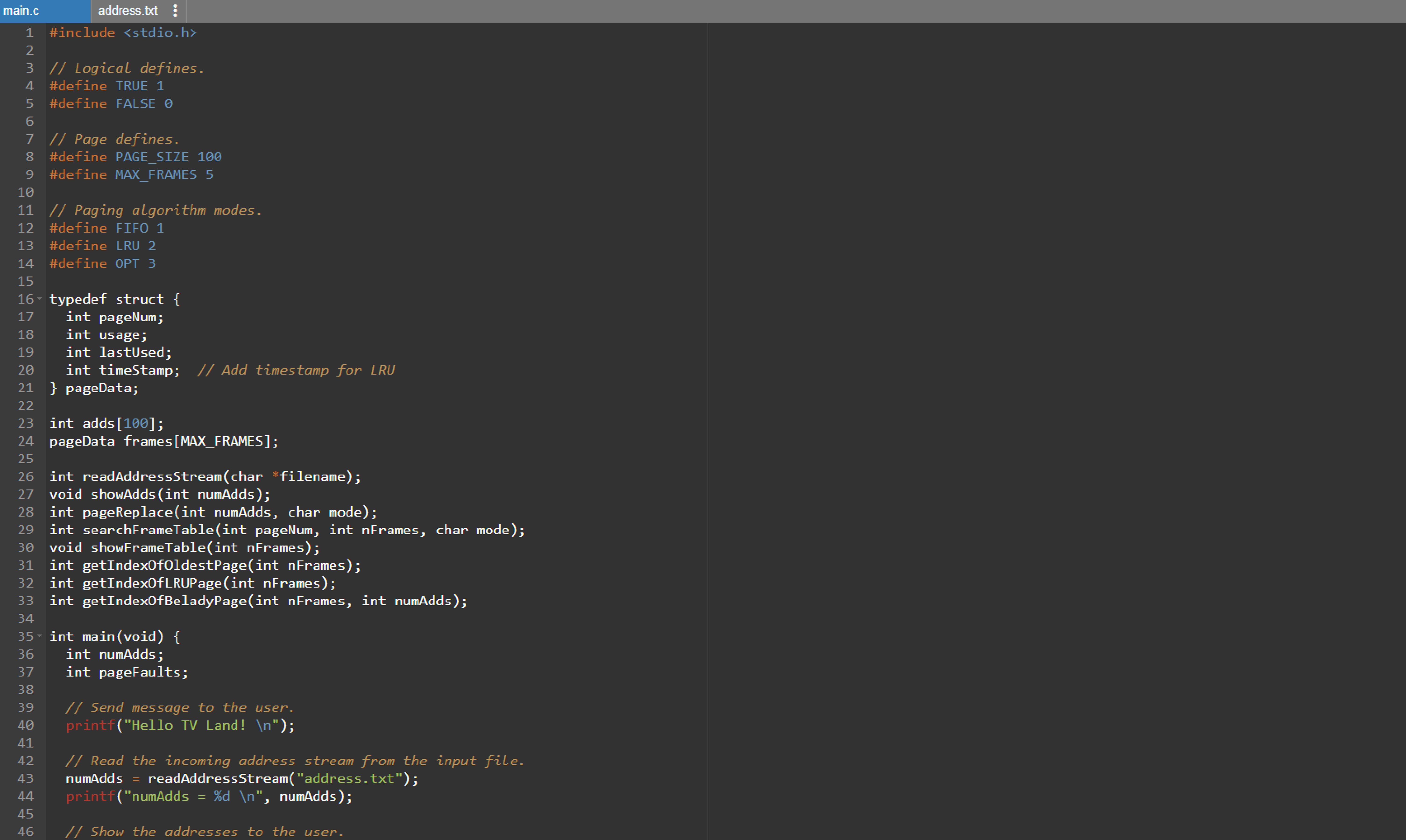
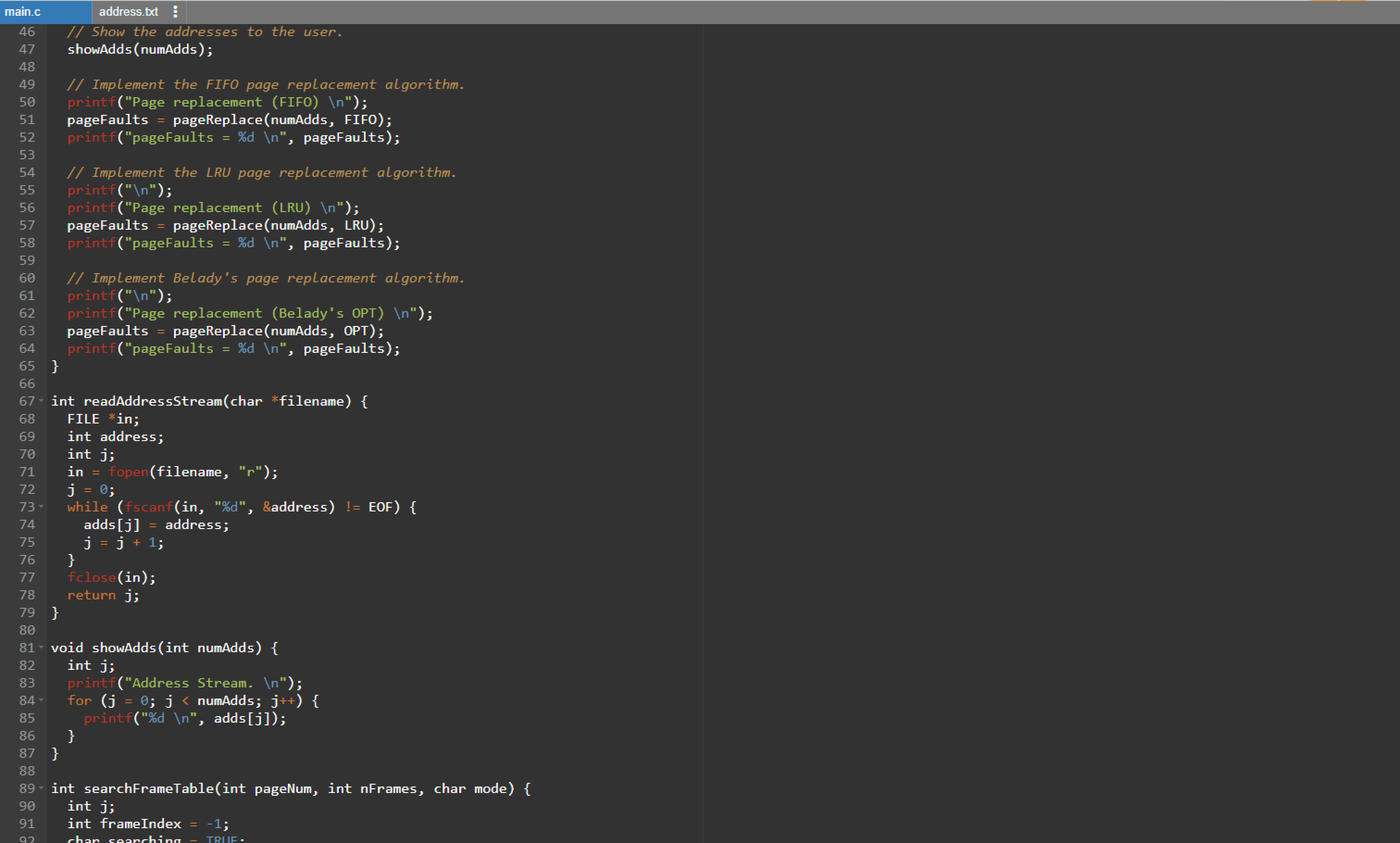
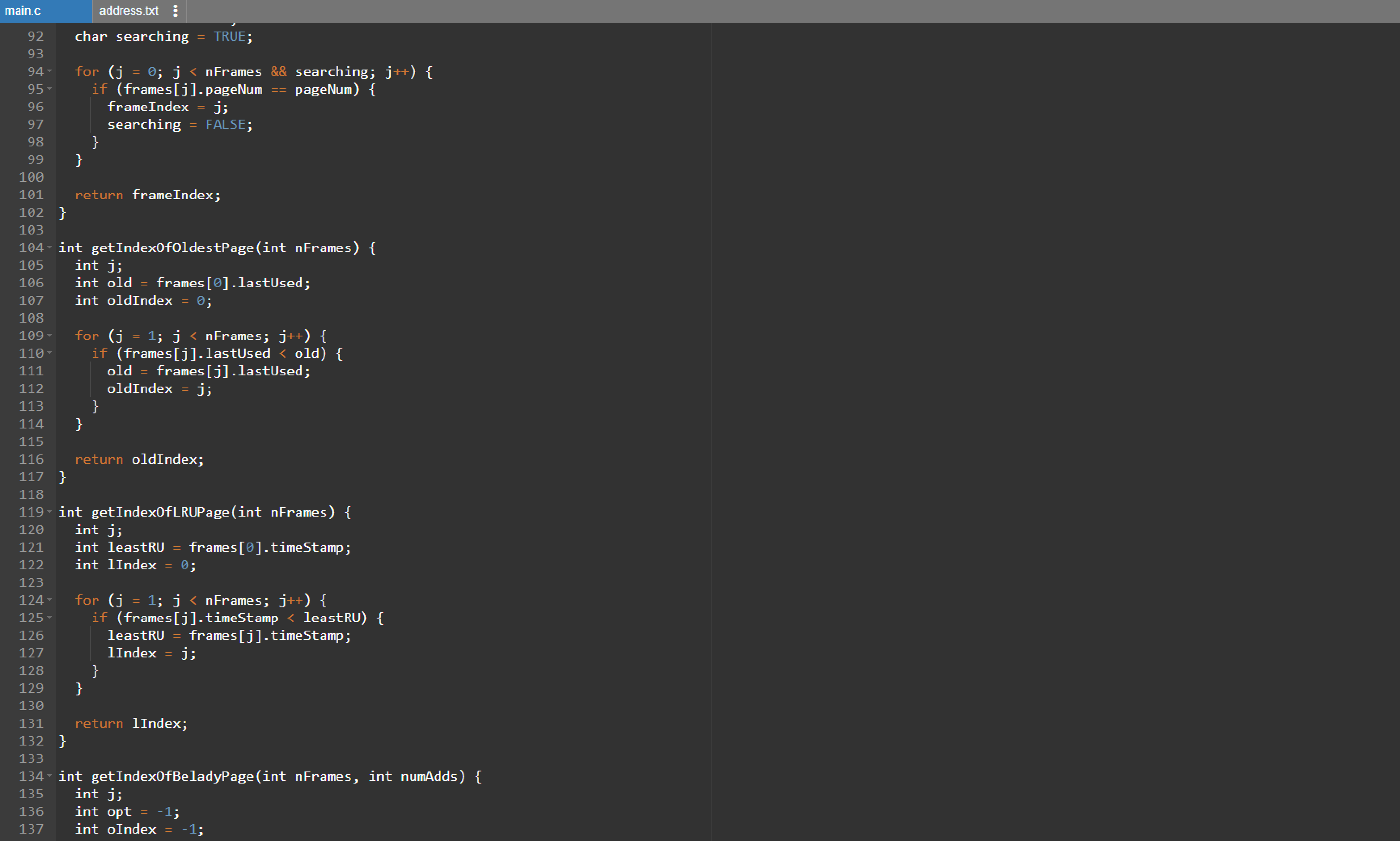


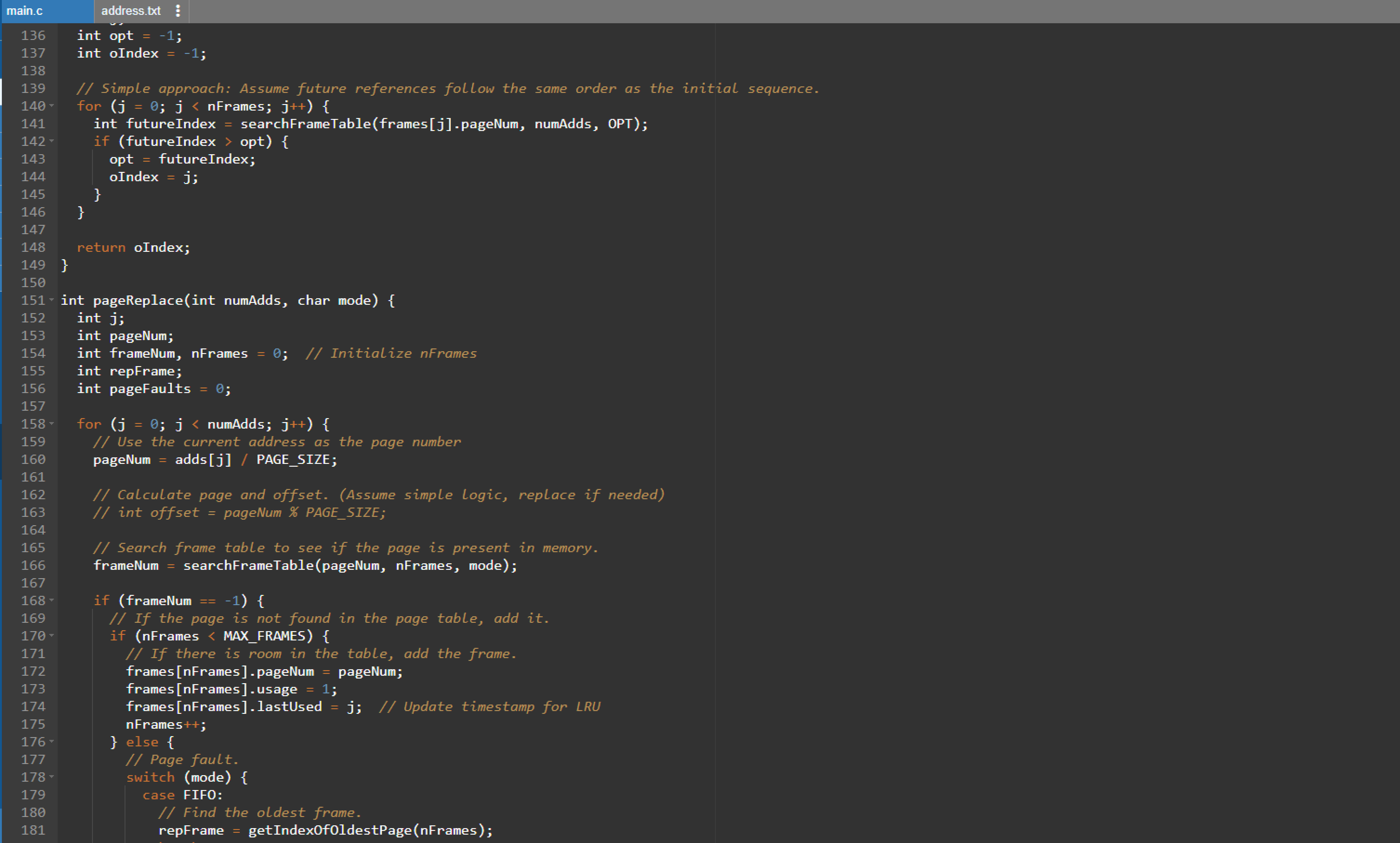
```c
#include <stdio.h>

// Logical defines.
#define TRUE 1
#define FALSE 0

// Page defines.
#define PAGE_SIZE 100
#define MAX_FRAMES 5

// Paging algorithm modes.
#define FIFO 1
#define LRU 2
#define OPT 3

typedef struct {
  int pageNum;
  int usage;
  int lastUsed;
  int timeStamp;  // Add timestamp for LRU
} pageData;

int adds[100];
pageData frames[MAX_FRAMES];

int readAddressStream(char *filename);
void showAdds(int numAdds);
int pageReplace(int numAdds, char mode);
int searchFrameTable(int pageNum, int nFrames, char mode);
void showFrameTable(int nFrames);
int getIndexOfOldestPage(int nFrames);
int getIndexOfLRUPage(int nFrames);
int getIndexOfBeladyPage(int nFrames, int numAdds);

int main(void) {
  int numAdds;
  int pageFaults;

  // Send message to the user.
  printf("Hello TV Land! \n");

  // Read the incoming address stream from the input file.
  numAdds = readAddressStream("address.txt");
  printf("numAdds = %d \n", numAdds);

  // Show the addresses to the user.
```



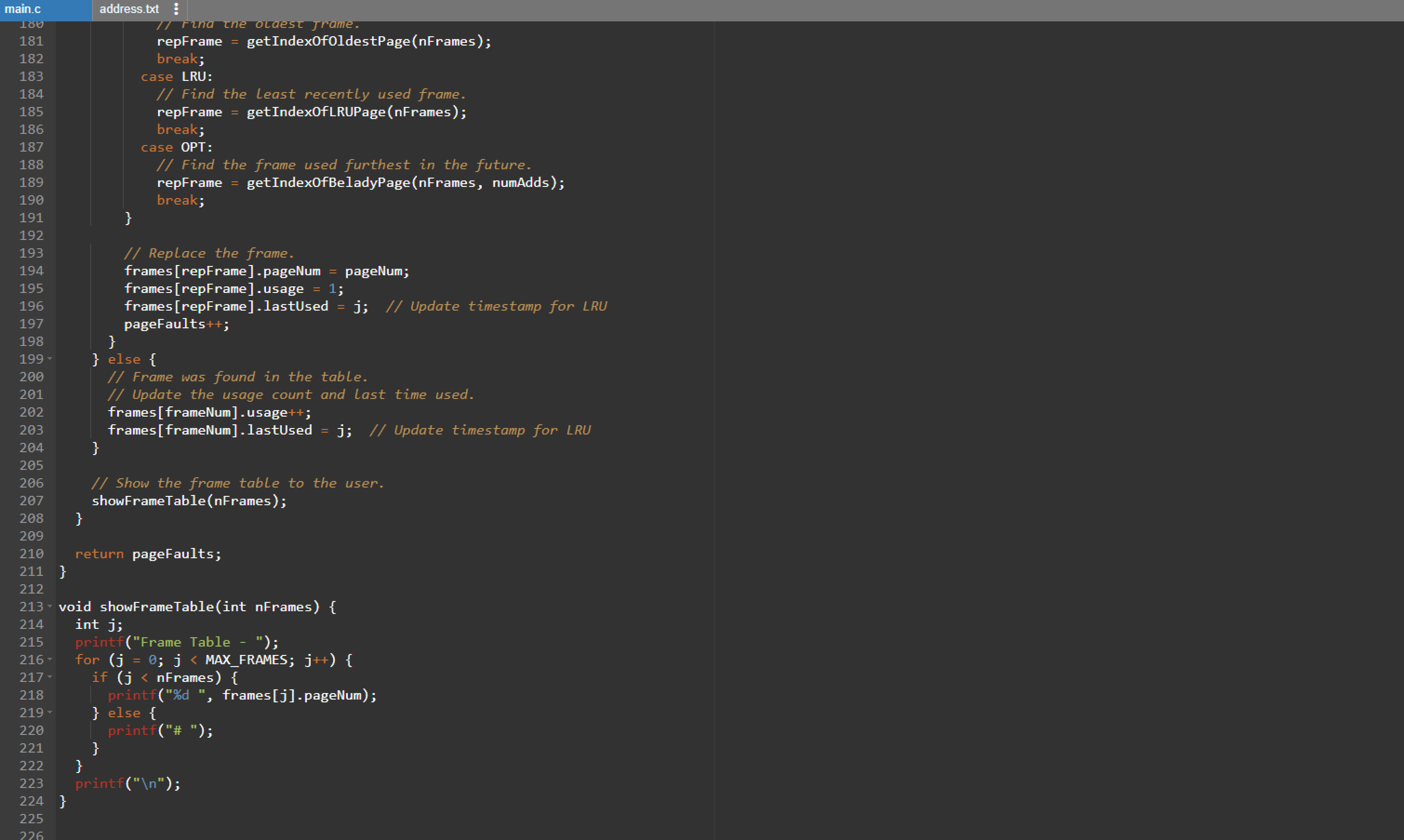
```c
  // Show the addresses to the user.
  showAdds(numAdds);

  // Implement the FIFO page replacement algorithm.
  printf("Page replacement (FIFO) \n");
  pageFaults = pageReplace(numAdds, FIFO);
  printf("pageFaults = %d \n", pageFaults);

  // Implement the LRU page replacement algorithm.
  printf("\n");
  printf("Page replacement (LRU) \n");
  pageFaults = pageReplace(numAdds, LRU);
  printf("pageFaults = %d \n", pageFaults);

  // Implement Belady's page replacement algorithm.
  printf("\n");
  printf("Page replacement (Belady's OPT) \n");
  pageFaults = pageReplace(numAdds, OPT);
  printf("pageFaults = %d \n", pageFaults);
}

int readAddressStream(char *filename) {
  FILE *in;
  int address;
  int j;
  in = fopen(filename, "r");
  j = 0;
  while (fscanf(in, "%d", &address) != EOF) {
    adds[j] = address;
    j = j + 1;
  }
  fclose(in);
  return j;
}

void showAdds(int numAdds) {
  int j;
  printf("Address Stream. \n");
  for (j = 0; j < numAdds; j++) {
    printf("%d \n", adds[j]);
  }
}

int searchFrameTable(int pageNum, int nFrames, char mode) {
  int j;
  int frameIndex = -1;
  char searching = TRUE;
```

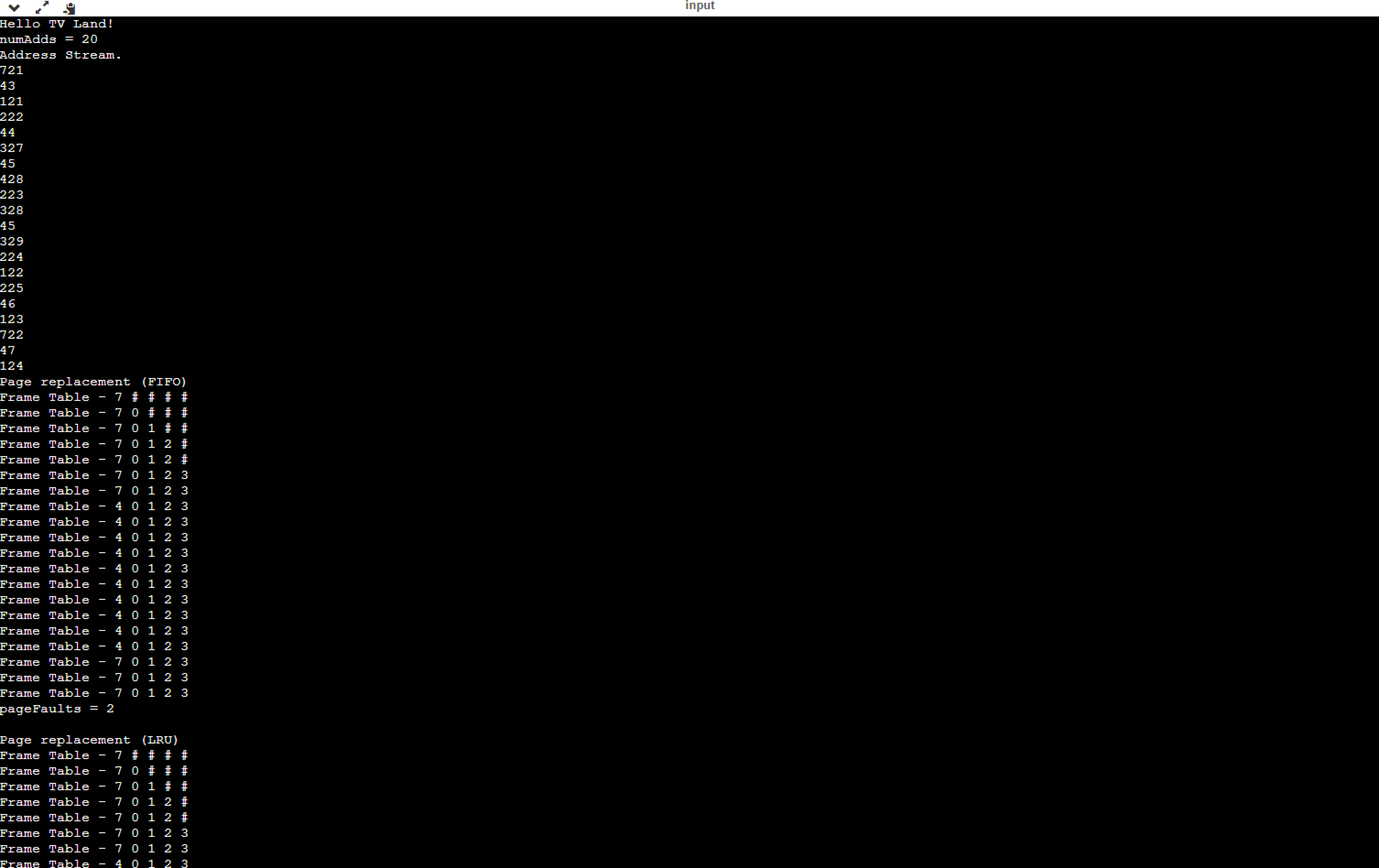


```c
  char searching = TRUE;

  for (j = 0; j < nFrames && searching; j++) {
    if (frames[j].pageNum == pageNum) {
      frameIndex = j;
      searching = FALSE;
    }
  }

  return frameIndex;
}

int getIndexOfOldestPage(int nFrames) {
  int j;
  int old = frames[0].lastUsed;
  int oldIndex = 0;

  for (j = 1; j < nFrames; j++) {
    if (frames[j].lastUsed < old) {
      old = frames[j].lastUsed;
      oldIndex = j;
    }
  }

  return oldIndex;
}

int getIndexOfLRUPage(int nFrames) {
  int j;
  int leastRU = frames[0].timeStamp;
  int lIndex = 0;

  for (j = 1; j < nFrames; j++) {
    if (frames[j].timeStamp < leastRU) {
      leastRU = frames[j].timeStamp;
      lIndex = j;
    }
  }

  return lIndex;
}

int getIndexOfBeladyPage(int nFrames, int numAdds) {
  int j;
  int opt = -1;
  int oIndex = -1;
```

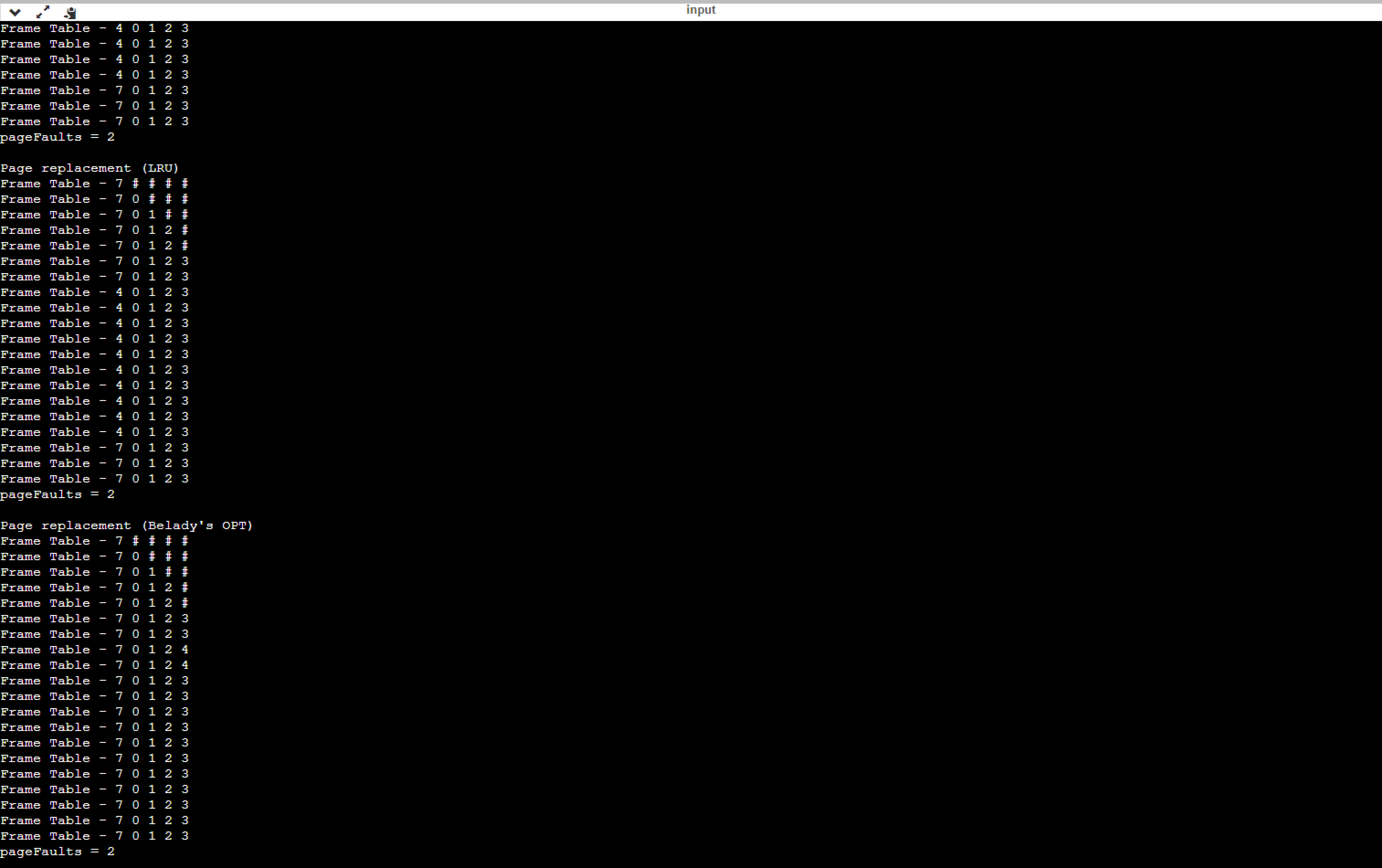


```c
  int opt = -1;
  int oIndex = -1;

  // Simple approach: Assume future references follow the same order as the initial sequence.
  for (j = 0; j < nFrames; j++) {
    int futureIndex = searchFrameTable(frames[j].pageNum, numAdds, OPT);
    if (futureIndex > opt) {
      opt = futureIndex;
      oIndex = j;
    }
  }

  return oIndex;
}

int pageReplace(int numAdds, char mode) {
  int j;
  int pageNum;
  int frameNum, nFrames = 0;  // Initialize nFrames
  int repFrame;
  int pageFaults = 0;

  for (j = 0; j < numAdds; j++) {
    // Use the current address as the page number
    pageNum = adds[j] / PAGE_SIZE;

    // Calculate page and offset. (Assume simple logic, replace if needed)
    // int offset = pageNum % PAGE_SIZE;

    // Search frame table to see if the page is present in memory.
    frameNum = searchFrameTable(pageNum, nFrames, mode);

    if (frameNum == -1) {
      // If the page is not found in the page table, add it.
      if (nFrames < MAX_FRAMES) {
        // If there is room in the table, add the frame.
        frames[nFrames].pageNum = pageNum;
        frames[nFrames].usage = 1;
        frames[nFrames].lastUsed = j;  // Update timestamp for LRU
        nFrames++;
      } else {
        // Page fault.
        switch (mode) {
          case FIFO:
            // Find the oldest frame.
            repFrame = getIndexOfOldestPage(nFrames);
```


```c
            // Find the oldest frame.
            repFrame = getIndexOfOldestPage(nFrames);
            break;
          case LRU:
            // Find the least recently used frame.
            repFrame = getIndexOfLRUPage(nFrames);
            break;
          case OPT:
            // Find the frame used furthest in the future.
            repFrame = getIndexOfBeladyPage(nFrames, numAdds);
            break;
        }

        // Replace the frame.
        frames[repFrame].pageNum = pageNum;
        frames[repFrame].usage = 1;
        frames[repFrame].lastUsed = j;  // Update timestamp for LRU
        pageFaults++;
      }
    } else {
      // Frame was found in the table.
      // Update the usage count and last time used.
      frames[frameNum].usage++;
      frames[frameNum].lastUsed = j;  // Update timestamp for LRU
    }

    // Show the frame table to the user.
    showFrameTable(nFrames);
  }

  return pageFaults;
}

void showFrameTable(int nFrames) {
  int j;
  printf("Frame Table - ");
  for (j = 0; j < MAX_FRAMES; j++) {
    if (j < nFrames) {
      printf("%d ", frames[j].pageNum);
    } else {
      printf("# ");
    }
  }
  printf("\n");
}
```



```
Hello TV Land!
numAdds = 20
Address Stream.
721
43
121
222
44
327
45
428
223
328
45
329
224
122
225
46
123
722
47
124
Page replacement (FIFO)
Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2

Page replacement (LRU)
Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 4 0 1 2 3
```



```
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2

Page replacement (LRU)
Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 4 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2

Page replacement (Belady's OPT)
Frame Table - 7 # # # #
Frame Table - 7 0 # # #
Frame Table - 7 0 1 # #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 #
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 4
Frame Table - 7 0 1 2 4
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
Frame Table - 7 0 1 2 3
pageFaults = 2
```