```c
#include <stdio.h>

int search(int key, int frameItems[], int frame_occupied)
{
  for (int i = 0; i < frame_occupied; i++)
    if (frameItems[i] == key)
      return 1;
  return 0;
}
void printOuterStructure(int frames)
{
  printf("Incoming ");

  for (int i = 0; i < frames; i++)
    printf("\tFrame%d\t", i + 1);
}
void printCurrFrames(int item, int frameItems[], int frame_occupied, int frames)
{
  printf("\n%d \t\t", item);
  for (int i = 0; i < frames; i++)
  {
    if (i < frame_occupied)
      printf("%d \t\t", frameItems[i]);
    else
      printf("- \t\t");
  }
}
int predict(int incomingStream[], int frameItems[], int page, int index, int frame_occupied)
{
  int result = -1, farthest = index;
  for (int i = 0; i < frame_occupied; i++)
  {
    int j;
    for (j = index; j < page; j++)
    {
      if (frameItems[i] == incomingStream[j])
      {
        if (j > farthest)
        {
          farthest = j;
          result = i;
        }
        break;
      }
    }

    if (j == page)
      return i;
  }

  return (result == -1) ? 0 : result;
}

void optimalPage(int incomingStream[], int page, int frameItems[], int frames)
{
  int frame_occupied = 0;
  printOuterStructure(frames);

  int hits = 0;
  for (int i = 0; i < page; i++)
  {
    if (search(incomingStream[i], frameItems, frame_occupied))
    {
      hits++;
      printCurrFrames(incomingStream[i], frameItems, frame_occupied, frames);
      continue;
    }
    if (frame_occupied < frames)
    {
      frameItems[frame_occupied] = incomingStream[i];
      frame_occupied++;
      printCurrFrames(incomingStream[i], frameItems, frame_occupied, frames);
    }
    else
    {
      int pos = predict(incomingStream, frameItems, page, i + 1, frame_occupied);
      frameItems[pos] = incomingStream[i];
      printCurrFrames(incomingStream[i], frameItems, frame_occupied, frames);
    }
  }
  printf("\n\nHits: %d\n", hits);
  printf("Misses: %d", page - hits);
}

int main()
{
  int incomingStream[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};
  int page = sizeof(incomingStream) / sizeof(incomingStream[0]);
  int frames = 4;
  int frameItems[frames];

  optimalPage(incomingStream, page, frameItems, frames);
  return 0;
}
```