LAB 9

**YASHDEEP ROY 22BCE1532**

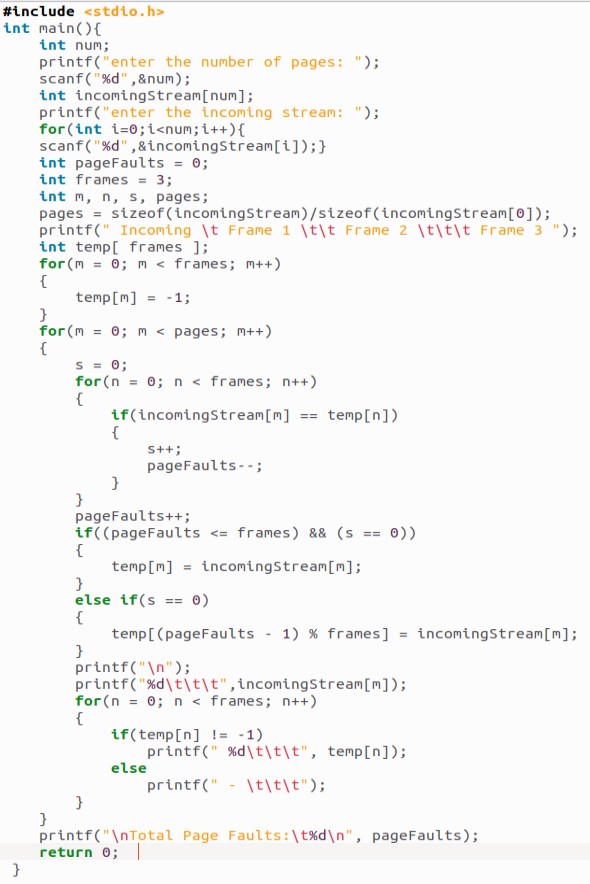
**Aim:**To write c code to implement page replacement algorithms(FIFO,OPR and LRU).

1. **FIFO:**

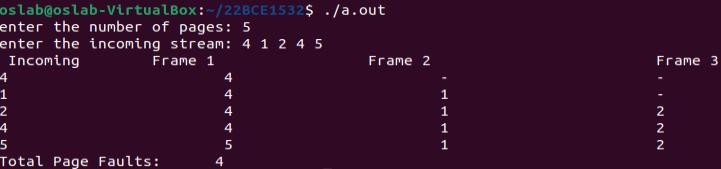
**Algorithm:**

* 1. Initialize Variables:
     + Declare variables for the number of pages (num), the incoming stream of pages (incomingStream), the number of page faults (pageFaults), the number of frames available (frames), and temporary storage for frames (temp).
  2. Input the Number of Pages and Incoming Stream:
     + Prompt the user to enter the number of pages and the incoming stream of pages.
     + Store the incoming stream in an array (incomingStream).
  3. Initialize Frames:
     + Create an array temp to simulate the frames with a size equal to the number of frames available.
     + Initialize all elements of temp to -1, indicating that the frames are initially empty.
  4. Process Each Page in the Incoming Stream:
     + For each page in the incoming stream:
       - Initialize a counter s to track if the page is already in the frames.
       - Check if the current page is already in the frames:
         * If it is, increment s and decrement pageFaults (this step seems incorrect in the context of FIFO, as it should not decrease the page fault count; it's likely a mistake in the code).
       - Increment pageFaults to account for the current page being processed.
       - If the number of page faults is less than or equal to the number of frames and the page is not already in the frames (s == 0):
         * Add the current page to the next available frame in temp.
       - If the page is not in the frames and there's no space left in the frames:
         * Replace the oldest page (the first one in temp) with the current page.
  5. Display the State of Frames After Each Page Processing:
     + After processing each page, display the current page and the state of the frames.
  6. Calculate and Display Total Page Faults:
     + After processing all pages, calculate the total number of page faults.
     + Display the total number of page faults.

**Code:**



**Output:**

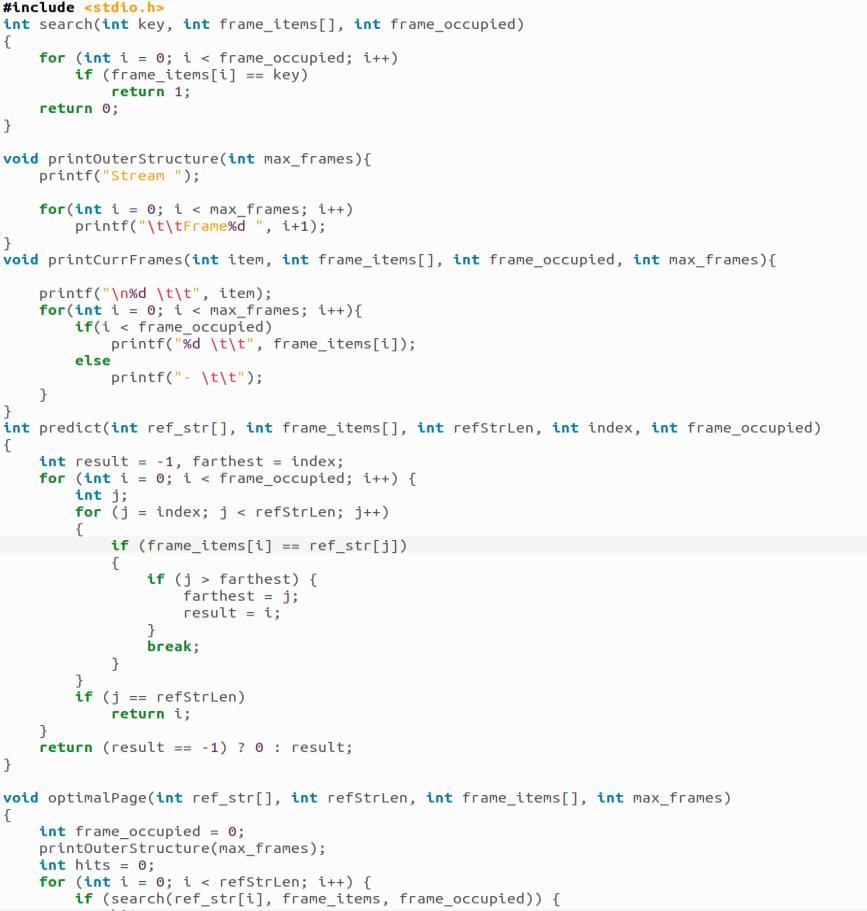


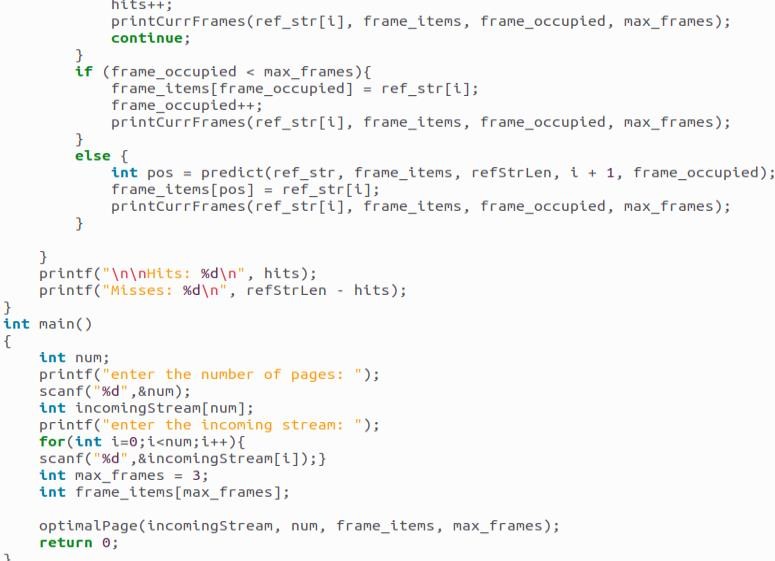
1. **OPR:**

**Algorithm:**

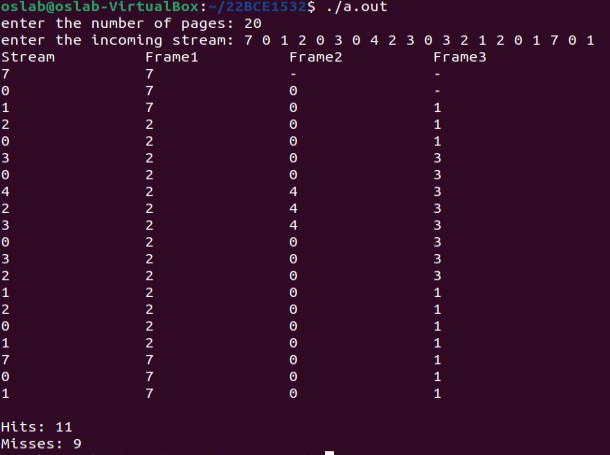
* 1. Initialize Variables:
     + Declare variables for the number of pages (num), the incoming stream of pages (incomingStream), the maximum number of frames available (max\_frames), and an array to store the current pages in the frames (frame\_items).
  2. Input the Number of Pages and Incoming Stream:
     + Prompt the user to enter the number of pages and the incoming stream of pages.
     + Store the incoming stream in an array (incomingStream).
  3. Initialize Frames:
     + Create an array frame\_items to simulate the frames with a size equal to the maximum number of frames available.
     + Initialize all elements of frame\_items to indicate that the frames are initially empty.
  4. Process Each Page in the Incoming Stream:
     + For each page in the incoming stream:
       - Check if the current page is already in the frames using the search function.
         * If it is, increment the hit counter and continue to the next page.
       - If the current page is not in the frames and there's space left in the frames:
         * Add the current page to the next available frame in frame\_items.
       - If the page is not in the frames and there's no space left in the frames:
         * Use the predict function to find the page that will not be used for the longest time in the future.
         * Replace the identified page with the current page in frame\_items.
  5. Display the State of Frames After Each Page Processing:
     + After processing each page, display the current page and the state of the frames using the printCurrFrames function.
  6. Calculate and Display Total Hits and Misses:
     + After processing all pages, calculate the total number of hits (pages found in the frames) and misses (pages not found in the frames).
     + Display the total number of hits and misses.

**Code:**





**Output:**



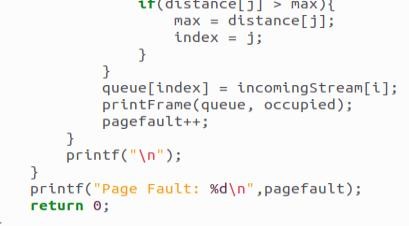
1. **LRU**

**Algorithm:**

* Initialize an array of frames with a size equal to the number of frames available.
* For each page in the sequence:
* Check if the page is already in the frames.
* If it is, update its recency (set it to the current page number).
* If it's not, and there's space in the frames:
* Add the page to the next available frame.
* Increment the page fault counter.
* If there's no space in the frames, find the page with the lowest recency (the least recently used page) and replace it with the new page.

**Code:**





**Output:**

