```
#include <stdio.h>

#define MAX_REF 100

#define MAX_FRAMES 50


int n, nf;

int in[MAX_REF];  // Page reference string

int frames[MAX_FRAMES];  // Memory frames

int pgfaultcnt = 0;


// Input Function

void getData() {

   printf("\nEnter length of page reference sequence: ");

   scanf("%d", &n);

   printf("Enter the page reference sequence: ");

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

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

   printf("Enter number of frames: ");

   scanf("%d", &nf);

}


// Initialize memory frames

void initialize() {

   pgfaultcnt = 0;

   for (int i = 0; i < nf; i++)

      frames[i] = -1;  // -1 indicates an empty frame

}
```

```
// Check if a page is a hit

int isHit(int page) {

    for (int i = 0; i < nf; i++) {

        if (frames[i] == page)

            return 1;

    }

    return 0;

}


// Get index of a hit page

int getHitIndex(int page) {

    for (int i = 0; i < nf; i++) {

        if (frames[i] == page)

            return i;

    }

    return -1;

}


// Display the current frames

void dispPages() {

    printf(" Frames: ");

    for (int i = 0; i < nf; i++) {

        if (frames[i] != -1)

            printf("%d ", frames[i]);

        else

            printf("- ");

    }
    printf("\n");
}

// Display total page faults
void dispPgFaultCnt() {
    printf("Total page faults: %d\n", pgfaultcnt);
}

// FIFO Page Replacement
void fifo() {
    initialize();
    int pointer = 0;
    printf("\n=== FIFO Page Replacement ===\n");
    for (int i = 0; i < n; i++) {
        printf("Reference: %d -> ", in[i]);
        if (!isHit(in[i])) {
            frames[pointer] = in[i];
            pointer = (pointer + 1) % nf;
            pgfaultcnt++;
            dispPages();
        } else {
            printf("No page fault.\n");
        }
    }
    dispPgFaultCnt();
}
```

```
// Optimal Page Replacement

void optimal() {

  initialize();

  int near[MAX_FRAMES];

  printf("\n=== Optimal Page Replacement ===\n");

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

    printf("Reference: %d -> ", in[i]);

    if (!isHit(in[i])) {

      for (int j = 0; j < nf; j++) {

        int found = 0;

        for (int k = i + 1; k < n; k++) {

          if (frames[j] == in[k]) {

            near[j] = k;

            found = 1;

            break;

          }

        }

        if (!found) near[j] = 9999;

      }


      int max = -1, index = -1;

      for (int j = 0; j < nf; j++) {

        if (near[j] > max) {

          max = near[j];

          index = j;

        }

      }
```

```
            frames[index] = in[i];
            pgfaultcnt++;
            dispPages();
        } else {
            printf("No page fault.\n");
        }
    }
    dispPgFaultCnt();
}

// Least Recently Used (LRU)
void lru() {
    initialize();
    int lastUsed[MAX_FRAMES];
    printf("\n=== LRU Page Replacement ===\n");
    for (int i = 0; i < n; i++) {
        printf("Reference: %d -> ", in[i]);
        if (!isHit(in[i])) {
            for (int j = 0; j < nf; j++) {
                int found = 0;
                for (int k = i - 1; k >= 0; k--) {
                    if (frames[j] == in[k]) {
                        lastUsed[j] = k;
                        found = 1;
                        break;
                    }
                }
            if (!found) lastUsed[j] = -9999;
          }

          int min = 9999, index = -1;
          for (int j = 0; j < nf; j++) {
            if (lastUsed[j] < min) {
              min = lastUsed[j];
              index = j;
            }
          }
          frames[index] = in[i];
          pgfaultcnt++;
          dispPages();
        } else {
          printf("No page fault.\n");
        }
      }
      dispPgFaultCnt();
    }

    // Least Frequently Used (LFU)
    void lfu() {
      initialize();
      int usedcnt[MAX_FRAMES] = {0};
      int loaded = 0;
      printf("\n=== LFU Page Replacement ===\n");
      for (int i = 0; i < n; i++) {
        printf("Reference: %d -> ", in[i]);

        int hit = isHit(in[i]);

        if (hit) {

            int idx = getHitIndex(in[i]);

            usedcnt[idx]++;

            printf("No page fault.\n");

        } else {

            pgfaultcnt++;

            if (loaded < nf) {  // Fill empty frames first

                frames[loaded] = in[i];

                usedcnt[loaded] = 1;

                loaded++;

            } else {  // Replace least frequently used

                int min = usedcnt[0], index = 0;

                for (int j = 1; j < nf; j++) {

                    if (usedcnt[j] < min) {

                        min = usedcnt[j];

                        index = j;

                    }

                }

                frames[index] = in[i];

                usedcnt[index] = 1;

            }

            dispPages();

        }

    }

    dispPgFaultCnt();
```

```
}

// Second Chance Algorithm
void secondchance() {
    initialize();
    int refBits[MAX_FRAMES] = {0};
    int pointer = 0;
    printf("\n=== Second Chance (Clock) Algorithm ===\n");
    for (int i = 0; i < n; i++) {
        printf("Reference: %d -> ", in[i]);
        int hit = isHit(in[i]);
        if (hit) {
            int idx = getHitIndex(in[i]);
            refBits[idx] = 1;
            printf("No page fault.\n");
        } else {
            pgfaultcnt++;
            while (refBits[pointer] == 1) {
                refBits[pointer] = 0;
                pointer = (pointer + 1) % nf;
            }
            frames[pointer] = in[i];
            refBits[pointer] = 1;
            pointer = (pointer + 1) % nf;
            dispPages();
        }
    }
    dispPgFaultCnt();

}


int main() {

    int choice;

    while (1) {

        printf("\n=== Page Replacement Algorithms ===\n");

        printf("1. Enter Data\n2. FIFO\n3. Optimal\n4. LRU\n5. LFU\n6. Second Chance\n7.
Exit\nChoice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1: getData(); break;

            case 2: fifo(); break;

            case 3: optimal(); break;

            case 4: lru(); break;

            case 5: lfu(); break;

            case 6: secondchance(); break;

            case 7: return 0;

            default: printf("Invalid choice!\n"); break;

        }

    }

}


/*OUTPUT –

=== Page Replacement Algorithms ===

1. Enter Data

2. FIFO

3. Optimal
```

```
4. LRU

5. LFU

6. Second Chance

7. Exit

Choice: 1

Enter length of page reference sequence: 12

Enter the page reference sequence: 1 3 0 3 5 6 3 1 6 3 6 1

Enter number of frames: 3


Choice: 2

=== FIFO Page Replacement ===

Reference: 1 -> Frames: 1 - -

Reference: 3 -> Frames: 1 3 -

Reference: 0 -> Frames: 1 3 0

Reference: 3 -> No page fault.

Reference: 5 -> Frames: 5 3 0

Reference: 6 -> Frames: 6 3 0

Reference: 3 -> No page fault.

Reference: 1 -> Frames: 1 3 0

Reference: 6 -> Frames: 6 3 0

Reference: 3 -> No page fault.

Reference: 6 -> No page fault.

Reference: 1 -> Frames: 1 3 0

Total page faults: 7

*/
```