dispPgFaultCnt();
}

void optimal() {
   initialize();
   int near[50];
```

```
    for(int i = 0; i < n; i++) {
        if(!isHit(in[i])) {
            for(int j = 0; j < nf; j++) {
                int pg = p[j], found = 0;
                for(int k = i; k < n; k++) {
                    if(pg == in[k]) { near[j] = k; found = 1; break; }
                    else found = 0;
                }
                if(!found) near[j] = 9999;
            }
            int max = -9999, repindex = 0;
            for(int j = 0; j < nf; j++) {
                if(near[j] > max) { max = near[j]; repindex = j; }
            }
            p[repindex] = in[i];
            pgfaultcnt++;
            dispPages();
        } else {
            printf("No page fault");
        }
    }
    dispPgFaultCnt();
}

void lru() {
    initialize();
    int least[50];
    for(int i = 0; i < n; i++) {
        if(!isHit(in[i])) {
            for(int j = 0; j < nf; j++) {
                int pg = p[j], found = 0;
                for(int k = i - 1; k >= 0; k--) {
                    if(pg == in[k]) { least[j] = k; found = 1; break; }
                }
                if(!found) least[j] = -9999;
            }
            int min = 9999, repindex = 0;
            for(int j = 0; j < nf; j++) {
                if(least[j] < min) { min = least[j]; repindex = j; }
            }
            p[repindex] = in[i];
            pgfaultcnt++;
            dispPages();
        } else {
            printf("No page fault");
        }
    }
    dispPgFaultCnt();
}

int main() {
    int choice;
    while(1) {
        printf("\nPage Replacement Algorithms\n1.Enter data\n2.FIFO\n3.Optimal\n4.LRU\n5.Exit\nEnter your choice: ");
        scanf("%d", &choice);
        switch(choice) {
            case 1: getData(); break;
            case 2: fifo(); break;
            case 3: optimal(); break;
            case 4: lru(); break;
            default: return 0;
        }
    }
}
```