- **Page replacement algorithms:**

```
#include <stdio.h>
void printFrames(int frames[], int n) {
  for (int i = 0; i < n; i++) {
    if (frames[i] == -1)
      printf(" - ");
    else
      printf(" %d ", frames[i]);
  }
  printf("\n");
}

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

// FIFO Page Replacement
int fifo(int ref[], int n, int frames[], int f) {
  int faults = 0, index = 0;
  for (int i = 0; i < n; i++) {
    if (searchFrame(frames, f, ref[i]) == -1) {  // Page fault
      frames[index] = ref[i];
      index = (index + 1) % f;  // Circular index
      faults++;
    }
    printFrames(frames, f);
  }
  return faults;
}

// LRU Page Replacement
int lru(int ref[], int n, int frames[], int f) {
  int faults = 0, time[f], least;
  for (int i = 0; i < f; i++) frames[i] = -1;

  for (int i = 0; i < n; i++) {
    int pos = searchFrame(frames, f, ref[i]);
    if (pos == -1) {  // Page fault
      least = 0;
      for (int j = 1; j < f; j++) {
            if (time[j] < time[least])
                least = j;
        }
        frames[least] = ref[i];
        faults++;
    } else {  // Update the usage time for the found frame
        least = pos;
    }
    time[least] = i;  // Update time
    printFrames(frames, f);
  }
  return faults;
}

// MRU Page Replacement
int mru(int ref[], int n, int frames[], int f) {
  int faults = 0, time[f], most;
  for (int i = 0; i < f; i++) frames[i] = -1;

  for (int i = 0; i < n; i++) {
    int pos = searchFrame(frames, f, ref[i]);
    if (pos == -1) {  // Page fault
        most = 0;
        for (int j = 1; j < f; j++) {
            if (time[j] > time[most])
                most = j;
        }
        frames[most] = ref[i];
        faults++;
    } else {  // Update the usage time for the found frame
        most = pos;
    }
    time[most] = i;  // Update time
    printFrames(frames, f);
  }
  return faults;
}

// Optimal Page Replacement
int optimal(int ref[], int n, int frames[], int f) {
  int faults = 0;
  for (int i = 0; i < f; i++) frames[i] = -1;

  for (int i = 0; i < n; i++) {
    int pos = searchFrame(frames, f, ref[i]);
    if (pos == -1) {  // Page fault
        int farthest = -1, replace = -1;
        for (int j = 0; j < f; j++) {
            int k;
            for (k = i + 1; k < n; k++) {
                if (frames[j] == ref[k]) break;
            }
            if (k > farthest) {
                farthest = k;
                replace = j;
            }
        }
        frames[replace] = ref[i];
        faults++;
    }
    printFrames(frames, f);
  }
  return faults;
}

int main() {
  int ref[50], frames[10], n, f, choice, faults;

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

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

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

  printf("Choose Page Replacement Algorithm:\n");
  printf("1. FIFO\n2. LRU\n3. MRU\n4. Optimal\n");
  scanf("%d", &choice);

  // Initialize frames
  for (int i = 0; i < f; i++) frames[i] = -1;

  switch (choice) {
    case 1:
        printf("FIFO Page Replacement\n");
        faults = fifo(ref, n, frames, f);
        break;
    case 2:
        printf("LRU Page Replacement\n");
        faults = lru(ref, n, frames, f);
        break;
    case 3:
        printf("MRU Page Replacement\n");
        faults = mru(ref, n, frames, f);
        break;
    case 4:
        printf("Optimal Page Replacement\n");
        faults = optimal(ref, n, frames, f);
        break;
    default:
        printf("Invalid choice!\n");
        return 1;
  }

  printf("Total Page Faults: %d\n", faults);
  return 0;
}
```